

# The sTeX3 Package Collection \*

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

2022-09-26

## 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 2022-09-26)

# Contents

<b>I</b>	<b>Manual</b>	<b>1</b>
<b>1</b>	<b>What is sTeX?</b>	<b>2</b>
<b>2</b>	<b>Setup</b>	<b>3</b>
2.1	Setting up the sTeX Package . . . . .	3
2.1.1	Minimal Setup for the sTeX Package . . . . .	3
2.1.2	GIT-based Setup for the sTeX Development Version . . . . .	3
2.1.3	Setting your MathHub Directory . . . . .	4
2.2	Setting up the sTeX IDE . . . . .	4
2.2.1	The sTeX VSCode Extension . . . . .	4
2.2.2	Setting up MMT . . . . .	4
2.3	Manual Setup . . . . .	6
2.3.1	sTeX Archives (Manual Setup) . . . . .	6
2.3.2	Manual Setup for Active Documents and Knowledge Management Services . . . . .	6
<b>3</b>	<b>The sTeX IDE</b>	<b>7</b>
<b>4</b>	<b>A First sTeX Document</b>	<b>8</b>
4.1	OMDOC/xhtml Conversion . . . . .	11
4.2	MMT/OMDOC Conversion . . . . .	12
<b>5</b>	<b>Creating sTeX Content</b>	<b>14</b>
5.1	How Knowledge is Organized in sTeX . . . . .	14
5.2	sTeX Archives . . . . .	15
5.2.1	The Local MathHub-Directory . . . . .	15
5.2.2	The Structure of sTeX Archives . . . . .	16
5.2.3	MANIFEST.MF-Files . . . . .	16
5.2.4	Using Files in sTeX Archives Directly . . . . .	17
5.3	Module, Symbol and Notation Declarations . . . . .	18
5.3.1	The <code>smodule</code> -Environment . . . . .	18
5.3.2	Declaring New Symbols and Notations . . . . .	20
	Operator Notations . . . . .	24
5.3.3	Argument Modes . . . . .	24
	Mode- <code>b</code> Arguments . . . . .	24
	Mode- <code>a</code> Arguments . . . . .	25
	Mode- <code>B</code> Arguments . . . . .	26
5.3.4	Type and Definiens Components . . . . .	27
5.3.5	Precedences and Automated Bracketing . . . . .	28
5.3.6	Variables . . . . .	30
5.3.7	Variable Sequences . . . . .	31
5.4	Module Inheritance and Structures . . . . .	33
5.4.1	Multilinguality and Translations . . . . .	33
5.4.2	Simple Inheritance and Namespaces . . . . .	34
5.4.3	The <code>mathstructure</code> Environment . . . . .	36
5.4.4	The <code>copymodule</code> Environment . . . . .	39

5.4.5	The <code>interpretmodule</code> Environment	40
5.5	Primitive Symbols (The <code>sTeX</code> Metatheory)	41
<b>6</b>	<b>Using <code>sTeX</code> Symbols</b>	<b>42</b>
6.1	<code>\symref</code> and its variants	42
6.2	Marking Up Text and On-the-Fly Notations	43
<b>7</b>	<b><code>sTeX</code> Statements</b>	<b>47</b>
7.1	Definitions, Theorems, Examples, Paragraphs	47
7.2	Proofs	50
7.3	Highlighting and Presentation Customizations	55
<b>8</b>	<b>Cross References</b>	<b>57</b>
<b>9</b>	<b>Additional Packages</b>	<b>59</b>
9.1	Tikzinput: Treating TIKZ code as images	59
9.2	Modular Document Structuring	60
9.2.1	Introduction	60
9.2.2	Package Options	60
9.2.3	Document Fragments	60
9.2.4	Ending Documents Prematurely	62
9.2.5	Global Document Variables	62
9.3	Slides and Course Notes	62
9.3.1	Introduction	62
9.3.2	Package Options	63
9.3.3	Notes and Slides	63
9.3.4	Customizing Header and Footer Lines	64
9.3.5	Frame Images	65
9.3.6	Excursions	66
9.4	Representing Problems and Solutions	67
9.4.1	Introduction	67
9.4.2	Problems and Solutions	67
9.4.3	Markup for Added-Value Services	69
	Multiple Choice Blocks	69
	Filling-In Concrete Solutions	70
9.4.4	Including Problems	71
9.4.5	Testing and Spacing	72
9.5	Homeworks, Quizzes and Exams	72
9.5.1	Introduction	72
9.5.2	Package Options	72
9.5.3	Assignments	73
9.5.4	Including Assignments	73
9.5.5	Typesetting Exams	73
<b>II</b>	<b>Documentation</b>	<b>75</b>

<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>153</b>
30.1	SMS Mode . . . . .	153
30.2	Inheritance . . . . .	157
<b>31</b>	<b>STEX-Symbols Implementation</b>	<b>163</b>
31.1	Symbol Declarations . . . . .	163
31.2	Notations . . . . .	171
31.3	Variables . . . . .	181
<b>32</b>	<b>STEX-Terms Implementation</b>	<b>190</b>
32.1	Symbol Invocations . . . . .	190
32.2	Terms . . . . .	198
32.3	Notation Components . . . . .	203
32.4	Variables . . . . .	205
32.5	Sequences . . . . .	208

<b>33</b>	<b>STEX-Structural Features Implementation</b>	<b>209</b>
33.1	Imports with modification . . . . .	210
33.2	The feature environment . . . . .	218
33.3	Structure . . . . .	218
<b>34</b>	<b>STEX-Statements Implementation</b>	<b>230</b>
34.1	Definitions . . . . .	230
34.2	Assertions . . . . .	236
34.3	Examples . . . . .	239
34.4	Logical Paragraphs . . . . .	242
<b>35</b>	<b>The Implementation</b>	<b>247</b>
35.1	Proofs . . . . .	247
<b>36</b>	<b>STEX-Others Implementation</b>	<b>256</b>
<b>37</b>	<b>STEX-Metatheory Implementation</b>	<b>258</b>
<b>38</b>	<b>Tikzinput Implementation</b>	<b>261</b>
<b>39</b>	<b>document-structure.sty Implementation</b>	<b>264</b>
39.1	Package Options . . . . .	264
39.2	Document Structure . . . . .	265
39.3	Front and Backmatter . . . . .	269
39.4	Global Variables . . . . .	271
<b>40</b>	<b>NotesSlides – Implementation</b>	<b>272</b>
40.1	Class and Package Options . . . . .	272
40.2	Notes and Slides . . . . .	274
40.3	Header and Footer Lines . . . . .	278
40.4	Frame Images . . . . .	280
40.5	Sectioning . . . . .	281
40.6	Excursions . . . . .	284
<b>41</b>	<b>The Implementation</b>	<b>286</b>
41.1	Package Options . . . . .	286
41.2	Problems and Solutions . . . . .	287
41.3	Markup for Added Value Services . . . . .	294
41.4	Multiple Choice Blocks . . . . .	294
41.5	Filling in Concrete Solutions . . . . .	295
41.6	Including Problems . . . . .	296
41.7	Reporting Metadata . . . . .	297
41.8	Testing and Spacing . . . . .	298
<b>42</b>	<b>Implementation: The hwexam Package</b>	<b>300</b>
42.1	Package Options . . . . .	300
42.2	Assignments . . . . .	301
42.3	Including Assignments . . . . .	304
42.4	Typesetting Exams . . . . .	305
42.5	Leftovers . . . . .	307



# 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 sTeX’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 sTeX system needs to be told where to find them. There are several ways to do so (see [subsection 5.2.1](#)), 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 sTeX IDE

The sTeX 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 sTeX 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 sTeX extension by clicking on the *extensions* menu on the very left of the VSCode window and searching for “sTeX” in the “*Search Extensions in Marketplace*” field, as in [Figure 1](#), and clicking the *Install*-button of the sTeX extension by KWARC.

### 2.2.2 Setting up Mmt

Next, open any directory (**File** → **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 RuSTeX for you automatically.

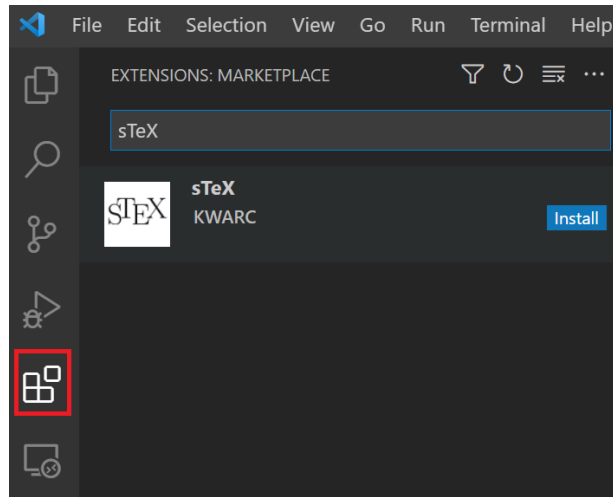


Figure 1: Installing the sTeX extension for VSCode

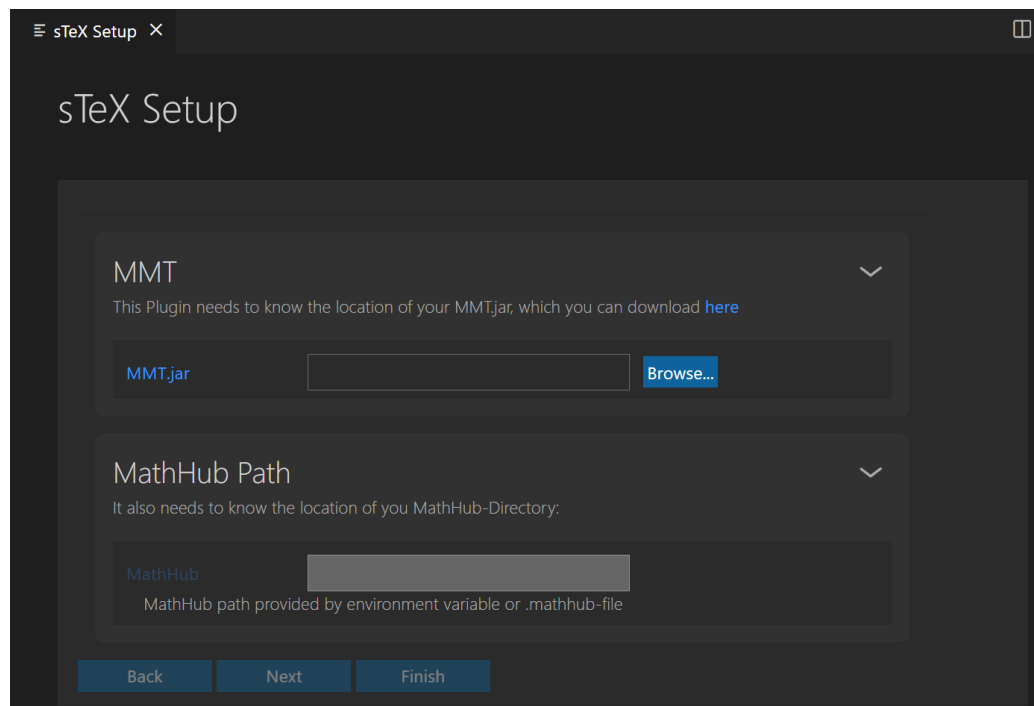


Figure 2: sTeX Setup Routine

## 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 5.2](#)).

```
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 \define{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 7.3](#).

Let’s investigate this document in detail to understand the respective parts of the  $\text{\TeX}$  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 \importmodule[smglom/arithmetics]{realarith}
```

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

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

---

`\define` The `\define{geometricSeries} ...`  
`\definiendum` The `sdefinition`-environment provides two additional macros, `\define` and `\definiendum` which behave similarly to `\symname` and `\symref`, but explicitly mark the symbols as *being defined* 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  $a/b$ .

---

`\svar` The `\svar{n}` command marks up the `n` as a variable with name `n` and notation `n`.

---



---

`\definiens` The `sdefinition`-environment additionally provides the `\definiens`-command, which allows for explicitly marking up its argument as the *definiens* 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>
      <mi resource="...?series?infinitesum" property="stex:comp">\infty</mi>
    </munderover>
    <mrow resource="3" property="stex:arg">
      <mfrac resource="...?realarith?division#frac#" property="stex:OMA">
        <mi resource="1" property="stex:arg">1</mi>
        <mrow resource="2" property="stex:arg">
          <msup resource="...realarith?exponentiation" property="stex:OMA">

```

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

```

<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 `TEX` archive (see [section 5.2](#)) and have MMT take care of everything.

Assuming the above file is `source/demo.tex` in an `TEX` 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](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 5.2](#)) 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 `\symsdecl{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
```

<code>teaser: Terminology for the mathematical study of change.</code> <code>description: desc.html</code>
---

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  $\text{\LaTeX}$  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.

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

When  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  steps into an argument of a semantic macro, it sets  $p_d$  to the respective argument precedence of the notation used.

In the example above:



1.  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  starts out with  $p_d = \text{\texttt{\textbackslash infprec}}$ .
2.  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  encounters `\addition` with  $p_{op} = 100$ . Since  $100 \not> \text{\texttt{\textbackslash infprec}}$ , it inserts no parentheses.
3. Next,  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  encounters the two arguments for `\addition`. Both have no specifically provided argument precedence, so  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  uses  $p_d = p_{op} = 100$  for both and recurses.
4. Next,  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  encounters `\multiplication{b,...}`, whose notation has  $p_{op} = 50$ .
5. We compare to the current downward precedence  $p_d$  set by `\addition`, arriving at  $p_{op} = 50 \not> 100 = p_d$ , so  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  again inserts no parentheses.
6. Since the notation of `\multiplication` has no explicitly set argument precedences,  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  uses the operator precedence for all arguments of `\multiplication`, hence sets  $p_d = p_{op} = 50$  and recurses.
7. Next,  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  encounters the inner `\addition{c,...}` whose notation has  $p_{op} = 100$ .
8. We compare to the current downward precedence  $p_d$  set by `\multiplication`, arriving at  $p_{op} = 100 > 50 = p_d$  – which finally prompts  $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  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  $\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$  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;\neginfpref
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  $\hookrightarrow M \rightarrow$  `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)`
- $\hookrightarrow M \rightarrow$  – a *dependent record type with manifest fields*, the fields of which are generated
- $\rightsquigarrow T \rightsquigarrow$  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 $\varM{op}!: \funtype{\varM{universe}, \varM{universe}}{\varM{universe}}$ ...

```

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 9.2.1](#)) 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,  $\text{\TeX}$  can decide based on the current document

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

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 *<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  $\text{\TeX}$ 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}3 document]
```

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\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 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 7.3](#) 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 8](#)), `type=` for customization (see [section 7.3](#)) 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   $\text{\addition{2,3}}$ is $5$, $\text{\multiplication{2,3}}$ 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.

---

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

---

`\definiens` Additionally, **sdefinition** (and **sparagraph** with `type=symdoc`) introduces `\definiens`[<optional sym 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.

---

**\spfidea** The **\spfidea** macro allows to give a one-paragraph description of the proof idea.

---

**\spfsketch** For one-line proof sketches, we use the **\spfsketch** macro, which takes the 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*`.

---

<code>\stexpatchmodule</code> <code>\stexpatchdefinition</code> <code>\stexpatchassertion</code> <code>\stexpatchexample</code> <code>\stexpatchparagraph</code> <code>\stexpatchproof</code>	<p>All of these commands take one optional and two proper arguments, i.e.</p> <pre>\stexpatch* [&lt;type&gt;] {&lt;begin-code&gt;} {&lt;end-code&gt;}</pre> <p>After <math>\text{\LaTeX}</math> reads and processes the optional arguments for these environments, (some of) their values are stored in the macros <code>\s*&lt;field&gt;</code> (i.e. <code>sexampleid</code>, <code>\sassertionname</code>, etc.). It then checks for all the values <code>&lt;type&gt;</code> in the <code>type=</code>-list, whether an <code>\stexpatch* [&lt;type&gt;]</code> for the current environment has been called. If it finds one, it uses the patches <code>&lt;begin-code&gt;</code> and <code>&lt;end-code&gt;</code> to mark up the current environment. If no patch for (any of) the type(s) is found, it checks whether and <code>\stexpatch*</code> was called without optional argument.</p>
--	---

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 9.2.1](#)) 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=&lt;name&gt;</code>	load <code>&lt;name&gt;.cls</code> instead of <code>article.cls</code>
<code>topsect=&lt;sect&gt;</code>	The top-level sectioning level; the default for <code>&lt;sect&gt;</code> is <code>section</code>

### 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:

<code>slides</code>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see <a href="#">subsection 9.3.3</a> ).
<code>notes</code>	
<code>sectocframes</code>	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.
<code>frameimages</code>	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.
<code>fiboxed</code>	

### 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 L<sup>A</sup>T<sub>E</sub>X 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

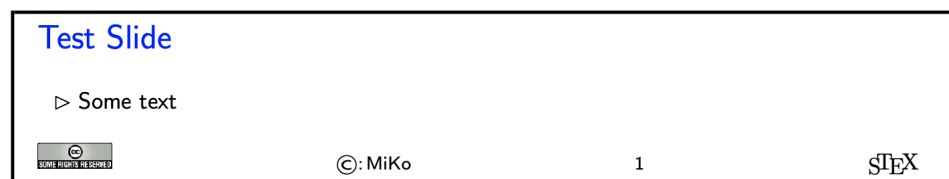
`nexample` (*env.*) level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,

`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`,  
`notes`      `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation  
`hints`      for a description of the intended behavior).  
`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)

2022-09-26

**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

## ST<sub>E</sub>X-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> XML
--------------------------	---

---

---

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

---

---

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

---

---

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

---

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

---

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

---

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

# Chapter 12

## STEX-References

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

### 12.1 Macros and Environments

---

<code>\stex_get_document_uri:</code>	Computes the current document uri from the current archive's <b>narr</b> -field and its location relative to the archive's <b>source</b> -directory. Reference targets are computed from this URI and the reference-id.
--------------------------------------	---

---

<code>\l_stex_current_docns_str</code>	Stores its result in <code>\l_stex_current_docns_str</code>
--	---

---

<code>\stex_get_document_url:</code>	Computes the current URL from the current archive's <b>docurl</b> -field and its location relative to the archive's <b>source</b> -directory. Reference targets are computed from this URL and the reference-id, if this document is only included in SMS mode.
--------------------------------------	---

---

<code>\l_stex_current_docurl_str</code>	Stores its result in <code>\l_stex_current_docurl_str</code>
---	--

#### 12.1.1 Setting Reference Targets

---

<code>\stex_ref_new_doc_target:n</code>	<code>\stex_ref_new_doc_target:n{&lt;id&gt;}</code> Sets a new reference target with id <code>&lt;id&gt;</code> .
---	--

---

<code>\stex_ref_new_sym_target:n</code>	<code>\stex_ref_new_sym_target:n{&lt;uri&gt;}</code> Sets a new reference target for the symbol <code>&lt;uri&gt;</code> .
---	---



### 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> <hr/>	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> <hr/>	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> <hr/>	The names of all constants declared in the module
---	---

<hr/> <hr/> <code>\c_stex_module_&lt;URI&gt;_constants</code> <hr/>	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.

---

---

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

## Chapter 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

---

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

---

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

---

$\backslash\text{l\_stex\_import\_name\_str}$ $\backslash\text{l\_stex\_import\_archive\_str}$ $\backslash\text{l\_stex\_import\_path\_str}$ $\backslash\text{l\_stex\_import\_ns\_str}$	stores the result in these four variables.
---	--

---



---

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

---

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

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



# Chapter 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

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

# sT<sub>E</sub>X-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 sTeXlogo:

**\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 \</package>

```

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

## Chapter 27

# STEX -MathHub Implementation

```
393 <*package>
394
395 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
396
397 <@@=stex_path>
398
399 Warnings and error messages
400 \msg_new:nnn{stex}{error/norepository}{
401   No~archive~#1~found~in~#2
402 }
403 \msg_new:nnn{stex}{error/notinarchive}{
404   Not~currently~in~an~archive,~but~\detokenize{#1}~
405   needs~one!
406 }
407 \msg_new:nnn{stex}{error/nofile}{
408   \detokenize{#1}~could~not~find~file~#2
409 }
410 \msg_new:nnn{stex}{error/twofiles}{
411   \detokenize{#1}~found~two~candidates~for~#2
412 }
```

### 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`

```
411 \cs_new_protected:Nn \stex_path_from_string:Nn {
412   \stex_debug:nn{files}{#2}
413   \str_set:Nx \l_tmpa_str { #2 }
414   \str_if_empty:NTF \l_tmpa_str {
415     \seq_clear:N #1
416   }{
417     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
418     \sys_if_platform_windows:T{
```

```

419     \seq_clear:N \l_tmpa_tl
420     \seq_map_inline:Nn #1 {
421         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
422         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
423     }
424     \seq_set_eq:NN #1 \l_tmpa_tl
425 }
426 \stex_path_canonicalize:N #1
427 }
428 \stex_debug:nn{files}{Yields: \stex_path_to_string:N#1}
429 }
430

```

(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
431 \cs_new_protected:Nn \stex_path_to_string:NN {
432     \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
433 }
434
435 \cs_new:Nn \stex_path_to_string:N {
436     \seq_use:Nn #1 /
437 }

```

(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
438 \str_const:Nn \c__stex_path_dot_str {.}
439 \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.

```

440 \cs_new_protected:Nn \stex_path_canonicalize:N {
441     \stex_debug:nn{paths}{canonicalizing~\seq_use:Nn #1 /}
442     \bool_set_false:N \l__stex_path_in_path_bool
443     \seq_if_empty:NF #1 {
444         \seq_clear:N \l_tmpa_seq
445         \seq_get_left:NN #1 \l_tmpa_tl
446         \str_if_empty:NT \l_tmpa_tl {
447             \seq_put_right:Nn \l_tmpa_seq {}
448         }
449         \seq_map_inline:Nn #1 {
450             \str_set:Nn \l_tmpa_tl { ##1 }
451             \str_if_eq:NnF \l_tmpa_tl \c__stex_path_dot_str {
452                 \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
453                     \bool_set_true:N \l__stex_path_in_path_bool
454                     \seq_if_empty:NNTF \l_tmpa_seq {
455                         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
456                             \c__stex_path_up_str
457                         }
458                     }{
459                         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl

```

```

460         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
461             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
462                 \c__stex_path_up_str
463             }
464         }{
465             \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
466         }
467     }
468     }{
469         \str_if_empty:NNTF \l_tmpa_tl {
470             \bool_if:NT \l__stex_path_in_path_bool {
471                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
472             }
473         } {
474             \bool_set_true:N \l__stex_path_in_path_bool
475             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
476         }
477     }
478 }
479 }
480 \seq_gset_eq:NN #1 \l_tmpa_seq
481 \stex_debug:nn{paths}{...returns~\seq_use:Nn #1 /}
482 }
483 }

```

(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**

```

484 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
485     \seq_if_empty:NNTF #1 {
486         \prg_return_false:
487     }{
488         \seq_get_left:NN #1 \l_tmpa_tl
489         \sys_if_platform_windows:TF{
490             \str_if_in:NnTF \l_tmpa_tl {:}{
491                 \prg_return_true:
492             }{
493                 \prg_return_false:
494             }
495         }{
496             \str_if_empty:NNTF \l_tmpa_tl {
497                 \prg_return_true:
498             }{
499                 \prg_return_false:
500             }
501         }
502     }
503 }

```

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

## 27.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

504 \str_new:N\l_stex_kpsewhich_return_str
505 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
506   \catcode'\ =12
507   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
508   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
509   \endgroup
510   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
511   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
512 }

```

(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`

```

513 \sys_if_platform_windows:TF{
514   \begingroup\escapechar=-1\catcode'\ =12
515   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
516   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
517   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
518   }}{
519     \stex_kpsewhich:n{-var-value~PWD}
520   }
521
522 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
523 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
524 \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

```

525 <@@=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

```

526 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`  
`\c_stex_mainfile_str`

```

527 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
528 \stex_path_from_string:Nn \c_stex_mainfile_seq
529   \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`

```

530 \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`

```

531 \cs_new_protected:Nn \stex_filestack_push:n {
532   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
533   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
534     \stex_path_from_string:Nn\g_stex_currentfile_seq{
535       \c_stex_pwd_str/#1
536     }
537   }
538   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
539   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
540   \stex_get_document_uri:
541 }

```

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

`\stex_filestack_pop:`

```

542 \cs_new_protected:Nn \stex_filestack_pop: {
543   \seq_if_empty:NF\g__stex_files_stack{
544     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
545   }
546   \seq_if_empty:NTF\g__stex_files_stack{
547     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
548   }{
549     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
550     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
551   }
552   \stex_get_document_uri:
553 }

```

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

Hooks for the current file:

```

554 \AddToHook{file/before}{
555   \tl_if_empty:NTF\CurrentFilePath{
556     \stex_filestack_push:n{\CurrentFile}
557   }{
558     \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
559   }
560 }
561 \AddToHook{file/after}{
562   \stex_filestack_pop:
563 }

```

## 27.4 MathHub Repositories

564 `<@=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.

```

565 \str_if_empty:NTF\mathhub{
566   \sys_if_platform_windows:TF{
567     \begingroup\escapechar=-1\catcode'\=12

```

```

568 \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
569 \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
570 \exp_args:NNx\str_if_eq:ont\l_stex_kpsewhich_return_str{\c_percent_str MATHHUB\c_percent
571 \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_st
572 }{
573 \stex_kpsewhich:n{-var-value-MATHHUB}
574 }
575 \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
576
577 \str_if_empty:NT \c_stex_mathhub_str {
578 \sys_if_platform_windows:TF{
579 \begingroup\escapechar=-1\catcode'\=12
580 \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
581 \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
582 \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
583 }{
584 \stex_kpsewhich:n{-var-value-HOME}
585 }
586 \ior_open:NnT \g_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
587 \begingroup\escapechar=-1\catcode'\=12
588 \ior_str_get:NN \g_tmpa_ior \l_tmpa_str
589 \sys_if_platform_windows:T{
590 \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
591 }
592 \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
593 \endgroup
594 \ior_close:N \g_tmpa_ior
595 }
596 }
597 \str_if_empty:NTF\c_stex_mathhub_str{
598 \msg_warning:nn{stex}{warning/nomathhub}
599 }{
600 \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
601 \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
602 }
603 }{
604 \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
605 \stex_path_if_absolute:NF \c_stex_mathhub_seq {
606 \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
607 \c_stex_pwd_str/\mathhub
608 }
609 }
610 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
611 \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
612 }

```

(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

```

613 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
614 \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
615 \str_set:Nx \l_tmpa_str { #1 }

```

```

616 \prop_new:c { c_stex_mathhub_#1_manifest_prop }
617 \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
618 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
619 \__stex_mathhub_find_manifest:N \l_tmpa_seq
620 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
621   \msg_error:nnxx{stex}{error/norepository}{#1}{
622     \stex_path_to_string:N \c_stex_mathhub_str
623   }
624   \input{Fatal-Error!}
625 } {
626   \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
627 }
628 }
629 }

```

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

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

```

630 \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:

```

631 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
632   \seq_set_eq:NN\l_tmpa_seq #1
633   \bool_set_true:N\l_tmpa_bool
634   \bool_while_do:Nn \l_tmpa_bool {
635     \seq_if_empty:NTF \l_tmpa_seq {
636       \bool_set_false:N\l_tmpa_bool
637     }{
638       \file_if_exist:nTF{
639         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
640       }{
641         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
642         \bool_set_false:N\l_tmpa_bool
643       }{
644         \file_if_exist:nTF{
645           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
646         }{
647           \seq_put_right:Nn\l_tmpa_seq{META-INF}
648           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
649           \bool_set_false:N\l_tmpa_bool
650         }{
651           \file_if_exist:nTF{
652             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
653           }{
654             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
655             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
656             \bool_set_false:N\l_tmpa_bool
657           }{
658             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
659           }
660         }

```



```

661     }
662   }
663 }
664 \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
665 }

```

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

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

```

666 \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:

```

667 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
668   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
669   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
670   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
671     \str_set:Nn \l_tmpa_str {##1}
672     \exp_args:NNoo \seq_set_split:Nnn
673       \l_tmpb_seq \c_colon_str \l_tmpa_str
674     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
675       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
676         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
677       }
678       \exp_args:No \str_case:nnTF \l_tmpa_tl {
679         {id} {
680           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
681             { id } \l_tmpb_tl
682         }
683         {narration-base} {
684           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
685             { narr } \l_tmpb_tl
686         }
687         {url-base} {
688           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
689             { docurl } \l_tmpb_tl
690         }
691         {source-base} {
692           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
693             { ns } \l_tmpb_tl
694         }
695         {ns} {
696           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
697             { ns } \l_tmpb_tl
698         }
699         {dependencies} {
700           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
701             { deps } \l_tmpb_tl
702         }
703       }{}{}
704     }{}
705   }
706   \ior_close:N \c__stex_mathhub_manifest_ior

```

```

707 \stex_persist:x {
708   \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
709     \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
710   }
711 }
712 }

```

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

`\stex_set_current_repository:n`

```

713 \cs_new_protected:Nn \stex_set_current_repository:n {
714   \stex_require_repository:n { #1 }
715   \prop_set_eq:Nc \l_stex_current_repository_prop {
716     c_stex_mathhub_#1_manifest_prop
717   }
718 }

```

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

`\stex_require_repository:n`

```

719 \cs_new_protected:Nn \stex_require_repository:n {
720   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
721     \stex_debug:nn{mathhub}{Opening~archive:~#1}
722     \__stex_mathhub_do_manifest:n { #1 }
723   }
724 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

725 %\prop_new:N \l_stex_current_repository_prop
726 \bool_if:NF \c_stex_persist_mode_bool {
727   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
728   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
729     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
730   } {
731     \__stex_mathhub_parse_manifest:n { main }
732     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
733     \l_tmpa_str
734     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
735     \c_stex_mathhub_main_manifest_prop
736     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
737     \stex_debug:nn{mathhub}{Current~repository:~
738       \prop_item:Nn \l_stex_current_repository_prop {id}
739     }
740   }
741 }

```

(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.

```

742 \cs_new_protected:Nn \stex_in_repository:nn {
743   \str_set:Nx \l_tmpa_str { #1 }
744   \cs_set:Npn \l_tmpa_cs ##1 { #2 }

```

```

745 \str_if_empty:NTF \l_tmpa_str {
746   \prop_if_exist:NTF \l_stex_current_repository_prop {
747     \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
748     \exp_args:Ne \l_tmpa_cs{
749       \prop_item:Nn \l_stex_current_repository_prop { id }
750     }
751   }{
752     \l_tmpa_cs{}
753   }
754 }{
755   \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
756   \stex_require_repository:n \l_tmpa_str
757   \str_set:Nx \l_tmpa_str { #1 }
758   \exp_args:Nne \use:nn {
759     \stex_set_current_repository:n \l_tmpa_str
760     \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
761   }{
762     \stex_debug:nn{mathhub}{switching~back~to:~
763     \prop_if_exist:NTF \l_stex_current_repository_prop {
764       \prop_item:Nn \l_stex_current_repository_prop { id }::~
765     \meaning\l_stex_current_repository_prop
766     }{
767       no~repository
768     }
769   }
770   \prop_if_exist:NTF \l_stex_current_repository_prop {
771     \stex_set_current_repository:n {
772       \prop_item:Nn \l_stex_current_repository_prop { id }
773     }
774   }{
775     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
776   }
777 }
778 }
779 }

```

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

## 27.5 Using Content in Archives

`\mhpath`

```

780 \def \mhpath #1 #2 {
781   \exp_args:Ne \tl_if_empty:NTF{#1}{
782     \c_stex_mathhub_str /
783     \prop_item:Nn \l_stex_current_repository_prop { id }
784     / source / #2
785   }{
786     \c_stex_mathhub_str / #1 / source / #2
787   }
788 }

```

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

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

```

789 \newif \ifinputref \inputreffalse
790
791 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
792   \stex_in_repository:nn {#1} {
793     \ifinputref
794       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
795     \else
796       \inputreftrue
797       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
798     \inputreffalse
799   \fi
800 }
801 }
802 \NewDocumentCommand \mhinput { 0{} m}{
803   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
804 }
805
806 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
807   \stex_in_repository:nn {#1} {
808     \stex_html_backend:TF {
809       \str_clear:N \l_tmpa_str
810       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
811         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
812       }
813
814       \tl_if_empty:nTF{ ##1 }{
815         \IfFileExists{#2}{
816           \stex_annotate_invisible:nnn{inputref}{
817             \l_tmpa_str / #2
818           }{}
819         }{
820           \input{#2}
821         }
822       }{
823         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
824           \stex_annotate_invisible:nnn{inputref}{
825             \l_tmpa_str / #2
826           }{}
827         }{
828           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
829         }
830       }
831
832     }{
833       \begingroup
834       \inputreftrue
835       \tl_if_empty:nTF{ ##1 }{
836         \input{#2}
837       }{
838         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
839       }
840       \endgroup
841     }

```

```

842 }
843 }
844 \NewDocumentCommand \inputref { 0{ } m }{
845   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
846 }

```

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

### `\addmhbibresource`

```

847 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
848   \stex_in_repository:nn {#1} {
849     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
850   }
851 }
852 \newcommand\addmhbibresource[2][]{
853   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
854 }

```

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

### `\libinput`

```

855 \cs_new_protected:Npn \libinput #1 {
856   \prop_if_exist:NF \l_stex_current_repository_prop {
857     \msg_error:nnn{stex}{error/notinarchive}\libinput
858   }
859   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
860     \msg_error:nnn{stex}{error/notinarchive}\libinput
861   }
862   \seq_clear:N \l__stex_mathhub_libinput_files_seq
863   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
864   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
865
866   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
867     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
868     \IfFileExists{ \l_tmpa_str }{
869       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
870     }{}
871     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
872     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
873   }
874
875   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
876   \IfFileExists{ \l_tmpa_str }{
877     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
878   }{}
879
880   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
881     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
882   }{
883     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
884       \input{ ##1 }
885     }
886   }
887 }

```

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

## `\libusepackage`

```

888 \NewDocumentCommand \libusepackage {0{} m} {
889   \prop_if_exist:NF \l_stex_current_repository_prop {
890     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
891   }
892   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
893     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
894   }
895   \seq_clear:N \l__stex_mathhub_libinput_files_seq
896   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
897   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
898
899   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
900     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
901     \IfFileExists{ \l_tmpa_str.sty }{
902       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
903     }{}
904     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
905     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
906   }
907
908   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
909   \IfFileExists{ \l_tmpa_str.sty }{
910     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
911   }{}
912
913   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
914     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
915   }{
916     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
917       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
918         \usepackage[#1]{ ##1 }
919       }
920     }{
921       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
922     }
923   }
924 }

```

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

## `\mhgraphics`

## `\cmhgraphics`

```

925
926 \AddToHook{begindocument}{
927   \ltx@ifpackageloaded{graphicx}{
928     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
929     \providecommand\mhgraphics[2][]{%
930       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
931       \includegraphics[#1]{\mhp@h\Gin@mhrepos{#2}}}
932     \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
933   }{}

```

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

`\lstinputmhlisting`  
`\clstinputmhlisting`

```
934 \ltx@ifpackageloaded{listings}{  
935   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}  
936   \newcommand\lstinputmhlisting[2][]{%  
937     \def\lst@mhrepos{}\setkeys{lst}{#1}%  
938     \lstinputlisting[#1]{\mhpath\lst@mhrepos{#2}}}  
939   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}  
940 }{}  
941 }  
942  
943 \end{package}
```

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

## Chapter 28

# STEX -References Implementation

```
944 <*package>
945
946 %%%%%%%%% stex-references.dtx %%%%%%%%%
947
948 <@@=stex_refs>
949
950 Warnings and error messages
951 \msg_new:nnn{stex}{error/extrefmissing}{
952   Missing~in~or~cite~value~for~\detokenize{\extref}!
953 }
954 \msg_new:nnn{stex}{warning/smsmissing}{
955   .sref~file~#1~doesn't~exist!
956 }
957 \msg_new:nnn{stex}{warning/smslabelmissing}{
958   No~label~#2~in~.sref~file~#1!
959 }
```

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

```
958 \iow_new:N \c__stex_refs_refs_iow
959 \AtBeginDocument{
960   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
961 }
962 \AtEndDocument{
963   \iow_close:N \c__stex_refs_refs_iow
964 }
```

### 28.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
965 \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:`

```
966 \cs_new_protected:Nn \stex_get_document_uri: {
```



```

967 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
968 \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
969 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
970 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
971 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
972
973 \str_clear:N \l_tmpa_str
974 \prop_if_exist:NT \l_stex_current_repository_prop {
975   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
976     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
977   }
978 }
979
980 \str_if_empty:NTF \l_tmpa_str {
981   \str_set:Nx \l_stex_current_docns_str {
982     file:/\stex_path_to_string:N \l_tmpa_seq
983   }
984 }{
985   \bool_set_true:N \l_tmpa_bool
986   \bool_while_do:Nn \l_tmpa_bool {
987     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
988     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
989       {source} { \bool_set_false:N \l_tmpa_bool }
990     }{}{
991       \seq_if_empty:NT \l_tmpa_seq {
992         \bool_set_false:N \l_tmpa_bool
993       }
994     }
995   }
996
997   \seq_if_empty:NTF \l_tmpa_seq {
998     \str_gset_eq:NN \l_stex_current_docns_str \l_tmpa_str
999   }{
1000     \str_gset:Nx \l_stex_current_docns_str {
1001       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1002     }
1003   }
1004 }
1005 %\stex_get_document_url:
1006 }

```

(End definition for `\stex_get_document_uri`:. This function is documented on page 81.)

`\l_stex_current_docurl_str`

```

1007 \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:`

```

1008 \cs_new_protected:Nn \stex_get_document_url: {
1009   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1010   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1011   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1012   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1013   \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

1014 \str_clear:N \l_tmpa_str
1015 \prop_if_exist:NT \l_stex_current_repository_prop {
1016   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
1017     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
1018       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
1019     }
1020   }
1021 }
1022 }
1023
1024 \str_if_empty:NTF \l_tmpa_str {
1025   \str_set:Nx \l_stex_current_docurl_str {
1026     file:/\stex_path_to_string:N \l_tmpa_seq
1027   }
1028 }{
1029   \bool_set_true:N \l_tmpa_bool
1030   \bool_while_do:Nn \l_tmpa_bool {
1031     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1032     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1033       {source} { \bool_set_false:N \l_tmpa_bool }
1034     }{}{
1035       \seq_if_empty:NT \l_tmpa_seq {
1036         \bool_set_false:N \l_tmpa_bool
1037       }
1038     }
1039   }
1040
1041   \seq_if_empty:NTF \l_tmpa_seq {
1042     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1043   }{
1044     \str_set:Nx \l_stex_current_docurl_str {
1045       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1046     }
1047   }
1048 }
1049 }

```

(End definition for `\stex_get_document_url`.. This function is documented on page 81.)

## 28.2 Setting Reference Targets

```

1050 \str_const:Nn \c__stex_refs_url_str{URL}
1051 \str_const:Nn \c__stex_refs_ref_str{REF}
1052 \str_new:N \l__stex_refs_curr_label_str
1053 % @currentlabel -> number
1054 % @currentlabelname -> title
1055 % @currentHref -> name.number <- id of some kind
1056 % @currentcounter <- name/id
1057 % \#autorefname <- "Section"
1058 % \theH# -> \arabic{section}
1059 % \the# -> number
1060 % \hyper@makecurrent{#}
1061 \int_new:N \l__stex_refs_unnamed_counter_int

```

Restoring references from .sref-files

\STEXInternalSrefRestoreTarget

1062 \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

```

1063 \seq_new:N \g_stex_ref_files_seq
1064
1065 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1066   %\stex_get_document_uri:
1067   \str_clear:N \l__stex_refs_curr_label_str
1068   \str_set:Nx \l_tmpa_str { #1 }
1069   \str_if_empty:NT \l_tmpa_str {
1070     \int_gincr:N \l__stex_refs_unnamed_counter_int
1071     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1072   }
1073   \str_set:Nx \l__stex_refs_curr_label_str {
1074     \l_stex_current_docns_str?\l_tmpa_str
1075   }
1076
1077   \exp_args:Noo \STEXInternalAuxAddDocRef\l_stex_current_docns_str\l_tmpa_str
1078
1079   %\seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1080   %   \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1081   %}
1082   %\seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1083   %   \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1084   %}
1085
1086
1087   \stex_if_smsmode:TF {
1088     %\stex_get_document_url:
1089     %\str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1090     %\str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1091   }{
1092     \iow_now:Nx \c__stex_refs_refs_iow {
1093       \STEXInternalSrefRestoreTarget
1094       {\l_stex_current_docns_str}
1095       {\l_tmpa_str}
1096       {\@currentcounter}
1097       {\@currentlabel}
1098       {\tl_if_exist:NT\@currentlabelname{\exp_args:No\unexpanded\@currentlabelname}}
1099     }
1100     %\iow_now:Nx \c__stex_refs_refs_iow {
1101     %   {\l_stex_current_docns_str?\l_tmpa_str}~=={\use:c{\@currentcounter autorefname}~\@currentlabel}
1102     %   \stex_debug:nn{sref}{New~label~\l__stex_refs_curr_label_str~at~\use:c{\use:c{\@currentcounter}~\@currentlabel}}
1103     %   \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1104     %   \immediate\write\auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_str}{\l__stex_refs_curr_label_str}}
1105     %   \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1106     %}
1107   }
1108   \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.

```

1109 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1110   \exp_args:NNx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1111     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1112       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1113     }
1114   }{
1115     \exp_args:NNx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1116     %\seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1117       \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1118     %}
1119     \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1120   }
1121
1122   %\str_set:Nn \l_tmpa_str {#1?#2}
1123   %\str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1124   %\seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1125     % \seq_new:c {g__stex_refs_labels_#2_seq}
1126     %}
1127   %\seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1128     % \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1129     %}
1130 }

```

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

```

1131 \AtEndDocument{
1132   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1133 }

```

`\stex_ref_new_sym_target:n`

```

1134 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1135
1136   % \stex_if_smsmode:TF {
1137   %   \str_if_exist:cF{sref_sym_#1_type}{
1138   %     \stex_get_document_url:
1139   %     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
1140   %     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1141   %   }
1142   % }{
1143   %   \str_if_empty:NF \l__stex_refs_curr_label_str {
1144   %     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1145   %     \immediate\write\@auxout{
1146   %       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label
1147   %       \l__stex_refs_curr_label_str
1148   %     }
1149   %   }
1150   % }
1151 % }
1152 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 81.)

## 28.3 Using References

`\sref` Optional arguments:

```

1153
1154 \keys_define:nn { stex / sref / 1 } {
1155   archive .str_set_x:N = \l__stex_refs_repo_str,
1156   file     .str_set_x:N = \l__stex_refs_file_str,
1157   % TODO get rid of this
1158   fallback .code:n = {},
1159   pre      .code:n = {},
1160   post     .code:n = {}
1161 }
1162 \cs_new_protected:Nn \__stex_refs_args_i:n {
1163   \str_clear:N \l__stex_refs_repo_str
1164   \str_clear:N \l__stex_refs_file_str
1165   \keys_set:nn { stex / sref / 1 } { #1 }
1166 }
1167 \keys_define:nn { stex / sref / 2 } {
1168   in .str_set_x:N = \l__stex_refs_in_str,
1169   archive .str_set_x:N = \l__stex_refs_repob_str,
1170   title .tl_set:N = \l__stex_refs_title_tl
1171 }
1172 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1173   \str_clear:N \l__stex_refs_in_str
1174   \tl_clear:N \l__stex_refs_title_tl
1175   \str_clear:N \l__stex_refs_repob_str
1176   \keys_set:nn { stex / sref / 2 } { #1 }
1177 }

```

The actual macro:

```

1178 \NewDocumentCommand \sref { 0{} m 0{} }{
1179   \__stex_refs_args_i:n{#1}
1180   \__stex_refs_args_ii:n{#3}
1181   \str_clear:N \l__stex_refs_uri_str
1182   \__stex_refs_find_uri:n{#2}
1183   \__stex_refs_do_sref:n{#2}
1184 }
1185 \NewDocumentCommand \extref { 0{} m m }{
1186   \__stex_refs_args_i:n{#1}
1187   \__stex_refs_args_ii:n{#3}
1188   \str_if_empty:NT \l__stex_refs_in_str {
1189     \msg_error:nn{stex}{error/extrefmissing}
1190   }
1191   \str_clear:N \l__stex_refs_uri_str
1192   \__stex_refs_find_uri:n{#2}
1193   \__stex_refs_do_sref_in:n{#2}
1194 }
1195
1196 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1197   \stex_debug:nn{sref}{File:~\l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1198   \str_if_empty:NTF \l__stex_refs_file_str {
1199     \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1200     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str_seq}{
1201       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str_seq}{

```

```

1202         \str_if_eq:nnT{#1}{##1}{
1203             \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1204             \stex_debug:nn{sref}{Found.}
1205             \seq_map_break:
1206         }
1207     }
1208 }
1209 \str_if_empty:NT \l__stex_refs_uri_str {
1210     \stex_debug:nn{sref}{Checking~other~files}
1211     \seq_map_inline:Nn \g_stex_ref_files_seq {
1212         \stex_debug:nn{sref}{##1...}
1213         \seq_map_inline:cn{g_stex_ref_##1_seq}{
1214             \str_if_eq:nnT{#1}{####1}{
1215                 \stex_debug:nn{sref}{Found~##1}
1216                 \str_set:Nn \l__stex_refs_uri_str {##1}
1217                 \seq_map_break:n{\seq_map_break:}
1218             }
1219         }
1220     }
1221 }
1222 }{
1223     \str_if_empty:NTF \l__stex_refs_repo_str {
1224         \prop_if_exist:NTF \l_stex_current_repository_prop {
1225             \stex_debug:nn{sref}{in~archive~\prop_item:Nn \l_stex_current_repository_prop { id }
1226             \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1227             \stex_debug:nn{sref}{namespace:~\l__stex_refs_uri_str}
1228             \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1229             \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1230             \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1231             \stex_debug:nn{sref}{Return:~\l__stex_refs_uri_str}
1232         }{
1233             \stex_debug:nn{sref}{Not~in~archive}
1234             \stex_path_from_string:Nn \l_tmpb_seq {
1235                 \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1236             }
1237             \str_set:Nx \l__stex_refs_uri_str {file:~\stex_path_to_string:N \l_tmpb_seq}
1238         }
1239     }{
1240         \stex_require_repository:n \l__stex_refs_repo_str
1241         \prop_get:cnN { c_stex_mathhub \l__stex_refs_repo_str _manifest_prop } { ns } \l__stex
1242         \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1243         \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1244         \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1245     }
1246 }
1247 }
1248
1249 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1250     \cs_if_exist:cTF{autoref}{
1251         \exp_args:Nx\autoref{sref_#1}
1252     }{
1253         \exp_args:Nx\ref{sref_#1}
1254     }
1255 }

```

```

1256
1257 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1258   \str_if_empty:NTF \l__stex_refs_uri_str {
1259     \str_if_empty:NTF \l__stex_refs_in_str {
1260       \stex_debug:nn{sref}{autoref~on~#1}
1261       \__stex_refs_do_autoref:n{#1}
1262     }{
1263       \stex_debug:nn{sref}{srefin~on~#1}
1264       \__stex_refs_do_sref_in:n{#1}
1265     }
1266   }{
1267     \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1268       \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref\_l__stex_refs_uri_str_seq}{\detokenize{#1}}{
1269         \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref~on~\l__stex_refs_uri_str?#1}
1270         \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1271       }{
1272         \str_if_empty:NTF \l__stex_refs_in_str {
1273           \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
1274           \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1275         }{
1276           \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l__stex_refs_uri_str?#1}
1277           \__stex_refs_do_sref_in:n{#1}
1278         }
1279       }
1280     }{
1281       \str_if_empty:NTF \l__stex_refs_in_str {
1282         \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
1283         \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1284       }{
1285         \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l__stex_refs_uri_str?#1}
1286         \__stex_refs_do_sref_in:n{#1}
1287       }
1288     }
1289   }
1290 }
1291
1292 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1293   \str_if_empty:NTF \l__stex_refs_uri_str {
1294     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1295       \tl_set:Nn \l__stex_refs_return_tl {
1296         \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1297         \tl_if_empty:NTF\l__stex_refs_title_tl{
1298           ???
1299         }\l__stex_refs_title_tl
1300       }
1301     }
1302   }{
1303     \stex_debug:nn{sref}{\l__stex_refs_uri_str{~ == ~ #1 ~ ?}}
1304     \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1305       \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}}
1306     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1307       \stex_debug:nn{sref}{success!}
1308       \tl_set:Nn \l__stex_refs_return_tl {
1309         \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~

```

```

1310         \tl_if_empty:nTF\l__stex_refs_title_tl{
1311             ???
1312         }\l__stex_refs_title_tl
1313     }
1314     \endinput
1315 }
1316 }
1317 }
1318 }
1319
1320 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1321     \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1322     \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1323     %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1324     \begingroup\catcode13=9\relax\catcode10=9\relax
1325     \str_if_empty:NTF \l__stex_refs_repob_str {
1326         \prop_if_exist:NTF \l_stex_current_repository_prop {
1327             \str_set:Nx \l_tmpa_str {
1328                 \c_stex_mathhub_str /
1329                 \prop_item:Nn \l_stex_current_repository_prop { id }
1330                 / source / \l__stex_refs_in_str .sref
1331             }
1332         }{
1333             \str_set:Nx \l_tmpa_str {
1334                 \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str . sref
1335             }
1336         }
1337     }{
1338         \str_set:Nx \l_tmpa_str {
1339             \c_stex_mathhub_str / \l__stex_refs_repob_str
1340             / source / \l__stex_refs_in_str . sref
1341         }
1342     }
1343     \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1344     \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1345     \stex_debug:nn{sref}{File: \l_tmpa_str}
1346     \exp_args:No \IfFileExists \l_tmpa_str {
1347         \tl_clear:N \l__stex_refs_return_tl
1348         \str_set:Nn \l__stex_refs_id_str {#1}
1349         \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1350         \use:c{@ @ input}{\l_tmpa_str}
1351         \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1352             \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1353             \__stex_refs_do_autoref:n{
1354                 \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1355             }
1356         }{
1357             \l__stex_refs_return_tl
1358         }
1359     }{
1360         \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l_tmpa_str
1361         \__stex_refs_do_autoref:n{
1362             \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1363         }

```



```

1364     }
1365 \endgroup
1366 }
1367
1368 % \__stex_refs_args:n { #1 }
1369 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1370 %   \str_set:Nx \l_tmpa_str { #2 }
1371 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1372 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1373 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1374 %       \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str_seq} \l_tmpa_str {
1375 %         \str_clear:N \l_tmpa_str
1376 %       }
1377 %     }{
1378 %       \str_clear:N \l_tmpa_str
1379 %     }
1380 %   }{
1381 %     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1382 %     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1383 %     \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1384 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1385 %       \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1386 %       \str_clear:N \l_tmpa_str
1387 %       \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1388 %         \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1389 %           \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1390 %         }{
1391 %           \seq_map_break:n {
1392 %             \str_set:Nn \l_tmpa_str { ##1 }
1393 %           }
1394 %         }
1395 %       }
1396 %     }{
1397 %       \str_clear:N \l_tmpa_str
1398 %     }
1399 %   }
1400 % \str_if_empty:NTF \l_tmpa_str {
1401 %   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_li
1402 % }{
1403 %   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1404 %     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1405 %       \cs_if_exist:cTF{autoref}{
1406 %         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1407 %       }{
1408 %         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1409 %       }
1410 %     }{
1411 %       \ltx@ifpackageloaded{hyperref}{
1412 %         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1413 %       }{
1414 %         \l__stex_refs_linktext_tl
1415 %       }
1416 %     }
1417 %   }{

```

```

1418 % \ltx@ifpackageloaded{hyperref}{
1419 % \href{\use:c{sref_url_1\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1420 % }{
1421 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_fallback_tl
1422 % }
1423 % }
1424 % }
1425 % }{
1426 % TODO
1427 % }
1428 %}

```

(End definition for `\sref`. This function is documented on page 82.)

### `\srefsym`

```

1429 \NewDocumentCommand \srefsym { 0{} m}{
1430 \stex_get_symbol:n { #2 }
1431 \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1432 }
1433
1434 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1435
1436 % \str_if_exist:cTF {sref_sym_#2_label_str }{
1437 % \sref[#1]{\use:c{sref_sym_#2_label_str}}
1438 % }{
1439 % \__stex_refs_args:n { #1 }
1440 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1441 % \tl_if_exist:cTF{sref_sym_#2_type}{
1442 % % doc uri in \l_tmpb_str
1443 % \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2_type}}
1444 % \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1445 % % reference
1446 % \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1447 % \cs_if_exist:cTF{autoref}{
1448 % \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1449 % }{
1450 % \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1451 % }
1452 % }{
1453 % \ltx@ifpackageloaded{hyperref}{
1454 % \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1455 % }{
1456 % \l__stex_refs_linktext_tl
1457 % }
1458 % }
1459 % }{
1460 % % URL
1461 % \ltx@ifpackageloaded{hyperref}{
1462 % \href{\use:c{sref_sym_url_#2_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1463 % }{
1464 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_fallback_tl
1465 % }
1466 % }
1467 % }{

```

```

1468 %          \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1469 %          }
1470 %      }{
1471 %          % TODO
1472 %      }
1473 %  }
1474 }

```

*(End definition for `\srefsym`. This function is documented on page 82.)*

**`\srefsymuri`**

```

1475 \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1476   #2%\__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1477 }

```

*(End definition for `\srefsymuri`. This function is documented on page 82.)*

```

1478 </package>

```

## Chapter 29

# STEX -Modules Implementation

```
1479 <*package>
1480
1481 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1482
1483 <@@=stex_modules>
1484
1485 Warnings and error messages
1486 \msg_new:nnn{stex}{error/unknownmodule}{
1487   No~module~#1~found
1488 }
1489 \msg_new:nnn{stex}{error/syntax}{
1490   Syntax~error:~#1
1491 }
1492 \msg_new:nnn{stex}{error/siglanguage}{
1493   Module~#1~declares~signature~#2,~but~does~not~
1494   declare~its~language
1495 }
1496 \msg_new:nnn{stex}{warning/deprecated}{
1497   #1~is~deprecated;~please~use~#2~instead!
1498 }
1499 \msg_new:nnn{stex}{error/conflictingmodules}{
1500   Conflicting~imports~for~module~#1
1501 }
```

```
\l_stex_current_module_str The current module:
1501 \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
1502 \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
1503 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1504   \str_if_empty:NTF \l_stex_current_module_str
1505   \prg_return_false: \prg_return_true:
1506 }

```

(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
1507 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1508   \prop_if_exist:cTF { c_stex_module_#1_prop }
1509   \prg_return_true: \prg_return_false:
1510 }

```

(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
1511 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1512   \stex_add_to_current_module:n { #1 }
1513   \stex_do_up_to_module:n { #1 }
1514 }}
1515 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1516
1517 \cs_new_protected:Nn \stex_add_to_current_module:n {
1518   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1519 }
1520 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1521 \cs_new_protected:Npn \STEXexport {
1522   \ExplSyntaxOn
1523   \__stex_modules_export:n
1524 }
1525 \cs_new_protected:Nn \__stex_modules_export:n {
1526   \ignorespacesandpars#1\ExplSyntaxOff
1527   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1528   \stex_smsmode_do:
1529 }
1530 \let \stex_module_export_helper:n \use:n
1531 \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
1532 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1533   \str_set:Nx \l_tmpa_str { #1 }
1534   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1535 }

```

(End definition for \stex\_add\_constant\_to\_current\_module:n. This function is documented on page 84.)

```

\stex_add_import_to_current_module:n
1536 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1537   \str_set:Nx \l_tmpa_str { #1 }
1538   \exp_args:Nno

```

```

1539 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1540 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1541 }
1542 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 84.)

`\stex_collect_imports:n`

```

1543 \cs_new_protected:Nn \stex_collect_imports:n {
1544 \seq_clear:N \l_stex_collect_imports_seq
1545 \__stex_modules_collect_imports:n {#1}
1546 }
1547 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1548 \seq_map_inline:cn {c_stex_module_#1_imports} {
1549 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1550 \__stex_modules_collect_imports:n { ##1 }
1551 }
1552 }
1553 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1554 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1555 }
1556 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 84.)

`\stex_do_up_to_module:n`

```

1557 \int_new:N \l__stex_modules_group_depth_int
1558 \cs_new_protected:Nn \stex_do_up_to_module:n {
1559 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1560 #1
1561 }{
1562 #1
1563 \expandafter \tl_gset:Nn
1564 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1565 \expandafter\expandafter\expandafter\endcsname
1566 \expandafter\expandafter\expandafter { \csname
1567 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1568 \aftergroup\__stex_modules_aftergroup_do:
1569 }
1570 }
1571 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1572 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1573 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1574 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1575 }}}
1576 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1577 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1578 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1579 }{
1580 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1581 \aftergroup\__stex_modules_aftergroup_do:
1582 }
1583 }
1584 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1585 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1586 }

(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).

1587

(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.

```
1588 \str_new:N \l_stex_module_ns_str
1589 \str_new:N \l_stex_module_subpath_str
1590 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1591   \seq_set_eq:NN \l_tmpa_seq #2
1592   % split off file extension
1593   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1594   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1595   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1596   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1597
1598   \bool_set_true:N \l_tmpa_bool
1599   \bool_while_do:Nn \l_tmpa_bool {
1600     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1601     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1602       {source} { \bool_set_false:N \l_tmpa_bool }
1603     }{}{
1604       \seq_if_empty:NT \l_tmpa_seq {
1605         \bool_set_false:N \l_tmpa_bool
1606       }
1607     }
1608   }
1609
1610   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1611   % \l_tmpa_seq <- sub-path relative to archive
1612   \str_if_empty:NTF \l_stex_module_subpath_str {
1613     \str_set:Nx \l_stex_module_ns_str {#1}
1614   }{
1615     \str_set:Nx \l_stex_module_ns_str {
1616       #1/\l_stex_module_subpath_str
1617     }
1618   }
1619 }
1620
1621 \cs_new_protected:Nn \stex_modules_current_namespace: {
1622   \str_clear:N \l_stex_module_subpath_str
1623   \prop_if_exist:NTF \l_stex_current_repository_prop {
1624     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1625     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1626   }{
1627     % split off file extension
1628     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1629     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

1630 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1631 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1632 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1633 \str_set:Nx \l_stex_module_ns_str {
1634   file:/\stex_path_to_string:N \l_tmpa_seq
1635 }
1636 }
1637 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 85.)

## 29.1 The smodule environment

smodule arguments:

```

1638 \keys_define:nn { stex / module } {
1639   title      .tl_set:N      = \smodulename ,
1640   type       .str_set_x:N   = \smodulename ,
1641   id         .str_set_x:N   = \smoduleid ,
1642   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1643   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1644   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1645   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1646   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1647   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1648   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1649   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1650 }
1651
1652 \cs_new_protected:Nn \__stex_modules_args:n {
1653   \str_clear:N \smodulename
1654   \str_clear:N \smodulename
1655   \str_clear:N \smoduleid
1656   \str_clear:N \l_stex_module_ns_str
1657   \str_clear:N \l_stex_module_deprecate_str
1658   \str_clear:N \l_stex_module_lang_str
1659   \str_clear:N \l_stex_module_sig_str
1660   \str_clear:N \l_stex_module_creators_str
1661   \str_clear:N \l_stex_module_contributors_str
1662   \str_clear:N \l_stex_module_meta_str
1663   \str_clear:N \l_stex_module_srccite_str
1664   \keys_set:nn { stex / module } { #1 }
1665 }
1666
1667 % module parameters here? In the body?
1668

```

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

```

1669 \cs_new_protected:Nn \stex_module_setup:nn {
1670   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1671   \str_set:Nx \l_stex_module_name_str { #2 }
1672   \__stex_modules_args:n { #1 }

```



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

Are we in a nested module?

```

1673 \stex_if_in_module:TF {
1674   % Nested module
1675   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1676   { ns } \l_stex_module_ns_str
1677   \str_set:Nx \l_stex_module_name_str {
1678     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1679     { name } / \l_stex_module_name_str
1680   }
1681   \str_if_empty:NT \l_stex_module_lang_str {
1682     \str_set:Nx \l_stex_module_lang_str {
1683       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1684       { lang }
1685     }
1686   }
1687 }{
1688   % not nested:
1689   \str_if_empty:NT \l_stex_module_ns_str {
1690     \stex_modules_current_namespace:
1691     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1692     / {\l_stex_module_ns_str}
1693     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1694     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1695       \str_set:Nx \l_stex_module_ns_str {
1696         \stex_path_to_string:N \l_tmpa_seq
1697       }
1698     }
1699   }
1700 }

```

Next, we determine the language of the module:

```

1701 \str_if_empty:NT \l_stex_module_lang_str {
1702   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1703   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1704   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1705   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1706     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1707       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1708     }
1709   }
1710   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1711   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1712     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1713     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1714       inferred~from~file~name}
1715   }
1716 }
1717
1718 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1719   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1720 }}

```

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

```

1721 \str_if_empty:NTF \l_stex_module_sig_str {
1722   \exp_args:Nnx \prop_gset_from_keyval:cn {
1723     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1724   } {
1725     name      = \l_stex_module_name_str ,
1726     ns        = \l_stex_module_ns_str ,
1727     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1728     lang      = \l_stex_module_lang_str ,
1729     sig       = \l_stex_module_sig_str ,
1730     deprecate = \l_stex_module_deprecate_str ,
1731     meta      = \l_stex_module_meta_str
1732   }
1733   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1734   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1735   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1736   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1737   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1738 \str_if_empty:NT \l_stex_module_meta_str {
1739   \str_set:Nx \l_stex_module_meta_str {
1740     \c_stex_metatheory_ns_str ? Metatheory
1741   }
1742 }
1743 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1744   \bool_set_true:N \l_stex_in_meta_bool
1745   \exp_args:Nx \stex_add_to_current_module:n {
1746     \bool_set_true:N \l_stex_in_meta_bool
1747     \stex_activate_module:n {\l_stex_module_meta_str}
1748     \bool_set_false:N \l_stex_in_meta_bool
1749   }
1750   \stex_activate_module:n {\l_stex_module_meta_str}
1751   \bool_set_false:N \l_stex_in_meta_bool
1752 }
1753 }{
1754   \str_if_empty:NT \l_stex_module_lang_str {
1755     \msg_error:nnxx{stex}{error/siglanguage}{
1756       \l_stex_module_ns_str?\l_stex_module_name_str
1757     }{\l_stex_module_sig_str}
1758   }
1759   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1760   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1761     \stex_debug:nn{modules}{(already exists)}
1762   }{
1763     \stex_debug:nn{modules}{(needs loading)}
1764     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1765     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1766     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1767     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1768     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1769     \str_set:Nx \l_tmpa_str {
1770       \stex_path_to_string:N \l_tmpa_seq /

```

```

1771     \l_tmpa_str . \l_stex_module_sig_str .tex
1772   }
1773   \IfFileExists \l_tmpa_str {
1774     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1775       \str_clear:N \l_stex_current_module_str
1776       \seq_clear:N \l_stex_all_modules_seq
1777       \stex_debug:nn{modules}{Loading~signature}
1778     }
1779   }{
1780     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1781   }
1782 }
1783 \stex_if_smsmode:F {
1784   \stex_activate_module:n {
1785     \l_stex_module_ns_str ? \l_stex_module_name_str
1786   }
1787 }
1788 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1789 }
1790 \str_if_empty:NF \l_stex_module_deprecate_str {
1791   \msg_warning:nnxx{stex}{warning/deprecated}{
1792     Module~\l_stex_current_module_str
1793   }{
1794     \l_stex_module_deprecate_str
1795   }
1796 }
1797 \seq_put_right:Nx \l_stex_all_modules_seq {
1798   \l_stex_module_ns_str ? \l_stex_module_name_str
1799 }
1800 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1801 }

```

(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}
1802 \cs_new_protected:Nn \__stex_modules_begin_module: {
1803   \stex_reactivate_macro:N \STEXexport
1804   \stex_reactivate_macro:N \importmodule
1805   \stex_reactivate_macro:N \symdecl
1806   \stex_reactivate_macro:N \notation
1807   \stex_reactivate_macro:N \symdef
1808
1809   \stex_debug:nn{modules}{
1810     New~module:\\
1811     Namespace:~\l_stex_module_ns_str\\
1812     Name:~\l_stex_module_name_str\\
1813     Language:~\l_stex_module_lang_str\\
1814     Signature:~\l_stex_module_sig_str\\
1815     Metatheory:~\l_stex_module_meta_str\\
1816     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1817   }
1818

```

```

1819 \stex_if_do_html:T{
1820   \begin{stex_annotate_env} {theory} {
1821     \l_stex_module_ns_str ? \l_stex_module_name_str
1822   }
1823
1824   \stex_annotate_invisible:nnn{header}{} {
1825     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1826     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1827     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1828       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1829     }
1830     \str_if_empty:NF \smoduletype {
1831       \stex_annotate:nnn{type}{\smoduletype}{}
1832     }
1833   }
1834 }
1835 % TODO: Inherit metatheory for nested modules?
1836 }
1837 \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}`

```

1838 \cs_new_protected:Nn \_stex_modules_end_module: {
1839   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1840   \stex_reset_up_to_module:n \l_stex_current_module_str
1841   \stex_if_smsmode:T {
1842     \stex_persist:x {
1843       \prop_set_from_keyval:cn{c_stex_module\_l_stex_current_module_str _prop}{
1844         \exp_after:wN \prop_to_keyval:N \csname c_stex_module\_l_stex_current_module_str _pr
1845       }
1846       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _constants}{
1847         \seq_use:cn{c_stex_module\_l_stex_current_module_str _constants},
1848       }
1849       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _imports}{
1850         \seq_use:cn{c_stex_module\_l_stex_current_module_str _imports},
1851       }
1852       \tl_set:cn {c_stex_module\_l_stex_current_module_str _code}
1853     }
1854     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module\_l_stex_current_module
1855     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1856   }
1857 }

```

(End definition for `\_stex_modules_end_module:.`)

The core environment

```

1858 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1859 \NewDocumentEnvironment { smodule } { 0 } { m } {
1860   \stex_module_setup:nn{#1}{#2}
1861   %\par
1862   \stex_if_smsmode:F{
1863     \tl_if_empty:NF \smoduletitle {
1864       \exp_args:No \stex_document_title:n \smoduletitle
1865     }

```

```

1866 \tl_clear:N \l_tmpa_tl
1867 \clist_map_inline:Nn \smodulotype {
1868   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1869     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1870   }
1871 }
1872 \tl_if_empty:NTF \l_tmpa_tl {
1873   \__stex_modules_smodule_start:
1874 }{
1875   \l_tmpa_tl
1876 }
1877 }
1878 \__stex_modules_begin_module:
1879 \str_if_empty:NF \smoduleid {
1880   \stex_ref_new_doc_target:n \smoduleid
1881 }
1882 \stex_smsmode_do:
1883 } {
1884   \__stex_modules_end_module:
1885   \stex_if_smsmode:F {
1886     \end{stex_annotate_env}
1887     \clist_set:Nn \l_tmpa_clist \smodulotype
1888     \tl_clear:N \l_tmpa_tl
1889     \clist_map_inline:Nn \l_tmpa_clist {
1890       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1891         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1892       }
1893     }
1894     \tl_if_empty:NTF \l_tmpa_tl {
1895       \__stex_modules_smodule_end:
1896     }{
1897       \l_tmpa_tl
1898     }
1899   }
1900 }

```

### **\stexpatchmodule**

```

1901 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1902 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1903
1904 \newcommand\stexpatchmodule[3] [] {
1905   \str_set:Nx \l_tmpa_str{ #1 }
1906   \str_if_empty:NTF \l_tmpa_str {
1907     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1908     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1909   }{
1910     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1911     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1912   }
1913 }

```

(End definition for \stexpatchmodule. This function is documented on page 85.)

## 29.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1914 \NewDocumentCommand \STEXModule { m } {
1915   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1916   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1917   \tl_set:Nn \l_tmpa_tl {
1918     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1919   }
1920   \seq_map_inline:Nn \l_stex_all_modules_seq {
1921     \str_set:Nn \l_tmpb_str { ##1 }
1922     \str_if_eq:eeT { \l_tmpa_str } {
1923       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1924     } {
1925       \seq_map_break:n {
1926         \tl_set:Nn \l_tmpa_tl {
1927           \stex_invoke_module:n { ##1 }
1928         }
1929       }
1930     }
1931   }
1932   \l_tmpa_tl
1933 }
1934
1935 \cs_new_protected:Nn \stex_invoke_module:n {
1936   \stex_debug:nn{modules}{Invoking~module~#1}
1937   \peek_charcode_remove:NTF ! {
1938     \__stex_modules_invoke_uri:nN { #1 }
1939   } {
1940     \peek_charcode_remove:NTF ? {
1941       \__stex_modules_invoke_symbol:nn { #1 }
1942     } {
1943       \msg_error:nnx{stex}{error/syntax}{
1944         ?~or~!~expected~after~
1945         \c_backslash_str STEXModule{#1}
1946       }
1947     }
1948   }
1949 }
1950
1951 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1952   \str_set:Nn #2 { #1 }
1953 }
1954
1955 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1956   \stex_invoke_symbol:n{#1?#2}
1957 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 85.)

```

\stex_activate_module:n
1958 \bool_new:N \l_stex_in_meta_bool
1959 \bool_set_false:N \l_stex_in_meta_bool

```

```

1960 \cs_new_protected:Nn \stex_activate_module:n {
1961   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1962     \stex_debug:nn{modules}{Activating~module~#1}
1963     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1964     \use:c{ c_stex_module_#1_code }
1965   }
1966 }

```

(End definition for `\stex_activate_module:n`. This function is documented on page 86.)

`mmtinterface` (env.)

```

1967 \NewDocumentEnvironment { mmtinterface } { 0{} m m } {
1968   \begin{smodule}[#1]{#3}
1969   \str_set:Nx \l_stex_module_mmtfor_str {#2}
1970   \MMTinclude{#2}
1971   \stex_reactivate_macro:N \mmtdecl
1972   \stex_reactivate_macro:N \mmtdef
1973 }{
1974   \end{smodule}
1975 }

1976 \</package>

```

## Chapter 30

# STEX -Module Inheritance Implementation

```
1977 <*package>
1978
1979 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1980
```

### 30.1 SMS Mode

```
1981 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1982 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1983 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1984 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1985
1986 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1987   \makeatletter
1988   \makeatother
1989   \ExplSyntaxOn
1990   \ExplSyntaxOff
1991   \rustexBREAK
1992 }
1993
1994 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1995   \symdef
1996   \importmodule
1997   \notation
1998   \symdecl
1999   \STEXexport
2000   \inlineass
2001   \inlinedef
2002   \inlineex
2003   \endinput
2004   \setnotation
```



```

2005 \copynotation
2006 \assign
2007 \renamedekl
2008 \donotcopy
2009 \instantiate
2010 \textsymdecl
2011 }
2012
2013 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
2014   \tl_to_str:n {
2015     smodule,
2016     copymodule,
2017     interpretmodule,
2018     realization,
2019     sdefinition,
2020     sexample,
2021     sassertion,
2022     sparagraph,
2023     mathstructure,
2024     extstructure,
2025     extstructure*
2026   }
2027 }

```

(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`

```

2028 \bool_new:N \g__stex_smsmode_bool
2029 \bool_set_false:N \g__stex_smsmode_bool
2030 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2031   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2032 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 87.)

`\_stex_smsmode_in_smsmode:nn`

```

2033 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
2034   \vbox_set:Nn \l_tmpa_box {
2035     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
2036     \bool_gset_true:N \g__stex_smsmode_bool
2037     #2
2038     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2039   }
2040   \box_clear:N \l_tmpa_box
2041 } }

```

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

`\stex_file_in_smsmode:nn`

```

2042 \quark_new:N \q__stex_smsmode_break
2043
2044 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
2045   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2046   \stex_smsmode_do:

```

```

2047 }
2048
2049 \cs_new_protected:Nn \__stex_smsmode_module:nn {
2050   \__stex_modules_args:n{#1}
2051   \stex_if_in_module:F {
2052     \str_if_empty:NF \l_stex_module_sig_str {
2053       \stex_modules_current_namespace:
2054       \str_set:Nx \l_stex_module_name_str { #2 }
2055       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2056         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2057         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2058         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2059         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2060         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2061         \str_set:Nx \l_tmpa_str {
2062           \stex_path_to_string:N \l_tmpa_seq /
2063           \l_tmpa_str . \l_stex_module_sig_str .tex
2064         }
2065         \IfFileExists \l_tmpa_str {
2066           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2067         }{
2068           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2069         }
2070       }
2071     }
2072   }
2073 }
2074
2075 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2076   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
2077   \tl_if_empty:nTF{#1}{
2078     \prop_if_exist:NTF \l_stex_current_repository_prop
2079     {
2080       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2081       \prg_return_true:
2082     } {
2083       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2084       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2085       \tl_if_empty:NT \l_tmpa_tl {
2086         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2087       }
2088       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
2089       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2090       \prg_return_true: \prg_return_false:
2091     }
2092   }\prg_return_true:
2093 }
2094
2095 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2096   \stex_filestack_push:n{#1}
2097   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2098   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2099   % ----- new -----
2100   \__stex_smsmode_in_smsmode:nn{#1}{

```

```

2101 \let\importmodule\__stex_smsmode_importmodule:
2102 \let\stex_module_setup:nn\__stex_smsmode_module:nn
2103 \let\__stex_modules_begin_module:\relax
2104 \let\__stex_modules_end_module:\relax
2105 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2106 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2107 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2108 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2109 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2110 \everyeof{\q__stex_smsmode_break\noexpand}
2111 \expandafter\expandafter\expandafter
2112 \stex_smsmode_do:
2113 \csname @ @ input\endcsname "#1"\relax
2114
2115 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2116   \stex_filestack_push:n{##1}
2117   \expandafter\expandafter\expandafter
2118   \stex_smsmode_do:
2119   \csname @ @ input\endcsname "##1"\relax
2120   \stex_filestack_pop:
2121 }
2122 }
2123 % ----- new -----
2124 \__stex_smsmode_in_smsmode:nn{#1} {
2125   #2
2126   % ----- new -----
2127   \begin{group}
2128   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2129   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2130     \__stex_smsmode_check_import_pair:nnT ##1 { \begin{group}
2131       \stex_import_module_uri:nn ##1
2132       \stex_import_require_module:nnnn
2133       \l_stex_import_ns_str
2134       \l_stex_import_archive_str
2135       \l_stex_import_path_str
2136       \l_stex_import_name_str \end{group}
2137     }
2138   }
2139   \end{group}
2140   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
2141   % ----- new -----
2142   \everyeof{\q__stex_smsmode_break\noexpand}
2143   \expandafter\expandafter\expandafter
2144   \stex_smsmode_do:
2145   \csname @ @ input\endcsname "#1"\relax
2146 }
2147 \stex_filestack_pop:
2148 }

```

(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:

```

2149 \cs_new_protected:Npn \stex_smsmode_do: {

```

```

2150 \stex_if_smsmode:T {
2151   \__stex_smsmode_do:w
2152 }
2153 }
2154 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2155   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2156     \expandafter\if\expandafter\relax\noexpand#1
2157     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2158   \else\expandafter\__stex_smsmode_do:w\fi
2159 }{
2160   \__stex_smsmode_do:w % #1
2161 }
2162 }
2163 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2164   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
2165     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2166       #1\__stex_smsmode_do:w
2167     }{
2168       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2169         #1
2170       }{
2171         \cs_if_eq:NNTF \begin #1 {
2172           \__stex_smsmode_check_begin:n
2173         }{
2174           \cs_if_eq:NNTF \end #1 {
2175             \__stex_smsmode_check_end:n
2176           }{
2177             \__stex_smsmode_do:w
2178           }
2179         }
2180       }
2181     }
2182   }
2183 }
2184
2185 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2186   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2187     \begin{#1}
2188   }{
2189     \__stex_smsmode_do:w
2190   }
2191 }
2192 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2193   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2194     \end{#1}\__stex_smsmode_do:w
2195   }{
2196     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2197   }
2198 }

```

(End definition for `\stex_smsmode_do:.` This function is documented on page 88.)

## 30.2 Inheritance

2199 <@@=stex\_importmodule>

\stex\_import\_module\_uri:nn

```
2200 \cs_new_protected:Nn \stex_import_module_uri:nn {
2201   \str_set:Nx \l_stex_import_archive_str { #1 }
2202   \str_set:Nn \l_stex_import_path_str { #2 }
2203
2204   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2205   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2206   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2207
2208   \stex_modules_current_namespace:
2209   \bool_lazy_all:nTF {
2210     {\str_if_empty_p:N \l_stex_import_archive_str}
2211     {\str_if_empty_p:N \l_stex_import_path_str}
2212     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2213   }{
2214     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2215     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2216   }{
2217     \str_if_empty:NT \l_stex_import_archive_str {
2218       \prop_if_exist:NT \l_stex_current_repository_prop {
2219         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2220       }
2221     }
2222     \str_if_empty:NTF \l_stex_import_archive_str {
2223       \str_if_empty:NF \l_stex_import_path_str {
2224         \stex_path_from_string:Nn \l_tmpb_seq {
2225           \l_stex_module_ns_str / .. / \l_stex_import_path_str
2226         }
2227         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2228         \str_replace_once:Nnn \l_stex_import_ns_str {file:} {file:///}
2229       }
2230     }{
2231       \stex_require_repository:n \l_stex_import_archive_str
2232       \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str_manifest_prop } { ns }
2233       \l_stex_import_ns_str
2234       \str_if_empty:NF \l_stex_import_path_str {
2235         \str_set:Nx \l_stex_import_ns_str {
2236           \l_stex_import_ns_str / \l_stex_import_path_str
2237         }
2238       }
2239     }
2240   }
2241 }
```

(End definition for \stex\_import\_module\_uri:nn. This function is documented on page 89.)

\l\_stex\_import\_name\_str Store the return values of \stex\_import\_module\_uri:nn.  
\l\_stex\_import\_archive\_str 2242 \str\_new:N \l\_stex\_import\_name\_str  
\l\_stex\_import\_path\_str 2243 \str\_new:N \l\_stex\_import\_archive\_str  
\l\_stex\_import\_ns\_str 2244 \str\_new:N \l\_stex\_import\_path\_str  
2245 \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 {{\ns}} {{\archive-ID}} {{\path}} {{\name}}
2246 \cs_new_protected:Nn \stex_import_require_module:nnnn {
2247 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2248
2249 \stex_debug:nn{requiremodule}{Here:\\~1:~#1\\~2:~#2\\~3:~#3\\~4:~#4}
2250
2251 \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2252 \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2253
2254 %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
2255
2256 % archive
2257 \str_set:Nx \l_tmpa_str { #2 }
2258 \str_if_empty:NTF \l_tmpa_str {
2259 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2260 \seq_put_right:Nn \l_tmpa_seq {...}
2261 } {
2262 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2263 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2264 \seq_put_right:Nn \l_tmpa_seq { source }
2265 }
2266
2267 % path
2268 \str_set:Nx \l_tmpb_str { #3 }
2269 \str_if_empty:NTF \l_tmpb_str {
2270 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2271
2272 \ltx@ifpackageloaded{babel} {
2273 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2274 { \language } \l_tmpb_str {
2275 \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2276 }
2277 } {
2278 \str_clear:N \l_tmpb_str
2279 }
2280
2281 \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2282 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2283 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2284 }{
2285 \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2286 \IfFileExists{ \l_tmpa_str.tex }{
2287 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2288 }{
2289 % try english as default
2290 \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2291 \IfFileExists{ \l_tmpa_str.en.tex }{
2292 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2293 }{
2294 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2295 }
2296 }
2297 }
2298

```

```

2299 } {
2300   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2301   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2302
2303   \ltx@ifpackageloaded{babel} {
2304     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2305       { \language } \l_tmpb_str {
2306       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2307     }
2308   } {
2309     \str_clear:N \l_tmpb_str
2310   }
2311
2312   \stex_path_canonicalize:N \l_tmpb_seq
2313   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2314
2315   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2316   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2317     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2318   }{
2319     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2320     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2321       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2322     }{
2323       % try english as default
2324       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2325       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2326         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2327       }{
2328         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2329         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2330           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2331         }{
2332           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2333           \IfFileExists{ \l_tmpa_str.tex }{
2334             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2335           }{
2336             % try english as default
2337             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2338             \IfFileExists{ \l_tmpa_str.en.tex }{
2339               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2340             }{
2341               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2342             }
2343           }
2344         }
2345       }
2346     }
2347   }
2348 }
2349
2350 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2351   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2352     \seq_clear:N \l_stex_all_modules_seq

```

```

2353     \str_clear:N \l_stex_current_module_str
2354     \str_set:Nx \l_tmpb_str { #2 }
2355     \str_if_empty:NF \l_tmpb_str {
2356       \stex_set_current_repository:n { #2 }
2357     }
2358     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2359   }
2360
2361   \stex_if_module_exists:nF { #1 ? #4 } {
2362     \msg_error:nnx{stex}{error/unknownmodule}{
2363       #1?#4~(in~file~\g__stex_importmodule_file_str)
2364     }
2365   }
2366 }
2367
2368 }
2369 \stex_activate_module:n { #1 ? #4 }
2370 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 89.)

## `\importmodule`

```

2371 \NewDocumentCommand \importmodule { 0{} m } {
2372   \stex_import_module_uri:nn { #1 } { #2 }
2373   \stex_debug:nn{modules}{Importing~module:~
2374     \l_stex_import_ns_str ? \l_stex_import_name_str
2375   }
2376   \stex_if_smsmode:F {
2377     \stex_annotate_invisible:nnn
2378     {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2379   }
2380   \stex_execute_in_module:x {
2381     \stex_import_require_module:nnnn
2382     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2383     { \l_stex_import_path_str } { \l_stex_import_name_str }
2384   }
2385   \exp_args:Nx \stex_add_import_to_current_module:n {
2386     \l_stex_import_ns_str ? \l_stex_import_name_str
2387   }
2388   \stex_smsmode_do:
2389   \ignorespacesandpars
2390 }
2391 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

(End definition for `\importmodule`. This function is documented on page 88.)

## `\usemodule`

```

2392 \NewDocumentCommand \usemodule { 0{} m } {
2393   \stex_if_smsmode:F {
2394     \stex_import_module_uri:nn { #1 } { #2 }
2395     \stex_import_require_module:nnnn
2396     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2397     { \l_stex_import_path_str } { \l_stex_import_name_str }
2398     \stex_annotate_invisible:nnn
2399     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}

```



```

2400 }
2401 \stex_smsmode_do:
2402 \ignorespacesandpars
2403 }

```

*(End definition for \usemodule. This function is documented on page 88.)*

```

2404 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2405   \tl_if_empty:nF{#2}{
2406     \clist_set:Nn \l_tmpa_clist {#2}
2407     \clist_map_inline:Nn \l_tmpa_clist {
2408       \tl_if_head_eq_charcode:nNTF {##1}[{
2409         #1 ##1
2410       }{
2411         #1{##1}
2412       }
2413     }
2414   }
2415 }
2416 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2417
2418
2419 </package>

```

## Chapter 31

# STEX -Symbols Implementation

```
2420 <*package>
2421
2422 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
2423
2424 Warnings and error messages
2425 \msg_new:nnn{stex}{error/wrongargs}{
2426   args~value~in~symbol~declaration~for~#1~
2427   needs~to~be~i,~a,~b~or~B,~but~#2~given
2428 }
2429 \msg_new:nnn{stex}{error/unknownsymbol}{
2430   No~symbol~#1~found!
2431 }
2432 \msg_new:nnn{stex}{error/seqlength}{
2433   Expected~#1~arguments;~got~#2!
2434 }
2435 \msg_new:nnn{stex}{error/unknownnotation}{
2436   Unknown~notation~#1~for~#2!
2437 }
```

### 31.1 Symbol Declarations

```
2437 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2438 \cs_new_protected:Nn \stex_all_symbols:n {
2439   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2440   \seq_map_inline:Nn \l_stex_all_modules_seq {
2441     \seq_map_inline:cn{c_stex_module_##1_constants}{
2442       \__stex_symdecl_all_symbols_cs{##1?####1}
2443     }
2444   }
2445 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page 91.)

## `\STEXsymbol`

```
2446 \NewDocumentCommand \STEXsymbol { m } {
2447   \stex_get_symbol:n { #1 }
2448   \exp_args:No
2449   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2450 }
```

(End definition for `\STEXsymbol`. This function is documented on page 92.)

`symdecl` arguments:

```
2451 \keys_define:nn { stex / symdecl } {
2452   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2453   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2454   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2455   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2456   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
2457   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
2458   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2459   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2460   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
2461   assoc     .choices:nn  =
2462             {bin,binl,binr,pre,conj,pwconj}
2463             {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2464 }
2465
2466 \bool_new:N \l_stex_symdecl_make_macro_bool
2467
2468 \cs_new_protected:Nn \__stex_symdecl_args:n {
2469   \str_clear:N \l_stex_symdecl_name_str
2470   \str_clear:N \l_stex_symdecl_args_str
2471   \str_clear:N \l_stex_symdecl_deprecate_str
2472   \str_clear:N \l_stex_symdecl_reorder_str
2473   \str_clear:N \l_stex_symdecl_assoctype_str
2474   \bool_set_false:N \l_stex_symdecl_local_bool
2475   \tl_clear:N \l_stex_symdecl_type_tl
2476   \tl_clear:N \l_stex_symdecl_definiens_tl
2477   \clist_clear:N \l_stex_symdecl_argnames_clist
2478
2479   \keys_set:nn { stex / symdecl } { #1 }
2480 }
```

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

```
2481
2482 \NewDocumentCommand \symdecl { s m O{} } {
2483   \__stex_symdecl_args:n { #3 }
2484   \IfBooleanTF #1 {
2485     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2486   } {
2487     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2488   }
2489   \stex_symdecl_do:n { #2 }
2490   \stex_smsmode_do:
2491 }
```

```

2492
2493 \cs_new_protected:Nn \stex_symdecl_do:nn {
2494   \__stex_symdecl_args:n{#1}
2495   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2496   \stex_symdecl_do:n{#2}
2497 }
2498
2499 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 90.)

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

```

2500 \cs_new_protected:Nn \stex_symdecl_do:n {
2501   \stex_if_in_module:F {
2502     % TODO throw error? some default namespace?
2503   }
2504
2505   \str_if_empty:NT \l_stex_symdecl_name_str {
2506     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2507   }
2508
2509   \prop_if_exist:cT { l_stex_symdecl_
2510     \l_stex_current_module_str ?
2511     \l_stex_symdecl_name_str
2512     _prop
2513   }{
2514     % TODO throw error (beware of circular dependencies)
2515   }
2516
2517   \prop_clear:N \l_tmpa_prop
2518   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2519   \seq_clear:N \l_tmpa_seq
2520   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2521   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2522
2523   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2524     \str_if_empty:NF \l_stex_module_deprecate_str {
2525       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2526     }
2527   }
2528   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2529
2530   \exp_args:No \stex_add_constant_to_current_module:n {
2531     \l_stex_symdecl_name_str
2532   }
2533
2534   % arity/args
2535   \int_zero:N \l_tmpb_int
2536
2537   \bool_set_true:N \l_tmpa_bool
2538   \str_map_inline:Nn \l_stex_symdecl_args_str {
2539     \token_case_meaning:NnF ##1 {
2540       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2541       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2542     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2543     {\tl_to_str:n a} {
2544         \bool_set_false:N \l_tmpa_bool
2545         \int_incr:N \l_tmpb_int
2546     }
2547     {\tl_to_str:n B} {
2548         \bool_set_false:N \l_tmpa_bool
2549         \int_incr:N \l_tmpb_int
2550     }
2551 }{
2552     \msg_error:nnxx{stex}{error/wrongargs}{
2553         \l_stex_current_module_str ?
2554         \l_stex_symdecl_name_str
2555     }{##1}
2556 }
2557 }
2558
2559 \bool_if:NTF \l_tmpa_bool {
2560     % possibly numeric
2561     \str_if_empty:NTF \l_stex_symdecl_args_str {
2562         \prop_put:Nnn \l_tmpa_prop { args } {}
2563         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2564     }{
2565         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2566         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2567         \str_clear:N \l_tmpa_str
2568         \int_step_inline:nn \l_tmpa_int {
2569             \str_put_right:Nn \l_tmpa_str i
2570         }
2571         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2572     }
2573 } {
2574     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2575     \prop_put:Nnx \l_tmpa_prop { arity }
2576     { \str_count:N \l_stex_symdecl_args_str }
2577 }
2578 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2579
2580 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2581     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2582 }{
2583     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2584 }
2585
2586 % argnames
2587
2588 \clist_clear:N \l_tmpa_clist
2589 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
2590     \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
2591         \clist_put_right:Nn \l_tmpa_clist {##1}
2592     }{
2593         \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl
2594         \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
2595     }

```

```

2596 }
2597 \prop_put:Nn \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
2598
2599 % semantic macro
2600
2601 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2602   \exp_args:Nx \stex_do_up_to_module:n {
2603     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2604       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2605     }}
2606   }
2607 }
2608
2609 \stex_debug:nn{symbols}{New~symbol:~
2610   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2611   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2612   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2613   Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2614 }
2615
2616 % circular dependencies require this:
2617 \stex_if_do_html:T {
2618   \stex_annotate_invisible:nnn {symdecl} {
2619     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2620   } {
2621     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2622       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2623     }
2624     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2625     \stex_annotate_invisible:nnn{macroname}{#1}{}
2626     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2627       \stex_annotate_invisible:nnn{definiens}{}
2628       {\l_stex_symdecl_definiens_tl$}
2629     }
2630     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2631       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2632     }
2633     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2634       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2635     }
2636   }
2637 }
2638 \prop_if_exist:cF {
2639   \l_stex_symdecl_
2640   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2641   _prop
2642 } {
2643   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2644     \__stex_symdecl_restore_symbol:nnnnnnnn
2645       {\l_stex_symdecl_name_str}
2646       { \prop_item:Nn \l_tmpa_prop {args} }
2647       { \prop_item:Nn \l_tmpa_prop {arity} }
2648       { \prop_item:Nn \l_tmpa_prop {assoc} }
2649       { \prop_item:Nn \l_tmpa_prop {defined} }

```

```

2650         {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2651         {\l_stex_current_module_str}
2652         { \prop_item:Nn \l_tmpa_prop {argnames} }
2653     }
2654 }
2655 }
2656 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2657     \prop_clear:N \l_tmpa_prop
2658     \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2659     \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2660     \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2661     \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2662     \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2663     \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2664     \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2665     \tl_if_empty:nF{#6}{
2666         \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2667     }
2668     \prop_set_eq:cN{\l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2669     \seq_clear:c{\l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2670 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 91.)

## `\textsymdecl`

```

2671
2672 \keys_define:nn { stex / textsymdecl } {
2673     name      .str_set_x:N = \l__stex_symdecl_name_str ,
2674     type      .tl_set:N    = \l__stex_symdecl_type_tl
2675 }
2676
2677 \cs_new_protected:Nn \stex_textsymdecl_args:n {
2678     \str_clear:N \l__stex_symdecl_name_str
2679     \tl_clear:N \l__stex_symdecl_type_tl
2680     \clist_clear:N \l_stex_symdecl_argnames_clist
2681     \keys_set:nn { stex / textsymdecl } { #1 }
2682 }
2683
2684 \NewDocumentCommand \textsymdecl {m O{} m} {
2685     \stex_textsymdecl_args:n { #2 }
2686     \str_if_empty:NTF \l__stex_symdecl_name_str {
2687         \__stex_symdecl_args:n{name=#1,#2}
2688     }{
2689         \__stex_symdecl_args:n{#2}
2690     }
2691     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2692     \stex_symdecl_do:n{#1-sym}
2693     \stex_execute_in_module:n{
2694         \cs_set_nopar:cpn{#1name}{
2695             \ifvmode\hbox_unpack:N\c_empty_box\fi
2696             \ifmmode\hbox{#3}\else#3\fi\hspace
2697         }
2698         \cs_set_nopar:cpn{#1}{
2699             \ifmmode\csname#1-sym\expandafter\endcsname\else

```

```

2700     \ifvmode\hbox_unpack:N\c_empty_box\fi
2701     \symref{#1-sym}{#3}\expandafter\xspace
2702     \fi
2703   }
2704 }
2705 \stex_execute_in_module:x{
2706   \__stex_notation_restore_notation:nnnnn
2707   {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl_name_str{#1}}{0}
2708   {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{\neginfprec}}{0}
2709   \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2710   }}}
2711   {}
2712 }
2713 }
2714 \stex_smsmode_do:
2715 }

```

(End definition for `\textsymdecl`. This function is documented on page 23.)

`\stex_get_symbol:n`

```

2716 \str_new:N \l_stex_get_symbol_uri_str
2717
2718 \cs_new_protected:Nn \stex_get_symbol:n {
2719   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2720     \tl_set:Nn \l_tmpa_tl { #1 }
2721     \__stex_symdecl_get_symbol_from_cs:
2722   }{
2723     % argument is a string
2724     % is it a command name?
2725     \cs_if_exist:cTF { #1 }{
2726       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2727       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2728       \str_if_empty:NTF \l_tmpa_str {
2729         \exp_args:Nx \cs_if_eq:NNTF {
2730           \tl_head:N \l_tmpa_tl
2731         } \stex_invoke_symbol:n {
2732           \__stex_symdecl_get_symbol_from_cs:
2733         }{
2734           \__stex_symdecl_get_symbol_from_string:n { #1 }
2735         }
2736       } {
2737         \__stex_symdecl_get_symbol_from_string:n { #1 }
2738       }
2739     }{
2740       % argument is not a command name
2741       \__stex_symdecl_get_symbol_from_string:n { #1 }
2742       % \l_stex_all_symbols_seq
2743     }
2744   }
2745   \str_if_eq:eeF {
2746     \prop_item:cn {
2747       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2748     }{ deprecate }
2749   }{}{

```



```

2750     \msg_warning:nnxx{stex}{warning/deprecated}{
2751       Symbol~\l_stex_get_symbol_uri_str
2752     }{
2753       \prop_item:cn {l_stex_symdecl_l\l_stex_get_symbol_uri_str _prop}{ deprecate }
2754     }
2755   }
2756 }
2757
2758 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2759   \tl_set:Nn \l_tmpa_tl {
2760     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2761   }
2762   \str_set:Nn \l_tmpa_str { #1 }
2763
2764   %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2765
2766   \str_if_in:NnTF \l_tmpa_str ? {
2767     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2768     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2769     \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2770   }{
2771     \str_clear:N \l_tmpb_str
2772   }
2773   \str_if_empty:NnTF \l_tmpb_str {
2774     \seq_map_inline:Nn \l_stex_all_modules_seq {
2775       \seq_map_inline:cn{c_stex_module_##1_constants}{
2776         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2777           \seq_map_break:n{\seq_map_break:n{
2778             \tl_set:Nn \l_tmpa_tl {
2779               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2780             }
2781           }}
2782         }
2783       }
2784     }
2785   }{
2786     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2787     \seq_map_inline:Nn \l_stex_all_modules_seq {
2788       \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2789         \seq_map_inline:cn{c_stex_module_##1_constants}{
2790           \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2791             \seq_map_break:n{\seq_map_break:n{
2792               \tl_set:Nn \l_tmpa_tl {
2793                 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2794               }
2795             }}
2796           }
2797         }
2798       }
2799     }
2800   }
2801
2802   \l_tmpa_tl
2803 }

```

```

2804
2805 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2806   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2807     { \tl_tail:N \l_tmpa_tl }
2808   \tl_if_single:NTF \l_tmpa_tl {
2809     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2810       \exp_after:wN \str_set:Nn \exp_after:wN
2811         \l_stex_get_symbol_uri_str \l_tmpa_tl
2812     }{
2813       % TODO
2814       % tail is not a single group
2815     }
2816   }{
2817     % TODO
2818     % tail is not a single group
2819   }
2820 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 91.)

## 31.2 Notations

```

2821 <@@=stex_notation>
      notation arguments:
2822 \keys_define:nn { stex / notation } {
2823   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2824   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2825   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2826   op       .tl_set:N = \l__stex_notation_op_tl ,
2827   primary .bool_set:N = \l__stex_notation_primary_bool ,
2828   primary .default:n = {true} ,
2829   hints    .str_set_x:N = \l__stex_notation_hints_str,
2830   unknown .code:n = \str_set:Nx
2831     \l__stex_notation_variant_str \l_keys_key_str
2832 }
2833
2834 \cs_new_protected:Nn \_stex_notation_args:n {
2835   % \str_clear:N \l__stex_notation_lang_str
2836   \str_clear:N \l__stex_notation_variant_str
2837   \str_clear:N \l__stex_notation_prec_str
2838   \str_clear:N \l__stex_notation_hints_str
2839   \tl_clear:N \l__stex_notation_op_tl
2840   \bool_set_false:N \l__stex_notation_primary_bool
2841
2842   \keys_set:nn { stex / notation } { #1 }
2843 }

```

**\notation**

```

2844 \NewDocumentCommand \notation { s m O{}} {
2845   \_stex_notation_args:n { #3 }
2846   \tl_clear:N \l_stex_symdecl_definiens_tl
2847   \stex_get_symbol:n { #2 }
2848   \tl_set:Nn \l_stex_notation_after_do_tl {

```

```

2849 \__stex_notation_final:
2850 \IfBooleanTF#1{
2851   \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2852 }{}
2853 \stex_smsmode_do:\ignorespacesandpars
2854 }
2855 \stex_notation_do:nnnnn
2856 { \prop_item:cn {\l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2857 { \prop_item:cn { \l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2858 { \l__stex_notation_variant_str }
2859 { \l__stex_notation_prec_str }
2860 }
2861 \stex_deactivate_macro:Nn \notation {module-environments}

```

(End definition for \notation. This function is documented on page 91.)

\stex\_notation\_do:nnnnn

```

2862 \seq_new:N \l__stex_notation_precedences_seq
2863 \tl_new:N \l__stex_notation_opprec_tl
2864 \int_new:N \l__stex_notation_currarg_int
2865 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2866
2867 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2868   \let\STEXInternalCurrentSymbolStr\relax
2869   \seq_clear:N \l__stex_notation_precedences_seq
2870   \tl_clear:N \l__stex_notation_opprec_tl
2871   \str_set:Nx \l__stex_notation_args_str { #1 }
2872   \str_set:Nx \l__stex_notation_arity_str { #2 }
2873   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2874   \str_set:Nx \l__stex_notation_prec_str { #4 }
2875
2876   % precedences
2877   \str_if_empty:NTF \l__stex_notation_prec_str {
2878     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2879       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2880     }{
2881       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2882     }
2883   } {
2884     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2885       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2886       \int_step_inline:nn { \l__stex_notation_arity_str } {
2887         \exp_args:NNo
2888         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2889       }
2890     }{
2891       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2892       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2893         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2894         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2895           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2896             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2897           \seq_map_inline:Nn \l_tmpa_seq {
2898             \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }

```

```

2899     }
2900   }
2901   }{
2902     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2903       \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2904     }{
2905       \tl_set:No \l__stex_notation_opprec_tl { 0 }
2906     }
2907   }
2908 }
2909 }
2910
2911 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2912 \int_step_inline:nn { \l__stex_notation_arity_str } {
2913   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2914     \exp_args:NNo
2915     \seq_put_right:No \l__stex_notation_precedences_seq {
2916       \l__stex_notation_opprec_tl
2917     }
2918   }
2919 }
2920 \tl_clear:N \l_stex_notation_dummyargs_tl
2921
2922 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2923   \exp_args:NNe
2924   \cs_set:Npn \l_stex_notation_macrocode_cs {
2925     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2926     { \l__stex_notation_suffix_str }
2927     { \l__stex_notation_opprec_tl }
2928     { \exp_not:n { #5 } }
2929   }
2930   \l_stex_notation_after_do_tl
2931 }{
2932   \str_if_in:NnTF \l__stex_notation_args_str b {
2933     \exp_args:Nne \use:nn
2934     {
2935       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2936       \cs_set:Npn \l__stex_notation_arity_str } { {
2937         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2938         { \l__stex_notation_suffix_str }
2939         { \l__stex_notation_opprec_tl }
2940         { \exp_not:n { #5 } }
2941       }
2942     }{
2943       \str_if_in:NnTF \l__stex_notation_args_str B {
2944         \exp_args:Nne \use:nn
2945         {
2946           \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2947           \cs_set:Npn \l__stex_notation_arity_str } { {
2948             \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2949             { \l__stex_notation_suffix_str }
2950             { \l__stex_notation_opprec_tl }
2951             { \exp_not:n { #5 } }
2952           } }

```

```

2953     }{
2954       \exp_args:Nne \use:nn
2955       {
2956         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2957         \cs_set:Npn \l__stex_notation_arity_str } { {
2958           \STEXInternalTermMathOMAiiai { \STEXInternalCurrentSymbolStr }
2959           { \l__stex_notation_suffix_str }
2960           { \l__stex_notation_opprec_tl }
2961           { \exp_not:n { #5 } }
2962         } }
2963       }
2964     }
2965
2966     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2967     \int_zero:N \l__stex_notation_currarg_int
2968     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2969     \__stex_notation_arguments:
2970   }
2971 }

```

(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

```

2972 \cs_new_protected:Nn \__stex_notation_arguments: {
2973   \int_incr:N \l__stex_notation_currarg_int
2974   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2975     \l_stex_notation_after_do_tl
2976   }{
2977     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2978     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2979     \str_if_eq:VnTF \l_tmpa_str a {
2980       \__stex_notation_argument_assoc:nn{a}
2981     }{
2982       \str_if_eq:VnTF \l_tmpa_str B {
2983         \__stex_notation_argument_assoc:nn{B}
2984       }{
2985         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2986         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2987           { \STEXInternalTermMathArgiii
2988             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2989             { \l_tmpb_str }
2990             { ###\int_use:N \l__stex_notation_currarg_int }
2991           }
2992         }
2993         \__stex_notation_arguments:
2994       }
2995     }
2996   }
2997 }

```

(End definition for `\__stex_notation_arguments:.`)

`\__stex_notation_argument_assoc:nn`

```

2998 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {

```

```

2999
3000 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
3001   {\l__stex_notation_arity_str}{
3002     #2
3003   }
3004 \int_zero:N \l_tmpa_int
3005 \tl_clear:N \l_tmpa_tl
3006 \str_map_inline:Nn \l__stex_notation_args_str {
3007   \int_incr:N \l_tmpa_int
3008   \tl_put_right:Nx \l_tmpa_tl {
3009     \str_if_eq:nnTF {##1}{a}{ } {} }{
3010     \str_if_eq:nnTF {##1}{B}{ } {} }{
3011     {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_int}
3012   }
3013 }
3014 }
3015 }
3016 \exp_after:wN\exp_after:wN\exp_after:wN \def
3017 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
3018 \exp_after:wN\exp_after:wN\exp_after:wN ##
3019 \exp_after:wN\exp_after:wN\exp_after:wN 1
3020 \exp_after:wN\exp_after:wN\exp_after:wN ##
3021 \exp_after:wN\exp_after:wN\exp_after:wN 2
3022 \exp_after:wN\exp_after:wN\exp_after:wN {
3023   \exp_after:wN \exp_after:wN \exp_after:wN
3024   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
3025     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3026   }
3027 }
3028
3029 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
3030 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
3031   \STEXInternalTermMathAssocArgiiii
3032   { \int_use:N \l__stex_notation_currarg_int }
3033   { \l_tmpa_str }
3034   { ####\int_use:N \l__stex_notation_currarg_int }
3035   { \l_tmpa_cs {####1} {####2} }
3036   {#1}
3037 } }
3038 \__stex_notation_arguments:
3039 }

```

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

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

```

3040 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
3041   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
3042   \cs_set_nopar:Npn {#3}{#4}
3043   \tl_if_empty:nF {#5}{
3044     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
3045   }
3046   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
3047     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
3048   }

```

```

3049 }
3050
3051 \cs_new_protected:Nn \__stex_notation_final: {
3052
3053   \stex_execute_in_module:x {
3054     \__stex_notation_restore_notation:nnnnn
3055     {\l_stex_get_symbol_uri_str}
3056     {\l__stex_notation_suffix_str}
3057     {\l__stex_notation_arity_str}
3058     {
3059       \exp_after:wN \exp_after:wN \exp_after:wN
3060       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3061       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3062     }
3063     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3064   }
3065
3066   \stex_debug:nn{symbols}{
3067     Notation~\l__stex_notation_suffix_str
3068     ~for~\l_stex_get_symbol_uri_str^^J
3069     Operator~precedence:~\l__stex_notation_opprec_tl^^J
3070     Argument~precedences:~
3071     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3072     Notation: \cs_meaning:c {
3073       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3074       \l__stex_notation_suffix_str
3075       _cs
3076     }
3077   }
3078   % HTML annotations
3079   \stex_if_do_html:T {
3080     \stex_annotate_invisible:nnn { notation }
3081     { \l_stex_get_symbol_uri_str } {
3082       \stex_annotate_invisible:nnn { notationfragment }
3083       { \l__stex_notation_suffix_str }{}
3084       \stex_annotate_invisible:nnn { precedence }
3085       { \l__stex_notation_prec_str }{}
3086
3087       \int_zero:N \l_tmpa_int
3088       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3089       \tl_clear:N \l_tmpa_tl
3090       \int_step_inline:nn { \l__stex_notation_arity_str }{
3091         \int_incr:N \l_tmpa_int
3092         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3093         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
3094         \str_if_eq:VnTF \l_tmpb_str a {
3095           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3096             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3097             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3098           } }
3099         }{
3100           \str_if_eq:VnTF \l_tmpb_str B {
3101             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3102               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,

```

```

3103         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{\}
3104     } }
3105     }{
3106         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3107             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{\}
3108         } }
3109     }
3110 }
3111 }
3112 \stex_annotate_invisible:nnn { notationcomp }{\}{
3113     \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3114     $ \exp_args:Nno \use:nn { \use:c {
3115         stex_notation_ \STEXInternalCurrentSymbolStr
3116         \c_hash_str \l__stex_notation_suffix_str_cs
3117     } } { \l_tmpa_tl } $
3118 }
3119 \tl_if_empty:NF \l__stex_notation_op_tl {
3120     \stex_annotate_invisible:nnn { notationopcomp }{\}{
3121         $\l__stex_notation_op_tl$
3122     }
3123 }
3124 }
3125 }
3126 }

```

(End definition for `\_stex_notation_final:`)

## `\setnotation`

```

3127 \keys_define:nn { stex / setnotation } {
3128 % lang .tl_set_x:N = \l__stex_notation_lang_str ,
3129 variant .tl_set_x:N = \l__stex_notation_variant_str ,
3130 unknown .code:n = \str_set:Nx
3131     \l__stex_notation_variant_str \l_keys_key_str
3132 }
3133
3134 \cs_new_protected:Nn \_stex_setnotation_args:n {
3135 % \str_clear:N \l__stex_notation_lang_str
3136 \str_clear:N \l__stex_notation_variant_str
3137 \keys_set:nn { stex / setnotation } { #1 }
3138 }
3139
3140 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
3141 \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
3142     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
3143     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
3144 }
3145 }
3146
3147 \cs_new_protected:Nn \stex_setnotation:n {
3148 \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
3149 { \l__stex_notation_variant_str }{
3150     \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
3151     \stex_debug:nn {notations}{
3152         Setting~default~notation~

```



```

3153         {\l__stex_notation_variant_str }~for~
3154         #1 \\\
3155         \expandafter\meaning\csname
3156         l_stex_symdecl_#1 _notations\endcsname
3157     }
3158 }{
3159     \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
3160 }
3161 }
3162
3163 \NewDocumentCommand \setnotation {m m} {
3164     \stex_get_symbol:n { #1 }
3165     \_stex_setnotation_args:n { #2 }
3166     \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3167     \stex_smsmode_do:\ignorespacesandpars
3168 }
3169
3170 \cs_new_protected:Nn \stex_copy_notations:nn {
3171     \stex_debug:nn {notations}{
3172         Copying~notations~from~#2~to~#1\\
3173         \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3174     }
3175     \tl_clear:N \l_tmpa_tl
3176     \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3177         \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
3178     }
3179     \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3180         \stex_debug:nn{Here}{Here:~##1}
3181         \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3182         \edef \l_tmpa_tl {
3183             \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3184             \exp_after:wN\exp_after:wN\exp_after:wN {
3185                 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3186             }
3187         }
3188
3189         \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3190         \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
3191         \exp_after:wN { \l_tmpa_tl }
3192
3193         \edef \l_tmpa_tl {
3194             \exp_after:wN \exp_not:n \exp_after:wN {
3195                 \l_tmpa_tl {##### 1}{##### 2}
3196             }
3197         }
3198
3199         \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3200
3201     \stex_execute_in_module:x {
3202         \_stex_notation_restore_notation:nnnnn
3203         {#1}{##1}
3204         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
3205         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
3206         {

```

```

3207         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3208             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3209             }
3210         }
3211     }\endgroup
3212 }
3213 }
3214
3215 \NewDocumentCommand \copynotation {m m} {
3216     \stex_get_symbol:n { #1 }
3217     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3218     \stex_get_symbol:n { #2 }
3219     \exp_args:Noo
3220     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3221     \stex_smsmode_do:\ignorespacesandpars
3222 }
3223

```

(End definition for \setnotation. This function is documented on page 23.)

**\symdef**

```

3224 \keys_define:nn { stex / symdef } {
3225     name .str_set_x:N = \l_stex_symdecl_name_str ,
3226     args .str_set_x:N = \l_stex_symdecl_args_str ,
3227     type .tl_set:N = \l_stex_symdecl_type_tl ,
3228     def .tl_set:N = \l_stex_symdecl_definiens_tl ,
3229     reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3230     op .tl_set:N = \l__stex_notation_op_tl ,
3231     % lang .str_set_x:N = \l__stex_notation_lang_str ,
3232     variant .str_set_x:N = \l__stex_notation_variant_str ,
3233     prec .str_set_x:N = \l__stex_notation_prec_str ,
3234     argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3235     assoc .choices:nn =
3236         {bin,binl,binr,pre,conj,pwconj}
3237         {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3238     unknown .code:n = \str_set:Nx
3239         \l__stex_notation_variant_str \l_keys_key_str
3240 }
3241
3242 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3243     \str_clear:N \l_stex_symdecl_name_str
3244     \str_clear:N \l_stex_symdecl_args_str
3245     \str_clear:N \l_stex_symdecl_assoctype_str
3246     \str_clear:N \l_stex_symdecl_reorder_str
3247     \bool_set_false:N \l_stex_symdecl_local_bool
3248     \tl_clear:N \l_stex_symdecl_type_tl
3249     \tl_clear:N \l_stex_symdecl_definiens_tl
3250     \clist_clear:N \l_stex_symdecl_argnames_clist
3251     % \str_clear:N \l__stex_notation_lang_str
3252     \str_clear:N \l__stex_notation_variant_str
3253     \str_clear:N \l__stex_notation_prec_str
3254     \tl_clear:N \l__stex_notation_op_tl
3255
3256     \keys_set:nn { stex / symdef } { #1 }

```

```

3257 }
3258
3259 \NewDocumentCommand \symdef { m O{} } {
3260   \__stex_notation_symdef_args:n { #2 }
3261   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3262   \stex_symdecl_do:n { #1 }
3263   \tl_set:Nn \l_stex_notation_after_do_tl {
3264     \__stex_notation_final:
3265     \stex_smsmode_do:\ignorespacesandpars
3266   }
3267   \str_set:Nx \l_stex_get_symbol_uri_str {
3268     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3269   }
3270   \exp_args:Nx \stex_notation_do:nnnnn
3271     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { args } }
3272     { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
3273     { \l__stex_notation_variant_str }
3274     { \l__stex_notation_prec_str}
3275 }
3276 \stex_deactivate_macro:Nn \symdef {module~environments}
3277
3278 \keys_define:nn { stex / mmtdef } {
3279   name .str_set_x:N = \l_stex_symdecl_name_str ,
3280   args .str_set_x:N = \l_stex_symdecl_args_str ,
3281   reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3282   op .tl_set:N = \l__stex_notation_op_tl ,
3283   % lang .str_set_x:N = \l__stex_notation_lang_str ,
3284   variant .str_set_x:N = \l__stex_notation_variant_str ,
3285   prec .str_set_x:N = \l__stex_notation_prec_str ,
3286   argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3287   assoc .choices:nn =
3288     {bin,binl,binr,pre,conj,pwconj}
3289     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3290   unknown .code:n = \str_set:Nx
3291     \l__stex_notation_variant_str \l_keys_key_str
3292 }
3293 \cs_new_protected:Nn \_stex_mmtdef_args:n {
3294   \str_clear:N \l_stex_symdecl_name_str
3295   \str_clear:N \l_stex_symdecl_args_str
3296   \str_clear:N \l_stex_symdecl_assoctype_str
3297   \str_clear:N \l_stex_symdecl_reorder_str
3298   \bool_set_false:N \l_stex_symdecl_local_bool
3299   \clist_clear:N \l_stex_symdecl_argnames_clist
3300   % \str_clear:N \l__stex_notation_lang_str
3301   \str_clear:N \l__stex_notation_variant_str
3302   \str_clear:N \l__stex_notation_prec_str
3303   \tl_clear:N \l__stex_notation_op_tl
3304
3305   \keys_set:nn { stex / mmtdef } { #1 }
3306 }
3307
3308 \NewDocumentCommand \mmtdef {m O{} }{
3309   \_stex_mmtdef_args:n{ #2 }
3310   \bool_set_true:N \l_stex_symdecl_make_macro_bool

```

```

3311 \str_if_empty:NT \l_stex_symdecl_name_str {
3312   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3313 }
3314 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3315 %  \stex_annotate:nnn{ OMID }{
3316 %    \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3317 %  }{}
3318 %}
3319 \stex_symdecl_do:n { #1 }
3320 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3321   \stex_annotate:nnn{ OMID }{
3322     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3323   }{},
3324   \stex_annotate:nnn{ OMID }{
3325     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3326   }{}
3327 }
3328 \tl_set:Nn \l_stex_notation_after_do_tl {
3329   \__stex_notation_final:
3330   \stex_smsmode_do:\ignorespacesandpars
3331 }
3332 \str_set:Nx \l_stex_get_symbol_uri_str {
3333   \l_stex_current_module_str ? \l_stex_symdecl_name_str
3334 }
3335 \exp_args:Nx \stex_notation_do:nnnnn
3336 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { args } }
3337 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
3338 { \l__stex_notation_variant_str }
3339 { \l__stex_notation_prec_str}
3340 }

```

(End definition for `\symdef`. This function is documented on page 91.)

### 31.3 Variables

```

3341 <@@=stex_variables>
3342
3343 \keys_define:nn { stex / vardef } {
3344   name .str_set_x:N = \l__stex_variables_name_str ,
3345   args .str_set_x:N = \l__stex_variables_args_str ,
3346   type .tl_set:N    = \l__stex_variables_type_tl ,
3347   def .tl_set:N     = \l__stex_variables_def_tl ,
3348   op .tl_set:N      = \l__stex_variables_op_tl ,
3349   prec .str_set_x:N = \l__stex_variables_prec_str ,
3350   reorder .str_set_x:N = \l__stex_variables_reorder_str ,
3351   argnames .clist_set:N = \l__stex_variables_argnames_clist ,
3352   assoc .choices:nn =
3353     {bin,binl,binr,pre,conj,pwconj}
3354     {\str_set:Nx \l__stex_variables ASSOCTYPE_str {\l_keys_choice_tl}},
3355   bind .choices:nn =
3356     {forall,exists}
3357     {\str_set:Nx \l__stex_variables BIND_str {\l_keys_choice_tl}}
3358 }
3359

```

```

3360 \cs_new_protected:Nn \__stex_variables_args:n {
3361   \str_clear:N \l__stex_variables_name_str
3362   \str_clear:N \l__stex_variables_args_str
3363   \str_clear:N \l__stex_variables_prec_str
3364   \str_clear:N \l__stex_variables_assoctype_str
3365   \str_clear:N \l__stex_variables_reorder_str
3366   \str_clear:N \l__stex_variables_bind_str
3367   \tl_clear:N \l__stex_variables_type_tl
3368   \tl_clear:N \l__stex_variables_def_tl
3369   \tl_clear:N \l__stex_variables_op_tl
3370   \clist_clear:N \l__stex_variables_argnames_clist
3371
3372   \keys_set:nn { stex / vardef } { #1 }
3373 }
3374
3375 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3376   \__stex_variables_args:n {#2}
3377   \str_if_empty:NT \l__stex_variables_name_str {
3378     \str_set:Nx \l__stex_variables_name_str { #1 }
3379   }
3380   \prop_clear:N \l_tmpa_prop
3381   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3382
3383   \int_zero:N \l_tmpb_int
3384   \bool_set_true:N \l_tmpa_bool
3385   \str_map_inline:Nn \l__stex_variables_args_str {
3386     \token_case_meaning:NnF ##1 {
3387       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3388       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3389       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3390       {\tl_to_str:n a} {
3391         \bool_set_false:N \l_tmpa_bool
3392         \int_incr:N \l_tmpb_int
3393       }
3394       {\tl_to_str:n B} {
3395         \bool_set_false:N \l_tmpa_bool
3396         \int_incr:N \l_tmpb_int
3397       }
3398     }{
3399       \msg_error:nxxx{stex}{error/wrongargs}{
3400         variable~\l__stex_variables_name_str
3401       }{##1}
3402     }
3403   }
3404   \bool_if:NTF \l_tmpa_bool {
3405     % possibly numeric
3406     \str_if_empty:NTF \l__stex_variables_args_str {
3407       \prop_put:Nnn \l_tmpa_prop { args } {}
3408       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3409     }{
3410       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3411       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3412       \str_clear:N \l_tmpa_str
3413       \int_step_inline:nn \l_tmpa_int {

```

```

3414     \str_put_right:Nn \l_tmpa_str i
3415   }
3416   \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3417   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3418 }
3419 } {
3420   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3421   \prop_put:Nnx \l_tmpa_prop { arity }
3422     { \str_count:N \l__stex_variables_args_str }
3423 }
3424 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3425 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
3426
3427 % argnames
3428
3429 \clist_clear:N \l_tmpa_clist
3430 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
3431   \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3432     \clist_put_right:Nn \l_tmpa_clist {##1}
3433   } {
3434     \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3435     \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
3436   }
3437 }
3438 \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3439
3440
3441 \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop } \l_tmpa_prop
3442
3443 \tl_if_empty:NF \l__stex_variables_op_tl {
3444   \cs_set:cpx {
3445     stex_var_op_notation_ \l__stex_variables_name_str _cs
3446   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3447 }
3448
3449 \tl_set:Nn \l_stex_notation_after_do_tl {
3450   \exp_args:Nne \use:nn {
3451     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3452     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3453   } {{
3454     \exp_after:wN \exp_after:wN \exp_after:wN
3455     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3456     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3457   }}
3458 \stex_if_do_html:T {
3459   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3460     \stex_annotate_invisible:nnn { precedence }
3461     { \l__stex_variables_prec_str }{}
3462   \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{\l__stex_variables_type_str}}
3463   \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }{}
3464   \stex_annotate_invisible:nnn{macroname}{#1}{\l__stex_variables_macroname_str}
3465   \tl_if_empty:NF \l__stex_variables_def_tl {
3466     \stex_annotate_invisible:nnn{definiens}{\l__stex_variables_def_str}
3467     {\l__stex_variables_def_tl}

```

```

3468     }
3469     \str_if_empty:NF \l__stex_variables_assoctype_str {
3470         \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3471     }
3472     \str_if_empty:NF \l__stex_variables_reorder_str {
3473         \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3474     }
3475     \int_zero:N \l_tmpa_int
3476     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3477     \tl_clear:N \l_tmpa_tl
3478     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3479         \int_incr:N \l_tmpa_int
3480         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3481         \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3482         \str_if_eq:VnTF \l_tmpb_str a {
3483             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3484                 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
3485                 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
3486             } }
3487         }{
3488             \str_if_eq:VnTF \l_tmpb_str B {
3489                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3490                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
3491                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
3492                 } }
3493             }{
3494                 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3495                     \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{ }
3496                 } }
3497             }
3498         }
3499     }
3500     \stex_annotate_invisible:nnn { notationcomp }{}{
3501         \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3502         $ \exp_args:Nno \use:nn { \use:c {
3503             stex_var_notation_\l__stex_variables_name_str_cs
3504         } } { \l_tmpa_tl } $
3505     }
3506     \tl_if_empty:NF \l__stex_variables_op_tl {
3507         \stex_annotate_invisible:nnn { notationopcomp }{}{
3508             $\l__stex_variables_op_tl$
3509         }
3510     }
3511 }
3512 \str_if_empty:NF \l__stex_variables_bind_str {
3513     \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3514 }
3515 }\ignorespacesandpars
3516 }
3517
3518 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3519 }
3520
3521 \cs_new:Nn \stex_reset:N {

```

```

3522 \tl_if_exist:NTF #1 {
3523   \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3524 }{
3525   \let \exp_not:N #1 \exp_not:N \undefined
3526 }
3527 }
3528
3529 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3530   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3531   \exp_args:Nnx \use:nn {
3532     % TODO
3533     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3534       #2
3535     }
3536   }{
3537     \_stex_reset:N \varnot
3538     \_stex_reset:N \vartype
3539     \_stex_reset:N \vardefi
3540   }
3541 }
3542
3543 \NewDocumentCommand \vardef { s } {
3544   \IfBooleanTF#1 {
3545     \__stex_variables_do_complex:nn
3546   }{
3547     \__stex_variables_do_simple:nnn
3548   }
3549 }
3550
3551 \NewDocumentCommand \svar { 0{} m }{
3552   \tl_if_empty:nTF {#1}{
3553     \str_set:Nn \l_tmpa_str { #2 }
3554   }{
3555     \str_set:Nn \l_tmpa_str { #1 }
3556   }
3557   \_stex_term_omv:nn {
3558     var://\l_tmpa_str
3559   }{
3560     \exp_args:Nnx \use:nn {
3561       \def\comp{\_varcomp}
3562       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3563       \comp{ #2 }
3564     }{
3565       \_stex_reset:N \comp
3566       \_stex_reset:N \STEXInternalCurrentSymbolStr
3567     }
3568   }
3569 }
3570
3571
3572
3573 \keys_define:nn { stex / varseq } {
3574   name .str_set_x:N = \l__stex_variables_name_str ,
3575   args .int_set:N = \l__stex_variables_args_int ,

```



```

3576 type      .tl_set:N      = \l__stex_variables_type_tl ,
3577 mid       .tl_set:N      = \l__stex_variables_mid_tl ,
3578 bind      .choices:nn    =
3579           {forall,exists}
3580           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3581 }
3582
3583 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3584   \str_clear:N \l__stex_variables_name_str
3585   \int_set:Nn \l__stex_variables_args_int 1
3586   \tl_clear:N \l__stex_variables_type_tl
3587   \str_clear:N \l__stex_variables_bind_str
3588
3589   \keys_set:nn { stex / varseq } { #1 }
3590 }
3591
3592 \NewDocumentCommand \varseq {m O{} m m m}{
3593   \__stex_variables_seq_args:n { #2 }
3594   \str_if_empty:NT \l__stex_variables_name_str {
3595     \str_set:Nx \l__stex_variables_name_str { #1 }
3596   }
3597   \prop_clear:N \l_tmpa_prop
3598   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3599
3600   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3601   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3602     \msg_error:nnxx{stex}{error/seqlength}
3603     {\int_use:N \l__stex_variables_args_int}
3604     {\seq_count:N \l_tmpa_seq}
3605   }
3606   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3607   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3608     \msg_error:nnxx{stex}{error/seqlength}
3609     {\int_use:N \l__stex_variables_args_int}
3610     {\seq_count:N \l_tmpb_seq}
3611   }
3612   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3613   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3614
3615   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str_cs}
3616   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3617
3618   % argnames
3619
3620   \clist_clear:N \l_tmpa_clist
3621   \int_step_inline:nn {\l__stex_variables_args_int} {
3622     \clist_put_right:Nn \l_tmpa_clist {##1}
3623   }
3624   \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3625
3626
3627
3628
3629   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}

```

```

3630 \int_step_inline:nn \l__stex_variables_args_int {
3631   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3632 }
3633 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3634 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3635 \tl_if_empty:NF \l__stex_variables_mid_tl {
3636   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3637   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3638 }
3639 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3640 \int_step_inline:nn \l__stex_variables_args_int {
3641   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3642 }
3643 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3644 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3645
3646
3647 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3648
3649 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3650
3651 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3652
3653 \int_step_inline:nn \l__stex_variables_args_int {
3654   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3655     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3656   }}
3657 }
3658
3659 \tl_set:Nx \l_tmpa_tl {
3660   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3661     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3662   }
3663 }
3664
3665 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3666
3667 \exp_args:Nno \use:nn {
3668   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3669   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3670
3671   \stex_debug:nn{sequences}{New~Sequence:~
3672     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3673     \prop_to_keyval:N \l_tmpa_prop
3674   }
3675   \prop_set_eq:cN {l_stex_symdecl_varseq://\l__stex_variables_name_str _prop}\l_tmpa_prop
3676
3677   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3678     \tl_if_empty:NF \l__stex_variables_type_tl {
3679       \stex_annotate:nnn {type}{\l__stex_variables_type_tl$}
3680     }
3681     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3682     \str_if_empty:NF \l__stex_variables_bind_str {
3683       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}

```

```

3684 }
3685 \stex_annotate:nnn{startindex}{#3$}
3686 \stex_annotate:nnn{endindex}{#4$}
3687
3688 \tl_clear:N \l_tmpa_tl
3689 \int_step_inline:nn \l__stex_variables_args_int {
3690   \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3691     \stex_annotate:nnn{argmarker}{##1}{}
3692   } }
3693 }
3694 \stex_annotate_invisible:nnn { notationcomp }{}{
3695   \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3696   $ \exp_args:Nno \use:nn { \use:c {
3697     stex_varseq_\l__stex_variables_name_str _cs
3698   } } { \l_tmpa_tl } $
3699 }
3700 \stex_annotate_invisible:nnn { notationopcomp }{}{
3701   $ \prop_item:Nn \l_tmpa_prop { notation } $
3702 }
3703
3704 }}
3705
3706 \ignorespacesandpars
3707 }
3708
3709
3710 \keys_define:nn { stex / mmtdecl } {
3711   name      .str_set_x:N = \l_stex_symdecl_name_str ,
3712   args      .str_set_x:N = \l_stex_symdecl_args_str ,
3713   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
3714   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
3715   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
3716   assoc     .choices:nn =
3717     {bin,binl,binr,pre,conj,pwconj}
3718     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
3719 }
3720
3721 \cs_new_protected:Nn \stex_mmtdecl_args:n {
3722   \str_clear:N \l_stex_symdecl_name_str
3723   \str_clear:N \l_stex_symdecl_args_str
3724   \str_clear:N \l_stex_symdecl_deprecate_str
3725   \str_clear:N \l_stex_symdecl_reorder_str
3726   \str_clear:N \l_stex_symdecl_assoctype_str
3727   \bool_set_false:N \l_stex_symdecl_local_bool
3728   \clist_clear:N \l_stex_symdecl_argnames_clist
3729
3730   \keys_set:nn { stex / symdecl } { #1 }
3731 }
3732
3733 \NewDocumentCommand \mmtdecl { s m O{} } {
3734   \stex_mmtdecl_args:n{#3}
3735   \IfBooleanTF #1 {
3736     \bool_set_false:N \l_stex_symdecl_make_macro_bool
3737   } {

```

```

3738     \bool_set_true:N \l_stex_symdecl_make_macro_bool
3739   }
3740   \str_if_empty:NT \l_stex_symdecl_name_str {
3741     \str_set:Nx \l_stex_symdecl_name_str { #1 }
3742   }
3743   %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3744   %   \stex_annotate:nnn{ OMID }{
3745   %     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3746   %   }{}
3747   %}
3748   \stex_symdecl_do:n{#2}
3749   \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3750     \stex_annotate:nnn{ OMID }{
3751       \l_stex_current_module_str ? \l_stex_symdecl_name_str
3752     }{},
3753     \stex_annotate:nnn{ OMID }{
3754       \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3755     }{}
3756   }
3757   \stex_smsmode_do:
3758 }
3759
3760 \stex_deactivate_macro:Nn \mmtdecl {mmtinterface~environments}
3761 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
3762
3763 </package>

```

## Chapter 32

# STEX -Terms Implementation

```
3764 <*package>
3765
3766 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3767
3768 <@@=stex_terms>
3769
3770 Warnings and error messages
3771 \msg_new:nnn{stex}{error/nonotation}{
3772   Symbol~#1~invoked,~but~has~no~notation#2!
3773 }
3774 \msg_new:nnn{stex}{error/notationarg}{
3775   Error~in~parsing~notation~#1
3776 }
3777 \msg_new:nnn{stex}{error/noop}{
3778   Symbol~#1~has~no~operator~notation~for~notation~#2
3779 }
3780 \msg_new:nnn{stex}{error/notallowed}{
3781   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3782 }
3783 \msg_new:nnn{stex}{error/doubleargument}{
3784   Argument~#1~of~symbol~#2~already~assigned
3785 }
3786 \msg_new:nnn{stex}{error/overarity}{
3787   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3788 }
3789
```

### 32.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3788
3789
3790 \bool_new:N \l_stex_allow_semantic_bool
3791 \bool_set_true:N \l_stex_allow_semantic_bool
3792
```

```

3793 \cs_new_protected:Nn \stex_invoke_symbol:n {
3794   \ifvmode\indent\fi
3795   \bool_if:NTF \l_stex_allow_semantic_bool {
3796     \str_if_eq:eeF {
3797       \prop_item:cn {
3798         l_stex_symdecl_#1_prop
3799       }{ deprecate }
3800     }{}{
3801       \msg_warning:nxxx{stex}{warning/deprecated}{
3802         Symbol~#1
3803       }{
3804         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3805       }
3806     }
3807     \if_mode_math:
3808       \exp_after:wN \__stex_terms_invoke_math:n
3809     \else:
3810       \exp_after:wN \__stex_terms_invoke_text:n
3811     \fi: { #1 }
3812   }{
3813     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3814   }
3815 }
3816
3817 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3818   \peek_charcode_remove:NTF ! {
3819     \__stex_terms_invoke_op_custom:nn {#1}
3820   }{
3821     \__stex_terms_invoke_custom:nn {#1}
3822   }
3823 }
3824
3825 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3826   \peek_charcode_remove:NTF ! {
3827     % operator
3828     \peek_charcode_remove:NTF * {
3829       % custom op
3830       \__stex_terms_invoke_op_custom:nn {#1}
3831     }{
3832       % op notation
3833       \peek_charcode:NTF [ {
3834         \__stex_terms_invoke_op_notation:nw {#1}
3835       }{
3836         \__stex_terms_invoke_op_notation:nw {#1}[]
3837       }
3838     }
3839   }{
3840     \peek_charcode_remove:NTF * {
3841       \__stex_terms_invoke_custom:nn {#1}
3842       % custom
3843     }{
3844       % normal
3845       \peek_charcode:NTF [ {
3846         \__stex_terms_invoke_notation:nw {#1}

```

```

3847     }{
3848       \_stex_terms_invoke_notation:nw {#1}[]
3849     }
3850   }
3851 }
3852 }
3853
3854
3855 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3856   \exp_args:Nnx \use:nn {
3857     \def\comp{\_comp}
3858     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3859     \bool_set_false:N \l_stex_allow_semantic_bool
3860     \stex_mathml_intent:nn{#1}{
3861       \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3862         \comp{ #2 }
3863       }
3864     }
3865   }{
3866     \_stex_reset:N \comp
3867     \_stex_reset:N \STEXInternalCurrentSymbolStr
3868     \bool_set_true:N \l_stex_allow_semantic_bool
3869   }
3870 }
3871
3872 \keys_define:nn { stex / terms } {
3873   % lang      .tl_set_x:N = \l_stex_notation_lang_str ,
3874   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3875   unknown .code:n      = \str_set:Nx
3876     \l_stex_notation_variant_str \l_keys_key_str
3877 }
3878
3879 \cs_new_protected:Nn \_stex_terms_args:n {
3880   % \str_clear:N \l_stex_notation_lang_str
3881   \str_clear:N \l_stex_notation_variant_str
3882
3883   \keys_set:nn { stex / terms } { #1 }
3884 }
3885
3886 \cs_new_protected:Nn \stex_find_notation:nn {
3887   \_stex_terms_args:n { #2 }
3888   \seq_if_empty:cTF {
3889     \l_stex_symdecl_ #1 _notations
3890   } {
3891     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3892   } {
3893     \str_if_empty:NTF \l_stex_notation_variant_str {
3894       \seq_get_left:cN { \l_stex_symdecl_ #1 _notations } \l_stex_notation_variant_str
3895     } {
3896       \seq_if_in:cxTF { \l_stex_symdecl_ #1 _notations } {
3897         \l_stex_notation_variant_str
3898       } {
3899         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3900       } {

```

```

3901         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3902         ~\l_stex_notation_variant_str
3903     }
3904 }
3905 }
3906 }
3907 }
3908
3909 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3910     \exp_args:Nnx \use:nn {
3911         \def\comp{\_comp}
3912         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3913         \stex_find_notation:nn { #1 }{ #2 }
3914         \bool_set_false:N \l_stex_allow_semantic_bool
3915         \cs_if_exist:cTF {
3916             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3917         }{
3918             \_stex_term_oms:nnn { #1 }{
3919                 #1 \c_hash_str \l_stex_notation_variant_str
3920             }{
3921                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3922             }
3923         }{
3924             \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3925                 \cs_if_exist:cTF {
3926                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3927                 }{
3928                     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3929                         \_stex_reset:N \comp
3930                         \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3931                         \_stex_reset:N \STEXInternalCurrentSymbolStr
3932                         \bool_set_true:N \l_stex_allow_semantic_bool
3933                     }
3934                     \def\comp{\_comp}
3935                     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3936                     \bool_set_false:N \l_stex_allow_semantic_bool
3937                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3938                 }{
3939                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3940                     ~\l_stex_notation_variant_str
3941                     }
3942                 }
3943             }{
3944                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3945             }
3946         }
3947     }{
3948         \_stex_reset:N \comp
3949         \_stex_reset:N \STEXInternalCurrentSymbolStr
3950         \bool_set_true:N \l_stex_allow_semantic_bool
3951     }
3952 }
3953
3954 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```



```

3955 \stex_find_notation:nn { #1 }{ #2 }
3956 \cs_if_exist:cTF {
3957   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3958 }{
3959   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3960     \_stex_reset:N \comp
3961     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3962     \_stex_reset:N \STEXInternalCurrentSymbolStr
3963     \bool_set_true:N \l_stex_allow_semantic_bool
3964   }
3965   \def\comp{\_comp}
3966   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3967   \bool_set_false:N \l_stex_allow_semantic_bool
3968   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3969 }{
3970   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3971     ~\l_stex_notation_variant_str
3972   }
3973 }
3974 }
3975
3976 \prop_new:N \l__stex_terms_custom_args_prop
3977 \clist_new:N \l_stex_argnames_seq
3978 \seq_new:N \l__stex_terms_tmp_seq
3979
3980 \cs_new_protected:Nn \__stex_terms_custom_comp:n{ \bool_set_false:N \l_stex_allow_semantic_bo
3981
3982 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3983   \exp_args:Nnx \use:nn {
3984     \def\comp{\__stex_terms_custom_comp:n}
3985     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3986     \prop_clear:N \l__stex_terms_custom_args_prop
3987     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3988     \prop_get:cnN {
3989       l_stex_symdecl_#1 _prop
3990     }{ args } \l_tmpa_str
3991     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
3992       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
3993     }
3994     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3995     \tl_set:Nn \arg { \__stex_terms_arg: }
3996     \str_if_empty:NTF \l_tmpa_str {
3997       \stex_mathml_intent:nn{#1}{
3998         \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3999       }
4000     }{
4001       \seq_clear:N \l__stex_terms_tmp_seq
4002       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4003         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4004         \bool_lazy_or:nnT{
4005           \str_if_eq_p:nn{a}{\str_item:Nn \l_tmpa_str{##1}}
4006         }{
4007           \str_if_eq_p:nn{B}{\str_item:Nn \l_tmpa_str{##1}}
4008         }{

```

```

4009         \tl_put_right:Nn \l__stex_terms_tmp_tl +
4010     }
4011     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4012 }
4013 \stex_mathml_intent:nn{
4014     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}]{ args }(
4015         \seq_use:Nn \l__stex_terms_tmp_seq ,
4016     )
4017 }{
4018     \str_if_in:NnTF \l_tmpa_str b {
4019         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4020     }{
4021         \str_if_in:NnTF \l_tmpa_str B {
4022             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4023         }{
4024             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4025         }
4026     }
4027 }
4028 }
4029 % TODO check that all arguments exist
4030 }{
4031     \_stex_reset:N \l_stex_argnames_seq
4032     \_stex_reset:N \STEXInternalCurrentSymbolStr
4033     \_stex_reset:N \arg
4034     \_stex_reset:N \comp
4035     \_stex_reset:N \l__stex_terms_custom_args_prop
4036     %\bool_set_true:N \l_stex_allow_semantic_bool
4037 }
4038 }
4039
4040 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
4041     \tl_if_empty:nTF {#2}{
4042         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
4043         \bool_set_true:N \l_tmpa_bool
4044         \bool_do_while:Nn \l_tmpa_bool {
4045             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
4046                 \int_incr:N \l_tmpa_int
4047         }{
4048             \bool_set_false:N \l_tmpa_bool
4049         }
4050     }
4051     }{
4052         \int_set:Nn \l_tmpa_int { #2 }
4053     }
4054     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
4055     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
4056         \msg_error:nnxxx{stex}{error/overarity}
4057         {\int_use:N \l_tmpa_int}
4058         {\STEXInternalCurrentSymbolStr}
4059         {\str_count:N \l_tmpa_str}
4060     }
4061     \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
4062     \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {

```

```

4063 \bool_lazy_any:nF {
4064   {\str_if_eq_p:Vn \l_tmpa_str {a}}
4065   {\str_if_eq_p:Vn \l_tmpa_str {B}}
4066 }{
4067   \msg_error:nnxx{stex}{error/doubleargument}
4068   {\int_use:N \l_tmpa_int}
4069   {\STEXInternalCurrentSymbolStr}
4070 }
4071 }
4072 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
4073 \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
4074   \bool_set_true:N \l_stex_allow_semantic_bool
4075   \use:nn
4076 }
4077 {
4078 \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4079   \IfBooleanTF#1{
4080     \stex_annotate_invisible:n { %TODO
4081       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4082     }
4083   }{ %TODO
4084     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4085   }
4086 }}
4087 {\bool_set_false:N \l_stex_allow_semantic_bool}
4088 }
4089
4090
4091 \cs_new_protected:Nn \_stex_term_arg:nn {
4092   \bool_set_true:N \l_stex_allow_semantic_bool
4093   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
4094   \bool_set_false:N \l_stex_allow_semantic_bool
4095 }
4096
4097 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
4098   \exp_args:Nnx \use:nn
4099   { \int_set:Nn \l__stex_terms_downprec { #2 }
4100     \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4101       \_stex_term_arg:nn { #1 }{ #3 }
4102     }
4103   }
4104   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4105 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 92.)

`\STEXInternalTermMathAssocArgiiii`

```

4106 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4#5 {
4107   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
4108   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
4109   \tl_if_empty:nTF { #3 }{
4110     \STEXInternalTermMathArgiii{#5#1}{#2}{}
4111   }{
4112     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{

```

```

4113     \expandafter\if\expandafter\relax\noexpand#3
4114     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
4115   \else
4116     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
4117   \fi
4118   \l_tmpa_tl
4119 }{
4120   \__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
4121 }
4122 }
4123 }
4124
4125 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nnn {
4126   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
4127   \str_if_empty:NTF \l_tmpa_str {
4128     \exp_args:Nx \cs_if_eq:NNTF {
4129       \tl_head:N #1
4130     } \stex_invoke_sequence:n {
4131       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
4132       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
4133       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
4134       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
4135       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
4136         \exp_not:n{\exp_args:Nnx \use:nn} {
4137           \exp_not:n {
4138             \def\comp{\_varcomp}
4139             \str_set:Nn \STEXInternalCurrentSymbolStr
4140             } {varseq://\l_tmpa_str}
4141             \exp_not:n{ ##1 }
4142           }{
4143             \exp_not:n {
4144               \_stex_reset:N \comp
4145               \_stex_reset:N \STEXInternalCurrentSymbolStr
4146             }
4147           }
4148         }}}
4149       \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4150       \seq_reverse:N \l_tmpa_seq
4151       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4152       \seq_map_inline:Nn \l_tmpa_seq {
4153         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4154           \exp_args:Nno
4155           \l_tmpa_cs { ##1 } \l_tmpa_tl
4156         }
4157       }
4158       \tl_set:Nx \l_tmpa_tl {
4159         \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4160           \exp_args:No \exp_not:n \l_tmpa_tl
4161         }
4162       }
4163       \exp_args:No\l_tmpb_tl\l_tmpa_tl
4164     }{
4165       \__stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4166     }

```

```

4167 } {
4168   \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4169 }
4170
4171 }
4172
4173 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nnn {
4174   \clist_set:Nn \l_tmpa_clist{ #2 }
4175   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4176     \tl_set:Nn \l_tmpa_tl {
4177       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4178         \_stex_term_arg:nn{A#3#1}{ #2 } }
4179     }
4180   }{
4181     \clist_reverse:N \l_tmpa_clist
4182     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4183     \tl_set:Nx \l_tmpa_tl {
4184       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4185         \_stex_term_arg:nn{A#3#1}{
4186           \exp_args:No \exp_not:n \l_tmpa_tl
4187         }
4188       }
4189     }
4190     \clist_map_inline:Nn \l_tmpa_clist {
4191       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4192         \l_tmpa_cs {
4193           \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4194             \_stex_term_arg:nn{A#3#1}{##1}
4195           }
4196         } \l_tmpa_tl
4197       }
4198     }
4199   }
4200   \exp_args:No\l_tmpb_tl\l_tmpa_tl
4201 }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 93.)

## 32.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
4202 \tl_const:Nx \infprec {\int_use:N \c_max_int}
4203 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4204 \int_new:N \l__stex_terms_downprec
4205 \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
4206 \tl_set:Nn \l__stex_terms_left_bracket_str (
4207 \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

```

4208 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4209   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4210     \bool_set_false:N \l__stex_terms_brackets_done_bool
4211     #2
4212   } {
4213     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4214       \bool_if:NTF \l__stex_inarray_bool { #2 }{
4215         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4216         \dobrackets { #2 }
4217       }
4218     }{ #2 }
4219   }
4220 }
```

(End definition for \\_stex\_terms\_maybe\_brackets:nn.)

**\dobrackets**

```

4221 \bool_new:N \l__stex_terms_brackets_done_bool
4222 %\RequirePackage{scalerel}
4223 \cs_new_protected:Npn \dobrackets #1 {
4224   %\ThisStyle{\if D\m@switch
4225   %   \exp_args:Nnx \use:nn
4226   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4227   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
4228   %   \else
4229   \exp_args:Nnx \use:nn
4230   {
4231     \bool_set_true:N \l__stex_terms_brackets_done_bool
4232     \int_set:Nn \l__stex_terms_downprec \infprec
4233     \l__stex_terms_left_bracket_str
4234     #1
4235   }
4236   {
4237     \bool_set_false:N \l__stex_terms_brackets_done_bool
4238     \l__stex_terms_right_bracket_str
4239     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4240   }
4241   %\fi}
4242 }
```

(End definition for \dobrackets. This function is documented on page 93.)

**\withbrackets**

```

4243 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4244   \exp_args:Nnx \use:nn
4245   {
4246     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4247     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4248     #3
4249   }
4250 }
```

```

4251 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4252 {\l__stex_terms_left_bracket_str}
4253 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4254 {\l__stex_terms_right_bracket_str}
4255 }
4256 }

```

(End definition for `\withbrackets`. This function is documented on page 93.)

## `\STEXinvisible`

```

4257 \cs_new_protected:Npn \STEXinvisible #1 {
4258 \stex_annotate_invisible:n { #1 }
4259 }

```

(End definition for `\STEXinvisible`. This function is documented on page 93.)

OMDoc terms:

## `\STEXInternalTermMathOMSiiii`

```

4260 \cs_new_protected:Nn \_stex_term_oms:nnn {
4261 \stex_annotate:nnn{ OMID }{ #2 }{
4262 #3
4263 }
4264 }
4265
4266 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4267 \__stex_terms_maybe_brackets:nn { #3 }{
4268 \stex_mathml_intent:nn{#1} {
4269 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4270 }
4271 }
4272 }

```

(End definition for `\STEXInternalTermMathOMSiiii`. This function is documented on page 92.)

## `\_stex_term_math_omv:nn`

```

4273 \cs_new_protected:Nn \_stex_term_omv:nn {
4274 \stex_annotate:nnn{ OMV }{ #1 }{
4275 #2
4276 }
4277 }

```

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

## `\STEXInternalTermMathOMAiiai`

```

4278 \cs_new_protected:Nn \_stex_term_oma:nnn {
4279 \stex_annotate:nnn{ OMA }{ #2 }{
4280 #3
4281 }
4282 }
4283
4284 \cs_new_protected:Npn \STEXInternalTermMathOMAiiai #1#2#3#4 {
4285 \exp_args:Nnx \use:nn {
4286 \seq_clear:N \l__stex_terms_tmp_seq
4287 \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4288 \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {

```

```

4289     \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4290   }
4291   \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4292     \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4293     \bool_lazy_or:nnT{
4294       \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4295     }{
4296       \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4297     }{
4298       \tl_put_right:Nn \l__stex_terms_tmp_tl +
4299     }
4300     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4301   }
4302 }
4303 \__stex_terms_maybe_brackets:nn { #3 }{
4304   \stex_mathml_intent:nn{
4305     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4306       \seq_use:Nn \l__stex_terms_tmp_seq ,
4307     )
4308   }{
4309     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4310   }
4311 }
4312 }{
4313   \_stex_reset:N \l_stex_argnames_seq
4314 }
4315 }

```

(End definition for `\STEXInternalTermMathOMAi`. This function is documented on page 92.)

`\STEXInternalTermMathOMBiiii`

```

4316 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4317   \stex_annotate:nnn{ OMBIND }{ #2 }{
4318     #3
4319   }
4320 }
4321
4322 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
4323   \exp_args:Nnx \use:nn {
4324     \seq_clear:N \l__stex_terms_tmp_seq
4325     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4326       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4327         \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4328       }
4329       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4330         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4331         \bool_lazy_or:nnT{
4332           \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4333         }{
4334           \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4335         }{
4336           \tl_put_right:Nn \l__stex_terms_tmp_tl +
4337         }
4338         \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl

```



```

4339   }
4340 }
4341 \__stex_terms_maybe_brackets:nn { #3 }{
4342   \stex_mathml_intent:nn{
4343     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4344       \seq_use:Nn \l__stex_terms_tmp_seq ,
4345     )
4346   }{
4347     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4348   }
4349 }
4350 }{
4351   \stex_reset:N \l_stex_argnames_seq
4352 }
4353 }

```

(End definition for `\STEXInternalTermMathOMBiiii`. This function is documented on page 92.)

`\symref`  
`\symname`

```

4354 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4355
4356 \keys_define:nn { stex / symname } {
4357   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
4358   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
4359   root     .tl_set_x:N      = \l__stex_terms_root_tl
4360 }
4361
4362 \cs_new_protected:Nn \stex_symname_args:n {
4363   \tl_clear:N \l__stex_terms_post_tl
4364   \tl_clear:N \l__stex_terms_pre_tl
4365   \tl_clear:N \l__stex_terms_root_str
4366   \keys_set:nn { stex / symname } { #1 }
4367 }
4368
4369 \NewDocumentCommand \symref { m m }{
4370   \let\compemph_uri_prev:\compemph@uri
4371   \let\compemph@uri\symrefemph@uri
4372   \STEXsymbol{#1}!\{ #2 }
4373   \let\compemph@uri\compemph_uri_prev:
4374 }
4375
4376 \NewDocumentCommand \synonym { 0{} m m }{
4377   \stex_symname_args:n { #1 }
4378   \let\compemph_uri_prev:\compemph@uri
4379   \let\compemph@uri\symrefemph@uri
4380   % TODO
4381   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4382   \let\compemph@uri\compemph_uri_prev:
4383 }
4384
4385 \NewDocumentCommand \symname { 0{} m }{
4386   \stex_symname_args:n { #1 }
4387   \stex_get_symbol:n { #2 }
4388   \str_set:Nx \l_tmpa_str {

```

```

4389   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4390 }
4391 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4392
4393 \let\compemph_uri_prev:\compemph@uri
4394 \let\compemph@uri\symrefemph@uri
4395 \exp_args:Nnx \use:nn
4396 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4397   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4398 } }
4399 \let\compemph@uri\compemph_uri_prev:
4400 }
4401
4402 \NewDocumentCommand \Symname { 0{} m }{
4403   \stex_symname_args:n { #1 }
4404   \stex_get_symbol:n { #2 }
4405   \str_set:Nx \l_tmpa_str {
4406     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4407   }
4408   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4409   \let\compemph_uri_prev:\compemph@uri
4410   \let\compemph@uri\symrefemph@uri
4411   \exp_args:Nnx \use:nn
4412   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4413     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4414     \l__stex_terms_post_tl
4415   } }
4416   \let\compemph@uri\compemph_uri_prev:
4417 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 92.)

## 32.3 Notation Components

```

4418 <@@=stex_notationcomps>

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

4419 \cs_new_protected:Npn \_comp #1 {
4420   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4421     \stex_html_backend:TF {
4422       \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4423     }{
4424       \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4425     }
4426   }
4427 }
4428
4429 \cs_new_protected:Npn \_varcomp #1 {
4430   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4431     \stex_html_backend:TF {
4432       \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4433     }{
4434       \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
4435     }

```

```

4436 }
4437 }
4438
4439 \def\comp{\_comp}
4440
4441 \cs_new_protected:Npn \compemph@uri #1 #2 {
4442   \compemph{ #1 }
4443 }
4444
4445
4446 \cs_new_protected:Npn \compemph #1 {
4447   #1
4448 }
4449
4450 \cs_new_protected:Npn \defemph@uri #1 #2 {
4451   \defemph{#1}
4452 }
4453
4454 \cs_new_protected:Npn \defemph #1 {
4455   \textbf{#1}
4456 }
4457
4458 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4459   \symrefemph{#1}
4460 }
4461
4462 \cs_new_protected:Npn \symrefemph #1 {
4463   \emph{#1}
4464 }
4465
4466 \cs_new_protected:Npn \varemp@uri #1 #2 {
4467   \varemp{#1}
4468 }
4469
4470 \cs_new_protected:Npn \varemp #1 {
4471   #1
4472 }

```

(End definition for `\comp` and others. These functions are documented on page 93.)

## `\ellipses`

```

4473 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 93.)

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
4474 \bool_new:N \l_stex_inparray_bool
4475 \bool_set_false:N \l_stex_inparray_bool
4476 \NewDocumentCommand \parray { m m } {
4477   \begingroup
4478   \bool_set_true:N \l_stex_inparray_bool
4479   \begin{array}{#1}
4480     #2
4481   \end{array}
4482 \endgroup

```

```

4483 }
4484
4485 \NewDocumentCommand \prmatrix { m } {
4486   \begingroup
4487   \bool_set_true:N \l_stex_inarray_bool
4488   \begin{matrix}
4489     #1
4490   \end{matrix}
4491   \endgroup
4492 }
4493
4494 \def \maybepline {
4495   \bool_if:NT \l_stex_inarray_bool {\hline}
4496 }
4497
4498 \def \parrayline #1 #2 {
4499   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
4500 }
4501
4502 \def \pmrow #1 { \parrayline{}{ #1 } }
4503
4504 \def \parraylineh #1 #2 {
4505   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
4506 }
4507
4508 \def \parraycell #1 {
4509   #1 \bool_if:NT \l_stex_inarray_bool {&}
4510 }

```

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

## 32.4 Variables

```

4511 <@@=stex_variables>

```

\stex\_invoke\_variable:n Invokes a variable

```

4512 \cs_new_protected:Nn \stex_invoke_variable:n {
4513   \if_mode_math:
4514     \exp_after:wN \__stex_variables_invoke_math:n
4515   \else:
4516     \exp_after:wN \__stex_variables_invoke_text:n
4517   \fi: {#1}
4518 }
4519
4520 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4521   \peek_charcode_remove:NTF ! {
4522     \__stex_variables_invoke_op_custom:nn {#1}
4523   }{
4524     \__stex_variables_invoke_custom:nn {#1}
4525   }
4526 }
4527
4528
4529 \cs_new_protected:Nn \__stex_variables_invoke_math:n {

```

```

4530 \peek_charcode_remove:NTF ! {
4531   \peek_charcode_remove:NTF ! {
4532     \peek_charcode:NTF [ {
4533       % TODO throw error
4534     }{
4535       \__stex_variables_invoke_op_custom:nn
4536     }
4537   }{
4538     \__stex_variables_invoke_op:n { #1 }
4539   }
4540 }{
4541   \peek_charcode_remove:NTF * {
4542     \__stex_variables_invoke_custom:nn { #1 }
4543   }{
4544     \__stex_variables_invoke_math_ii:n { #1 }
4545   }
4546 }
4547 }
4548
4549 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4550   \exp_args:Nnx \use:nn {
4551     \def\comp{\_varcomp}
4552     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4553     \bool_set_false:N \l_stex_allow_semantic_bool
4554     \stex_term_omv:nn {var://#1}{
4555       \comp{ #2 }
4556     }
4557   }{
4558     \stex_reset:N \comp
4559     \stex_reset:N \STEXInternalCurrentSymbolStr
4560     \bool_set_true:N \l_stex_allow_semantic_bool
4561   }
4562 }
4563
4564 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4565   \cs_if_exist:cTF {
4566     stex_var_op_notation_ #1 _cs
4567   }{
4568     \exp_args:Nnx \use:nn {
4569       \def\comp{\_varcomp}
4570       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4571       \stex_term_omv:nn { var://#1 }{
4572         \use:c{stex_var_op_notation_ #1 _cs }
4573       }
4574     }{
4575       \stex_reset:N \comp
4576       \stex_reset:N \STEXInternalCurrentSymbolStr
4577     }
4578   }{
4579     \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4580       \__stex_variables_invoke_math_ii:n {#1}
4581     }{
4582       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
4583     }

```

```

4584 }
4585 }
4586
4587 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4588   \cs_if_exist:cTF {
4589     stex_var_notation_#1_cs
4590   }{
4591     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4592       \_stex_reset:N \comp
4593       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4594       \_stex_reset:N \STEXInternalCurrentSymbolStr
4595       \bool_set_true:N \l_stex_allow_semantic_bool
4596     }
4597     \def\comp{\_varcomp}
4598     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4599     \bool_set_false:N \l_stex_allow_semantic_bool
4600     \use:c{stex_var_notation_#1_cs}
4601   }{
4602     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
4603   }
4604 }
4605
4606 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4607   \exp_args:Nnx \use:nn {
4608     \def\comp{\_varcomp}
4609     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4610     \prop_clear:N \l__stex_terms_custom_args_prop
4611     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4612     \prop_get:cnN {
4613       l_stex_symdecl_var://#1 _prop
4614     }{ args } \l_tmpa_str
4615     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4616     \tl_set:Nn \arg { \_stex_terms_arg: }
4617     \str_if_empty:NTF \l_tmpa_str {
4618       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4619     }{
4620       \str_if_in:NnTF \l_tmpa_str b {
4621         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4622       }{
4623         \str_if_in:NnTF \l_tmpa_str B {
4624           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4625         }{
4626           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4627         }
4628       }
4629     }
4630     % TODO check that all arguments exist
4631   }{
4632     \_stex_reset:N \STEXInternalCurrentSymbolStr
4633     \_stex_reset:N \arg
4634     \_stex_reset:N \comp
4635     \_stex_reset:N \l__stex_terms_custom_args_prop
4636     %\bool_set_true:N \l_stex_allow_semantic_bool
4637   }

```

```
4638 }
```

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

## 32.5 Sequences

```
4639 <@@=stex_sequences>
4640
4641 \cs_new_protected:Nn \stex_invoke_sequence:n {
4642   \peek_charcode_remove:NTF ! {
4643     \stex_term_omv:nn {varseq://#1}{
4644       \exp_args:Nnx \use:nn {
4645         \def\comp{\_varcomp}
4646         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4647         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4648       }{
4649         \_stex_reset:N \comp
4650         \_stex_reset:N \STEXInternalCurrentSymbolStr
4651       }
4652     }
4653   }{
4654     \bool_set_false:N \l_stex_allow_semantic_bool
4655     \def\comp{\_varcomp}
4656     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4657     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4658       \_stex_reset:N \comp
4659       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4660       \_stex_reset:N \STEXInternalCurrentSymbolStr
4661       \bool_set_true:N \l_stex_allow_semantic_bool
4662     }
4663     \use:c { stex_varseq_#1_cs }
4664   }
4665 }
4666 </package>
```

## Chapter 33

# STEX -Structural Features Implementation

```
4667 ⟨*package⟩
4668
4669 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4670
4671 Warnings and error messages
4672 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4673   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4674 }
4675 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4676   Symbol~#1~not~assigned~in~interpretmodule~#2
4677 }
4678 \msg_new:nnn{stex}{error/unknownstructure}{
4679   No~structure~#1~found!
4680 }
4681
4682 \msg_new:nnn{stex}{error/unknownfield}{
4683   No~field~#1~in~instance~#2~found!\#3
4684 }
4685
4686 \msg_new:nnn{stex}{error/keyval}{
4687   Invalid~key=value~pair~#1
4688 }
4689 \msg_new:nnn{stex}{error/instantiate/missing}{
4690   Assignments~missing~in~instantiate:~#1
4691 }
4692 \msg_new:nnn{stex}{error/incompatible}{
4693   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4694 }
4695
```



## 33.1 Imports with modification

```

4696 <@@=stex_copymodule>
4697 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4698   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4699     \tl_set:Nn \l_tmpa_tl { #1 }
4700     \__stex_copymodule_get_symbol_from_cs:
4701   }{
4702     % argument is a string
4703     % is it a command name?
4704     \cs_if_exist:cTF { #1 }{
4705       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4706       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4707       \str_if_empty:NTF \l_tmpa_str {
4708         \exp_args:Nx \cs_if_eq:NNTF {
4709           \tl_head:N \l_tmpa_tl
4710         } \stex_invoke_symbol:n {
4711           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4712         }{
4713           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4714         }
4715       } {
4716         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4717       }
4718     }{
4719       % argument is not a command name
4720       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4721       % \l_stex_all_symbols_seq
4722     }
4723   }
4724 }
4725
4726 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4727   \str_set:Nn \l_tmpa_str { #1 }
4728   \bool_set_false:N \l_tmpa_bool
4729   \bool_if:NF \l_tmpa_bool {
4730     \tl_set:Nn \l_tmpa_tl {
4731       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4732     }
4733     \str_set:Nn \l_tmpa_str { #1 }
4734     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4735     \seq_map_inline:Nn #2 {
4736       \str_set:Nn \l_tmpb_str { ##1 }
4737       \str_if_eq:eeT { \l_tmpa_str } {
4738         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4739       } {
4740         \seq_map_break:n {
4741           \tl_set:Nn \l_tmpa_tl {
4742             \str_set:Nn \l_stex_get_symbol_uri_str {
4743               ##1
4744             }
4745           }
4746         }
4747       }

```

```

4748     }
4749     \l_tmpa_tl
4750   }
4751 }
4752
4753 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4754   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4755     { \tl_tail:N \l_tmpa_tl }
4756   \tl_if_single:NTF \l_tmpa_tl {
4757     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4758       \exp_after:wN \str_set:Nn \exp_after:wN
4759         \l_stex_get_symbol_uri_str \l_tmpa_tl
4760       \__stex_copymodule_get_symbol_check:n { #1 }
4761     }{
4762       % TODO
4763       % tail is not a single group
4764     }
4765   }{
4766     % TODO
4767     % tail is not a single group
4768   }
4769 }
4770
4771 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4772   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4773     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4774       :~\seq_use:Nn #1 {,~}
4775     }
4776   }
4777 }
4778
4779 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4780   % import module
4781   \stex_import_module_uri:nn { #1 } { #2 }
4782   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4783   \stex_import_require_module:nnnn
4784     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4785     { \l_stex_import_path_str } { \l_stex_import_name_str }
4786
4787   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4788   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4789
4790   % fields
4791   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4792   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4793     \seq_map_inline:cn {c_stex_module_##1_constants}{
4794       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4795         ##1 ? #####1
4796       }
4797     }
4798   }
4799
4800   % setup prop
4801   \seq_clear:N \l_tmpa_seq

```

```

4802 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4803   name      = \l_stex_current_copymodule_name_str ,
4804   module    = \l_stex_current_module_str ,
4805   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4806   includes  = \l_tmpa_seq %,
4807 % fields    = \l_tmpa_seq
4808 }
4809 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4810   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4811 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4812 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4813
4814 \stex_if_do_html:T {
4815   \begin{stex_annotate_env} {#4} {
4816     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4817   }
4818   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
4819 }
4820 }
4821
4822 \cs_new_protected:Nn \stex_copymodule_end:n {
4823   % apply to every field
4824   \def \l_tmpa_cs ##1 ##2 {#1}
4825
4826   \tl_clear:N \__stex_copymodule_module_tl
4827   \tl_clear:N \__stex_copymodule_exec_tl
4828
4829   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4830   \seq_clear:N \__stex_copymodule_fields_seq
4831
4832   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4833     \seq_map_inline:cn {c_stex_module_##1_constants}{
4834
4835       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4836       \l_tmpa_cs{##1}{####1}
4837
4838       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4839         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4840         \stex_if_do_html:T {
4841           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4842             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4843           }
4844         }
4845       }{
4846         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4847       }
4848
4849       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4850       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4851       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4852
4853       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4854         \stex_if_do_html:T {
4855           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4856         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4857     }
4858 }
4859 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4860 }
4861
4862 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4863 \tl_put_right:Nx \__stex_copymodule_module_tl {
4864     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4865     \prop_set_from_keyval:cn {
4866         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4867     }{
4868         \prop_to_keyval:N \l_tmpa_prop
4869     }
4870 }
4871
4872 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4873     \stex_if_do_html:T {
4874         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4875             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4876         }
4877     }
4878     \tl_put_right:Nx \__stex_copymodule_module_tl {
4879         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4880             \stex_invoke_symbol:n {
4881                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4882             }
4883         }
4884     }
4885 }
4886
4887 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4888
4889 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4890     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4891 }
4892
4893 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4894     \stex_if_do_html:TF{
4895         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4896     }{
4897         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4898     }
4899 }
4900 }
4901 }
4902
4903
4904 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4905 \tl_put_left:Nx \__stex_copymodule_module_tl {
4906     \prop_set_from_keyval:cn {
4907         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4908     }{
4909         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4910     }
4911   }
4912
4913   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4914     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4915   }
4916
4917   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4918   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4919   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4920
4921   \__stex_copymodule_exec_tl
4922   \stex_if_do_html:T {
4923     \end{stex_annotate_env}
4924   }
4925 }
4926
4927 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4928   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4929   \stex_deactivate_macro:Nn \symdecl {module~environments}
4930   \stex_deactivate_macro:Nn \symdef {module~environments}
4931   \stex_deactivate_macro:Nn \notation {module~environments}
4932   \stex_reactivate_macro:N \assign
4933   \stex_reactivate_macro:N \renamedekl
4934   \stex_reactivate_macro:N \donotcopy
4935   \stex_smsmode_do:
4936 }{
4937   \stex_copymodule_end:n {}
4938 }
4939
4940 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4941   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4942   \stex_deactivate_macro:Nn \symdecl {module~environments}
4943   \stex_deactivate_macro:Nn \symdef {module~environments}
4944   \stex_deactivate_macro:Nn \notation {module~environments}
4945   \stex_reactivate_macro:N \assign
4946   \stex_reactivate_macro:N \renamedekl
4947   \stex_reactivate_macro:N \donotcopy
4948   \stex_smsmode_do:
4949 }{
4950   \stex_copymodule_end:n {
4951     \tl_if_exist:cF {
4952       l__stex_copymodule_copymodule_##1?##2_def_tl
4953     }{
4954       \str_if_eq:eeF {
4955         \prop_item:cn{
4956           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4957       }{ true }{
4958         \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4959           ##1?##2
4960         }{\l_stex_current_copymodule_name_str}
4961       }
4962     }
4963   }

```

```

4964 }
4965
4966 \iffalse \begin{stex_annotate_env} \fi
4967 \NewDocumentEnvironment {realization} { 0{} m}{
4968   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #2 }{ realize }
4969   \stex_deactivate_macro:Nn \symdecl {module~environments}
4970   \stex_deactivate_macro:Nn \symdef {module~environments}
4971   \stex_deactivate_macro:Nn \notation {module~environments}
4972   \stex_reactivate_macro:N \donotcopy
4973   \stex_reactivate_macro:N \assign
4974   \stex_smsmode_do:
4975 }{
4976   \stex_import_module_uri:nn { #1 } { #2 }
4977   \tl_clear:N \__stex_copymodule_exec_tl
4978   \tl_set:Nx \__stex_copymodule_module_tl {
4979     \stex_import_require_module:nnnn
4980     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4981     { \l_stex_import_path_str } { \l_stex_import_name_str }
4982   }
4983   \exp_args:Nx \stex_add_import_to_current_module:n{
4984     \l_stex_import_ns_str ? \l_stex_import_name_str
4985   }
4986
4987   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4988     \seq_map_inline:cn {c_stex_module_###1_constants}{
4989       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #
4990       \tl_if_exist:cT {l__stex_copymodule_copymodule_###1?###1_def_tl}{
4991         \stex_if_do_html:T {
4992           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4993             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4994               $\stex_annotate_invisible:nnn{definiens}{}{\exp_after:wN \exp_not:N\csname l__
4995             }
4996           }
4997         }
4998         \tl_put_right:Nx \__stex_copymodule_module_tl {
4999           \prop_put:cnn {l_stex_symdecl_###1?###1_prop}{ defined }{ true }
5000         }
5001       }
5002     }}
5003
5004   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
5005
5006   \__stex_copymodule_exec_tl
5007   \stex_if_do_html:T {\end{stex_annotate_env}}
5008 }
5009
5010 \NewDocumentCommand \donotcopy { m }{
5011   \str_clear:N \l_stex_import_name_str
5012   \str_set:Nn \l_tmpa_str { #1 }
5013   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5014   \seq_map_inline:Nn \l_stex_all_modules_seq {
5015     \str_set:Nn \l_tmpb_str { ##1 }
5016     \str_if_eq:eeT { \l_tmpa_str } {
5017       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }

```

```

5018 } {
5019   \seq_map_break:n {
5020     \stex_if_do_html:T {
5021       \stex_if_smsmode:F {
5022         \stex_annotate_invisible:nnn{donotcopy}{##1}{
5023           \stex_annotate:nnn{domain}{##1}{}}
5024       }
5025     }
5026   }
5027   \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
5028 }
5029 }
5030 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
5031   \str_set:Nn \l_tmpb_str { ###1 }
5032   \str_if_eq:eeT { \l_tmpa_str } {
5033     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5034   } {
5035     \seq_map_break:n {\seq_map_break:n {
5036       \stex_if_do_html:T {
5037         \stex_if_smsmode:F {
5038           \stex_annotate_invisible:nnn{donotcopy}{####1}{
5039             \stex_annotate:nnn{domain}{
5040               \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
5041             }{}}
5042         }
5043       }
5044     }
5045     \str_set:Nx \l_stex_import_name_str {
5046       \prop_item:cn {l_stex_copymodule_ ####1 _prop}{module}
5047     }
5048   }}
5049 }
5050 }
5051 }
5052 \str_if_empty:NTF \l_stex_import_name_str {
5053   % TODO throw error
5054 }{
5055   \stex_collect_imports:n {\l_stex_import_name_str }
5056   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5057     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
5058     \seq_map_inline:cn {c_stex_module_###1_constants}{
5059       \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? ####1 }
5060       \bool_lazy_any:nT {
5061         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?####1_name_str}}
5062         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?####1_macroname_str}}
5063         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?####1_def_tl}}
5064       }{
5065         % TODO throw error
5066       }
5067     }
5068   }
5069   \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
5070   \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
5071   \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq

```

```

5072 }
5073 \stex_smsmode_do:
5074 }
5075
5076 \NewDocumentCommand \assign { m m }{
5077   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
5078   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
5079   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
5080   \stex_smsmode_do:
5081 }
5082
5083 \keys_define:nn { stex / renamedec1 } {
5084   name          .str_set_x:N = \l_stex_renamedec1_name_str
5085 }
5086 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
5087   \str_clear:N \l_stex_renamedec1_name_str
5088   \keys_set:nn { stex / renamedec1 } { #1 }
5089 }
5090
5091 \NewDocumentCommand \renamedec1 { O{} m m }{
5092   \__stex_copymodule_renamedec1_args:n { #1 }
5093   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
5094   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
5095   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
5096   \str_if_empty:NTF \l_stex_renamedec1_name_str {
5097     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5098       \l_stex_get_symbol_uri_str
5099     } }
5100   } {
5101     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
5102       \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
5103       \prop_set_eq:cc {l_stex_symdecl_
5104         \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5105       _prop
5106       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
5107       \seq_set_eq:cc {l_stex_symdecl_
5108         \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5109       _notations
5110       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
5111       \prop_put:cnx {l_stex_symdecl_
5112         \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5113       _prop
5114       }{ name }{ \l_stex_renamedec1_name_str }
5115       \prop_put:cnx {l_stex_symdecl_
5116         \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5117       _prop
5118       }{ module }{ \l_stex_current_module_str }
5119       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
5120         \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5121       }
5122       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5123         \l_stex_current_module_str ? \l_stex_renamedec1_name_str
5124       } }
5125     }

```



```

5126 \stex_smsmode_do:
5127 }
5128
5129 \stex_deactivate_macro:Nn \assign {copymodules}
5130 \stex_deactivate_macro:Nn \renamedekl {copymodules}
5131 \stex_deactivate_macro:Nn \donotcopy {copymodules}
5132
5133

```

## 33.2 The feature environment

`structural@feature (env.)`

```

5134 <@@=stex_features>
5135
5136 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
5137 \stex_if_in_module:F {
5138 \msg_set:nnn{stex}{error/nomodule}{
5139 Structural~Feature~has~to~occur~in~a~module:\\
5140 Feature~#2~of~type~#1\\
5141 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
5142 }
5143 \msg_error:nn{stex}{error/nomodule}
5144 }
5145
5146 \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
5147
5148 \stex_module_setup:nn{meta=NONE}{#2 - #1}
5149
5150 \stex_if_do_html:T {
5151 \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
5152 \stex_annotate_invisible:nnn{header}{}{ #3 }
5153 }
5154 }{
5155 \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5156 \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5157 \stex_debug:nn{features}{
5158 Feature: \l_stex_last_feature_str
5159 }
5160 \stex_if_do_html:T {
5161 \end{stex_annotate_env}
5162 }
5163 }

```

## 33.3 Structure

`structure (env.)`

```

5164 <@@=stex_structures>
5165 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5166 \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
5167 \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
5168 }
5169 \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}

```

```

5170     {#1}{#2}
5171 }
5172
5173 \keys_define:nn { stex / features / structure } {
5174   name          .str_set_x:N = \l__stex_structures_name_str ,
5175 }
5176
5177 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5178   \str_clear:N \l__stex_structures_name_str
5179   \keys_set:nn { stex / features / structure } { #1 }
5180 }
5181 \NewDocumentEnvironment{mathstructure}{m O{}}{
5182   \begin{mathstructure_inner}{#1}[#2]
5183     \stex_smsmode_do:
5184     \ignorespacesandpars
5185   }\end{mathstructure_inner}}
5186 \NewDocumentEnvironment{mathstructure_inner}{m O{}}{
5187   \__stex_structures_structure_args:n { #2 }
5188   \str_if_empty:NT \l__stex_structures_name_str {
5189     \str_set:Nx \l__stex_structures_name_str { #1 }
5190   }
5191   \stex_suppress_html:n {
5192     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5193     \exp_args:Nx \stex_symdecl_do:nn {
5194       name = \l__stex_structures_name_str ,
5195       def  = {\STEXsymbol{module-type}}{
5196         \STEXInternalTermMathOMSiiii {
5197           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5198             { ns } ?
5199           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5200             { name } / \l__stex_structures_name_str - structure
5201         }{}{0}{}
5202       }}
5203     }{ #1 }
5204   }
5205   \exp_args:Nnnx
5206   \begin{structural_feature_module}{ structure }
5207     { \l__stex_structures_name_str }{}
5208 }{
5209   \end{structural_feature_module}
5210   \_stex_reset_up_to_module:n \l_stex_last_feature_str
5211   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5212   \seq_clear:N \l_tmpa_seq
5213   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5214     \seq_map_inline:cn{c_stex_module_##1_constants}{
5215       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5216     }
5217   }
5218   \exp_args:Nnno
5219   \prop_gput:cn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5220   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5221   \stex_add_structure_to_current_module:nn
5222     \l__stex_structures_name_str
5223     \l_stex_last_feature_str

```

```

5224
5225 \stex_execute_in_module:x {
5226   \tl_set:cn { #1 }{
5227     \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
5228   }
5229 }
5230 }
5231
5232 \cs_new:Nn \stex_invoke_structure:nn {
5233   \stex_invoke_symbol:n { #1?#2 }
5234 }
5235
5236 \cs_new_protected:Nn \stex_get_structure:n {
5237   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5238     \tl_set:Nn \l_tmpa_tl { #1 }
5239     \__stex_structures_get_from_cs:
5240   }{
5241     \cs_if_exist:cTF { #1 }{
5242       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5243       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5244       \str_if_empty:NNTF \l_tmpa_str {
5245         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
5246           \__stex_structures_get_from_cs:
5247         }{
5248           \__stex_structures_get_from_string:n { #1 }
5249         }
5250       }{
5251         \__stex_structures_get_from_string:n { #1 }
5252       }
5253     }{
5254       \__stex_structures_get_from_string:n { #1 }
5255     }
5256   }
5257 }
5258
5259 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5260   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5261     { \tl_tail:N \l_tmpa_tl }
5262   \str_set:Nx \l_tmpa_str {
5263     \exp_after:wN \use_i:nn \l_tmpa_tl
5264   }
5265   \str_set:Nx \l_tmpb_str {
5266     \exp_after:wN \use_ii:nn \l_tmpa_tl
5267   }
5268   \str_set:Nx \l_stex_get_structure_str {
5269     \l_tmpa_str ? \l_tmpb_str
5270   }
5271   \str_set:Nx \l_stex_get_structure_module_str {
5272     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5273   }
5274 }
5275
5276 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5277   \tl_set:Nn \l_tmpa_tl {

```

```

5278 \msg_error:nnn{stex}{error/unknownstructure}{#1}
5279 }
5280 \str_set:Nn \l_tmpa_str { #1 }
5281 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5282
5283 \seq_map_inline:Nn \l_stex_all_modules_seq {
5284   \prop_if_exist:cT {c_stex_module_##1_structures} {
5285     \prop_map_inline:cn {c_stex_module_##1_structures} {
5286       \exp_args:No \str_if_eq:nnT \l_tmpa_str {####1}{
5287         \% \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5288           \prop_map_break:n{\seq_map_break:n{
5289             \tl_set:Nn \l_tmpa_tl {
5290               \str_set:Nn \l_stex_get_structure_str {##1?####1}
5291               \str_set:Nn \l_stex_get_structure_module_str {####2}
5292             }
5293           }}
5294         }
5295       }
5296     }
5297   }
5298   \l_tmpa_tl
5299 }

```

**\instantiate**

```

5300
5301 \NewDocumentEnvironment{usestructure}{m}{
5302   \stex_get_structure:n {#1}
5303   \exp_args:Nnx \stex_debug:nn{features}{using~structure:~\l_stex_get_structure_module_str}
5304   \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5305 }{}
5306
5307 \keys_define:nn { stex / instantiate } {
5308   name .str_set_x:N = \l__stex_structures_name_str
5309 }
5310 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5311   \str_clear:N \l__stex_structures_name_str
5312   \keys_set:nn { stex / instantiate } { #1 }
5313 }
5314
5315 \NewDocumentEnvironment{extstructure}{m m O{}}{
5316   \begin{mathstructure_inner}{#1}[#3]
5317     \seq_set_split:Nnn\__stex_structures_extstructure_imports_seq,{#2}
5318     \seq_map_inline:Nn\__stex_structures_extstructure_imports_seq {
5319       \stex_get_structure:n {##1}
5320       \exp_args:Nnx \stex_debug:nn{features}{importing~structure:~\l_stex_get_structure_modu
5321       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5322       \stex_if_smsmode:F {
5323         \stex_annotate_invisible:nnn
5324         {import} {\l_stex_get_structure_module_str} {}
5325       }
5326       \exp_args:Nx \stex_add_import_to_current_module:n {
5327         \l_stex_get_structure_module_str
5328       }
5329       \exp_args:Nx \stex_add_to_current_module:n {

```

```

5330         \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5331     }
5332 }
5333 \stex_smsmode_do:
5334 \ignorespacesandpars
5335 }{
5336 \end{mathstructure_inner}
5337 }
5338
5339 \NewDocumentEnvironment{extstructure*}{m m O{}}{
5340 % TODO
5341 \begin{extstructure}{#1}{#2}{#3}
5342 }{
5343 \end{extstructure}
5344 }
5345
5346 \NewDocumentCommand \instantiate {m O{ } m m O{}}{
5347 \begin{group}
5348 \stex_get_structure:n {#3}
5349 \__stex_structures_instantiate_args:n { #2 }
5350 \str_if_empty:NT \l__stex_structures_name_str {
5351 \str_set:Nn \l__stex_structures_name_str { #1 }
5352 }
5353 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5354 \seq_clear:N \l__stex_structures_fields_seq
5355 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5356 \seq_map_inline:Nn \l_stex_collect_imports_seq {
5357 \seq_map_inline:cn {c_stex_module_##1_constants}{
5358 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
5359 }
5360 }
5361
5362 \tl_if_empty:nF{#5}{
5363 \seq_set_split:Nnn \l_tmpa_seq , {#5}
5364 \prop_clear:N \l_tmpa_prop
5365 \seq_map_inline:Nn \l_tmpa_seq {
5366 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5367 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5368 \msg_error:nnn{stex}{error/keyval}{##1}
5369 }
5370 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5371 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5372 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5373 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
5374 \exp_args:Nxx \str_if_eq:nnF
5375 {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5376 {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5377 \msg_error:nnxxx{stex}{error/incompatible}
5378 {\l__stex_structures_dom_str}
5379 {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5380 {\l_stex_get_symbol_uri_str}
5381 {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5382 }
5383 \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str

```

```

5384     }
5385 }
5386
5387 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5388   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5389   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5390
5391   \stex_add_constant_to_current_module:n {\l_tmpa_str}
5392   \stex_execute_in_module:x {
5393     \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
5394       name   = \l_tmpa_str ,
5395       args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5396       arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5397       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5398       argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5399   }
5400   \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5401 }
5402
5403 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5404   \stex_find_notation:nn{##1}{ }
5405   \stex_execute_in_module:x {
5406     \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
5407   }
5408
5409   \stex_copy_control_sequence_ii:ccN
5410     {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5411     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5412     \l_tmpa_tl
5413   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5414
5415
5416   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5417     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5418     \stex_execute_in_module:x {
5419       \tl_set:cn
5420         {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
5421         { \exp_args:No \exp_not:n \l_tmpa_cs}
5422     }
5423   }
5424
5425 }
5426
5427 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
5428 }
5429
5430 \stex_execute_in_module:x {
5431   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
5432     domain = \l_stex_get_structure_module_str ,
5433     \prop_to_keyval:N \l_tmpa_prop
5434   }
5435   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
5436 }
5437 \stex_debug:nn{instantiate}{

```

```

5438 Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\  

5439 \prop_to_keyval:N \l_tmpa_prop  

5440 }  

5441 \exp_args:Nxx \stex_symdecl_do:nn {  

5442   type={\STEXsymbol{module-type}}{  

5443     \STEXInternalTermMathOMSiiii {  

5444       \l_stex_get_structure_module_str  

5445     }{}{0}{}  

5446   }}  

5447 }{\l__stex_structures_name_str}  

5448 % {  

5449   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures  

5450   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}  

5451   \stex_notation_do:nnnnn{}{0}{}{\comp{#4}}  

5452 % }  

5453 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}  

5454 \endgroup  

5455 \stex_smsmode_do:\ignorespacesandpars  

5456 }  

5457  

5458 \cs_new_protected:Nn \stex_symbol_or_var:n {  

5459   \cs_if_exist:cTF{#1}{  

5460     \cs_set_eq:Nc \l_tmpa_tl { #1 }  

5461     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }  

5462     \str_if_empty:NTF \l_tmpa_str {  

5463       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }  

5464       \stex_invoke_variable:n {  

5465         \bool_set_true:N \l_stex_symbol_or_var_bool  

5466         \bool_set_false:N \l_stex_instance_or_symbol_bool  

5467         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}  

5468         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}  

5469         \str_set:Nx \l_stex_get_symbol_uri_str {  

5470           \exp_after:wN \use:n \l_tmpa_tl  

5471         }  

5472       }{ % TODO \stex_invoke_varinstance:n  

5473         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {  

5474           \bool_set_true:N \l_stex_symbol_or_var_bool  

5475           \bool_set_true:N \l_stex_instance_or_symbol_bool  

5476           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}  

5477           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}  

5478           \str_set:Nx \l_stex_get_symbol_uri_str {  

5479             \exp_after:wN \use:n \l_tmpa_tl  

5480           }  

5481         }{  

5482           \bool_set_false:N \l_stex_symbol_or_var_bool  

5483           \stex_get_symbol:n{#1}  

5484         }  

5485       }  

5486     }{  

5487       \__stex_structures_symbolorvar_from_string:n{ #1 }  

5488     }  

5489   }{  

5490     \__stex_structures_symbolorvar_from_string:n{ #1 }  

5491   }  


```

```

5492 }
5493
5494 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
5495   \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5496     \bool_set_true:N \l_stex_symbol_or_var_bool
5497     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5498   }{
5499     \bool_set_false:N \l_stex_symbol_or_var_bool
5500     \stex_get_symbol:n{#1}
5501   }
5502 }
5503
5504 \keys_define:nn { stex / varinstantiate } {
5505   name .str_set_x:N = \l__stex_structures_name_str,
5506   bind .choices:nn =
5507     {forall,exists}
5508     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5509 }
5510
5511 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5512   \str_clear:N \l__stex_structures_name_str
5513   \str_clear:N \l__stex_structures_bind_str
5514   \keys_set:nn { stex / varinstantiate } { #1 }
5515 }
5516
5517 \NewDocumentCommand \varinstantiate {m O{}} m m O{}}{
5518   \begingroup
5519     \stex_get_structure:n {#3}
5520     \__stex_structures_varinstantiate_args:n { #2 }
5521     \str_if_empty:NT \l__stex_structures_name_str {
5522       \str_set:Nn \l__stex_structures_name_str { #1 }
5523     }
5524     \stex_if_do_html:TF{
5525       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5526     }{\use:n}
5527     {
5528       \stex_if_do_html:T{
5529         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}}
5530       }
5531       \seq_clear:N \l__stex_structures_fields_seq
5532       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5533       \seq_map_inline:Nn \l_stex_collect_imports_seq {
5534         \seq_map_inline:cn {c_stex_module_##1_constants}{
5535           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5536         }
5537       }
5538       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5539       \prop_clear:N \l_tmpa_prop
5540       \tl_if_empty:nF {#5} {
5541         \seq_set_split:Nnn \l_tmpa_seq , {#5}
5542         \seq_map_inline:Nn \l_tmpa_seq {
5543           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5544           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5545             \msg_error:nnn{stex}{error/keyval}{##1}

```



```

5546     }
5547     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
5548     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5549     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri_str
5550     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5551     \stex_if_do_html:T{
5552         \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5553         \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\l_stex_get_symbol_uri_str}{}}
5554     }
5555     \bool_if:NTF \l_stex_symbol_or_var_bool {
5556         \exp_args:Nxx \str_if_eq:nnF
5557             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5558             {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}}{
5559             \msg_error:nnxxxx{stex}{error/incompatible}
5560             {\l__stex_structures_dom_str}
5561             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5562             {\l_stex_get_symbol_uri_str}
5563             {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}
5564         }
5565         \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {}}
5566     }{
5567         \exp_args:Nxx \str_if_eq:nnF
5568             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5569             {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}}{
5570             \msg_error:nnxxxx{stex}{error/incompatible}
5571             {\l__stex_structures_dom_str}
5572             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5573             {\l_stex_get_symbol_uri_str}
5574             {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5575         }
5576         \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {}}
5577     }
5578 }
5579 }
5580 \tl_gclear:N \g__stex_structures_aftergroup_tl
5581 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5582     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l__stex_structures_fields_seq \l__stex_structures_name_str} \l__stex_structures_name_str}
5583     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5584     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5585         \stex_find_notation:nn{##1}{}
5586         \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
5587             {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5588         \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
5589         \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5590             \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5591                 {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5592             \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_op_\l_tmpa_str _cs}}
5593         }
5594     }
5595 }
5596 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5597     \prop_set_from_keyval:cn { l_stex_symdecl_ var://\l_tmpa_str _prop}{
5598         name      = \l_tmpa_str ,
5599         args      = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,

```

```

5600         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5601         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5602         argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}} ,
5603     }
5604     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
5605     {g__stex_structures_tmpa_\l_tmpa_str_cs}
5606     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
5607     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5608 }
5609 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5610 }
5611 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5612     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
5613         domain = \l_stex_get_structure_module_str ,
5614         \prop_to_keyval:N \l_tmpa_prop
5615     }
5616     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5617     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
5618         \exp_args:Nnx \exp_not:N \use:nn {
5619             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5620                 \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5621                     \exp_not:n{
5622                         \_varcomp{#4}
5623                     }
5624                 }
5625             }{
5626                 \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5627             }
5628         }
5629     }
5630 }
5631 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5632 \aftergroup\g__stex_structures_aftergroup_tl
5633 \endgroup
5634 \stex_smsmode_do:\ignorespacesandpars
5635 }
5636
5637 \cs_new_protected:Nn \stex_invoke_instance:n {
5638     \peek_charcode_remove:NTF ! {
5639         \stex_invoke_symbol:n{#1}
5640     }{
5641         \_stex_invoke_instance:nn {#1}
5642     }
5643 }
5644
5645
5646 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5647     \peek_charcode_remove:NTF ! {
5648         \exp_args:Nnx \use:nn {
5649             \def\comp{\_varcomp}
5650             \use:c{l_stex_varinstance_#1_op_tl}
5651         }{
5652             \_stex_reset:N \comp
5653         }

```

```

5654 }{
5655   \_stex_invoke_varinstance:nn {#1}
5656 }
5657 }
5658
5659 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5660   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5661     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5662   }{
5663     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5664     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5665       \prop_to_keyval:N \l_tmpa_prop
5666     }
5667   }
5668 }
5669
5670 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5671   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5672     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5673     \l_tmpa_tl
5674   }{
5675     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
5676     }
5677   }

```

(End definition for `\instantiate`. This function is documented on page 38.)

`\stex_invoke_structure:nnn`

```

5678 % #1: URI of the instance
5679 % #2: URI of the instantiated module
5680 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5681   \tl_if_empty:nTF{ #3 }{
5682     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5683       c_stex_feature_ #2 _prop
5684     }
5685     \tl_clear:N \l_tmpa_tl
5686     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5687     \seq_map_inline:Nn \l_tmpa_seq {
5688       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5689       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5690       \cs_if_exist:cT {
5691         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5692       }{
5693         \tl_if_empty:NF \l_tmpa_tl {
5694           \tl_put_right:Nn \l_tmpa_tl {,}
5695         }
5696         \tl_put_right:Nx \l_tmpa_tl {
5697           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5698         }
5699       }
5700     }
5701     \exp_args:No \mathstrut \l_tmpa_tl
5702   }{
5703     \stex_invoke_symbol:n{#1/#3}

```

```

5704 }
5705 }

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

5706 </package>

```

## Chapter 34

# STEX -Statements Implementation

```
5707 <*package>
5708
5709 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5710
5711 <@@=stex_statements>
    Warnings and error messages
5712
\titleemph
5713 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

### 34.1 Definitions

#### definiendum

```
5714 \keys_define:nn {stex / definiendum }{
5715   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5716   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5717   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5718   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5719 }
5720 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5721   \str_clear:N \l__stex_statements_definiendum_root_str
5722   \tl_clear:N \l__stex_statements_definiendum_post_tl
5723   \str_clear:N \l__stex_statements_definiendum_gfa_str
5724   \keys_set:nn { stex / definiendum }{ #1 }
5725 }
5726 \NewDocumentCommand \definiendum { O{} m m } {
5727   \__stex_statements_definiendum_args:n { #1 }
5728   \stex_get_symbol:n { #2 }
5729   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5730   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5731     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5732     \tl_set:Nn \l_tmpa_tl { #3 }
5733   } {
5734     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5735     \tl_set:Nn \l_tmpa_tl {
5736       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5737     }
5738   }
5739 } {
5740   \tl_set:Nn \l_tmpa_tl { #3 }
5741 }
5742
5743 % TODO root
5744 \stex_html_backend:TF {
5745   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5746 } {
5747   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5748 }
5749 }
5750 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 48.)

#### definame

```

5751
5752 \NewDocumentCommand \definame { 0{ } m } {
5753   \__stex_statements_definiendum_args:n { #1 }
5754   % TODO: root
5755   \stex_get_symbol:n { #2 }
5756   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5757   \str_set:Nx \l_tmpa_str {
5758     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5759   }
5760   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5761   \stex_html_backend:TF {
5762     \stex_if_do_html:T {
5763       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5764         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5765       }
5766     }
5767   } {
5768     \exp_args:Nnx \defemph@uri {
5769       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5770     } { \l_stex_get_symbol_uri_str }
5771   }
5772 }
5773 \stex_deactivate_macro:Nn \definame {definition~environments}
5774
5775 \NewDocumentCommand \Definame { 0{ } m } {
5776   \__stex_statements_definiendum_args:n { #1 }
5777   \stex_get_symbol:n { #2 }
5778   \str_set:Nx \l_tmpa_str {
5779     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5780   }
5781   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5782 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5783 \stex_html_backend:TF {
5784   \stex_if_do_html:T {
5785     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5786       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5787     }
5788   }
5789 } {
5790   \exp_args:Nnx \defemph@uri {
5791     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5792   } { \l_stex_get_symbol_uri_str }
5793 }
5794 }
5795 \stex_deactivate_macro:Nn \Definame {definition-environments}
5796
5797 \NewDocumentCommand \premise { m }{
5798   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5799 }
5800 \NewDocumentCommand \conclusion { m }{
5801   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5802 }
5803 \NewDocumentCommand \definiens { 0{} m }{
5804   \str_clear:N \l_stex_get_symbol_uri_str
5805   \tl_if_empty:nF {#1} {
5806     \stex_get_symbol:n { #1 }
5807   }
5808   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5809     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5810       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5811     }{
5812       % TODO throw error
5813     }
5814   }
5815   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5816   {\l_stex_current_module_str}{
5817     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5818   }{true}{
5819     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5820     \exp_args:Nx \stex_add_to_current_module:n {
5821       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5822     }
5823   }
5824 }
5825 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5826 }
5827
5828 \NewDocumentCommand \varbindforall {m}{
5829   \stex_symbol_or_var:n {#1}
5830   \bool_if:NTF\l_stex_symbol_or_var_bool{
5831     \stex_if_do_html:T {
5832       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5833     }
5834   }{
5835     % todo throw error

```

```

5836 }
5837 }
5838
5839 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5840 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5841 \stex_deactivate_macro:Nn \definiens {definition~environments}
5842 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5843

```

(End definition for definame. This function is documented on page 48.)

sdefinition (env.)

```

5844
5845 \keys_define:nn {stex / sdefinition }{
5846   type      .str_set_x:N = \sdefinitiontype,
5847   id        .str_set_x:N = \sdefinitionid,
5848   name      .str_set_x:N = \sdefinitionname,
5849   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5850   title     .tl_set:N    = \sdefinitiontitle
5851 }
5852 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5853   \str_clear:N \sdefinitiontype
5854   \str_clear:N \sdefinitionid
5855   \str_clear:N \sdefinitionname
5856   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5857   \tl_clear:N \sdefinitiontitle
5858   \keys_set:nn { stex / sdefinition }{ #1 }
5859 }
5860
5861 \NewDocumentEnvironment{sdefinition}{0{}}{
5862   \__stex_statements_sdefinition_args:n{ #1 }
5863   \stex_reactivate_macro:N \definiendum
5864   \stex_reactivate_macro:N \definame
5865   \stex_reactivate_macro:N \Definame
5866   \stex_reactivate_macro:N \premise
5867   \stex_reactivate_macro:N \definiens
5868   \stex_reactivate_macro:N \varbindforall
5869   \stex_if_smsmode:F{
5870     \seq_clear:N \l_tmpb_seq
5871     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5872       \tl_if_empty:nF{ ##1 }{
5873         \stex_get_symbol:n { ##1 }
5874         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5875           \l_stex_get_symbol_uri_str
5876         }
5877       }
5878     }
5879     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5880     \exp_args:Nnnx
5881     \begin{sbox}{\stex_annotate_env}{\definition}{\seq_use:Nn \l_tmpb_seq {,}}
5882     \str_if_empty:NF \sdefinitiontype {
5883       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5884     }
5885     \str_if_empty:NF \sdefinitionname {

```



```

5886     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5887   }
5888   \clist_set:No \l_tmpa_clist \sdefinitiontype
5889   \tl_clear:N \l_tmpa_tl
5890   \clist_map_inline:Nn \l_tmpa_clist {
5891     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5892       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5893     }
5894   }
5895   \tl_if_empty:NTF \l_tmpa_tl {
5896     \__stex_statements_sdefinition_start:
5897   }{
5898     \l_tmpa_tl
5899   }
5900 }
5901 \stex_ref_new_doc_target:n \sdefinitionid
5902 \stex_smsmode_do:
5903 ){
5904   \stex_suppress_html:n {
5905     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5906   }
5907   \stex_if_smsmode:F {
5908     \clist_set:No \l_tmpa_clist \sdefinitiontype
5909     \tl_clear:N \l_tmpa_tl
5910     \clist_map_inline:Nn \l_tmpa_clist {
5911       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5912         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5913       }
5914     }
5915     \tl_if_empty:NTF \l_tmpa_tl {
5916       \__stex_statements_sdefinition_end:
5917     }{
5918       \l_tmpa_tl
5919     }
5920     \end{stex_annotate_env}
5921   }
5922 }

```

### **\stexpatchdefinition**

```

5923 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5924   \stex_par:\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
5925     ~(\sdefinitiontitle)
5926   }~}
5927 }
5928 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5929
5930 \newcommand\stexpatchdefinition[3]{} {
5931   \str_set:Nx \l_tmpa_str{ #1 }
5932   \str_if_empty:NTF \l_tmpa_str {
5933     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5934     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5935   }{
5936     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5937     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5938     }
5939 }

```

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

`\inlinedef` inline:

```

5940 \keys_define:nn {stex / inlinedef }{
5941   type      .str_set_x:N = \sdefinitiontype,
5942   id        .str_set_x:N = \sdefinitionid,
5943   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5944   name      .str_set_x:N = \sdefinitionname
5945 }
5946 \cs_new_protected:Nn \__stex_statements_inlinedef_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   \keys_set:nn { stex / inlinedef }{ #1 }
5952 }
5953 \NewDocumentCommand \inlinedef { 0{} m } {
5954   \begingroup
5955   \__stex_statements_inlinedef_args:n{ #1 }
5956   \stex_reactivate_macro:N \definiendum
5957   \stex_reactivate_macro:N \definame
5958   \stex_reactivate_macro:N \Definame
5959   \stex_reactivate_macro:N \premise
5960   \stex_reactivate_macro:N \definiens
5961   \stex_reactivate_macro:N \varbindforall
5962   \stex_ref_new_doc_target:n \sdefinitionid
5963   \stex_if_smsmode:TF{\stex_suppress_html:n {
5964     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5965   }}{
5966     \seq_clear:N \l_tmpb_seq
5967     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5968       \tl_if_empty:nF{ ##1 }{
5969         \stex_get_symbol:n { ##1 }
5970         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5971           \l_stex_get_symbol_uri_str
5972         }
5973       }
5974     }
5975     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5976     \ifvmode\noindent\fi
5977     \exp_args:Nnx
5978     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5979       \str_if_empty:NF \sdefinitiontype {
5980         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5981       }
5982       #2
5983       \str_if_empty:NF \sdefinitionname {
5984         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5985         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5986       }
5987     }

```

```

5988 }
5989 \endgroup
5990 \stex_smsmode_do:
5991 }

```

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

## 34.2 Assertions

sassertion (*env.*)

```

5993 \keys_define:nn {stex / sassertion }{
5994   type      .str_set_x:N = \sassertiontype,
5995   id        .str_set_x:N = \sassertionid,
5996   title     .tl_set:N     = \sassertiontitle ,
5997   for       .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5998   name      .str_set_x:N  = \sassertionname
5999 }
6000 \cs_new_protected:Nn \l__stex_statements_sassertion_args:n {
6001   \str_clear:N \sassertiontype
6002   \str_clear:N \sassertionid
6003   \str_clear:N \sassertionname
6004   \clist_clear:N \l__stex_statements_sassertion_for_clist
6005   \tl_clear:N \sassertiontitle
6006   \keys_set:nn { stex / sassertion }{ #1 }
6007 }
6008
6009 %\tl_new:N \g__stex_statements_aftergroup_tl
6010
6011 \NewDocumentEnvironment{sassertion}{0}{}{
6012   \l__stex_statements_sassertion_args:n{ #1 }
6013   \stex_reactivate_macro:N \premise
6014   \stex_reactivate_macro:N \conclusion
6015   \stex_reactivate_macro:N \varbindforall
6016   \stex_if_smsmode:F {
6017     \seq_clear:N \l_tmpb_seq
6018     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6019       \tl_if_empty:NF{ ##1 }{
6020         \stex_get_symbol:n { ##1 }
6021         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6022           \l_stex_get_symbol_uri_str
6023         }
6024       }
6025     }
6026     \exp_args:Nnnx
6027     \begin{sassertion_env}{assertion}{\seq_use:Nn \l_tmpb_seq {},}}
6028     \str_if_empty:NF \sassertiontype {
6029       \stex_annotate_invisible:nnn{type}{\sassertiontype}{-}
6030     }
6031     \str_if_empty:NF \sassertionname {
6032       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{-}
6033     }
6034     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

6035 \tl_clear:N \l_tmpa_tl
6036 \clist_map_inline:Nn \l_tmpa_clist {
6037   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
6038     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
6039   }
6040 }
6041 \tl_if_empty:NTF \l_tmpa_tl {
6042   \__stex_statements_sassertion_start:
6043 }{
6044   \l_tmpa_tl
6045 }
6046 }
6047 \str_if_empty:NTF \sassertionid {
6048   \str_if_empty:NF \sassertionname {
6049     \stex_ref_new_doc_target:n {}
6050   }
6051 } {
6052   \stex_ref_new_doc_target:n \sassertionid
6053 }
6054 \stex_smsmode_do:
6055 ){
6056   \str_if_empty:NF \sassertionname {
6057     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
6058   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6059 }
6060 \stex_if_smsmode:F {
6061   \clist_set:Nn \l_tmpa_clist \sassertiontype
6062   \tl_clear:N \l_tmpa_tl
6063   \clist_map_inline:Nn \l_tmpa_clist {
6064     \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
6065       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
6066     }
6067   }
6068   \tl_if_empty:NTF \l_tmpa_tl {
6069     \__stex_statements_sassertion_end:
6070   }{
6071     \l_tmpa_tl
6072   }
6073   \end{stex_annotate_env}
6074 }
6075 }

```

**\stexpatchassertion**

```

6076
6077 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
6078   \stex_par:\noindent\titileemph{Assertion~\tl_if_empty:NF \sassertiontitle {
6079     (\sassertiontitle)
6080   }~}
6081 }
6082 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
6083
6084 \newcommand\stexpatchassertion[3] [] {
6085   \str_set:Nx \l_tmpa_str{ #1 }
6086   \str_if_empty:NTF \l_tmpa_str {

```

```

6087     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
6088     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
6089   }{
6090     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
6091     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
6092   }
6093 }

```

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

`\inlineass` inline:

```

6094 \keys_define:nn {stex / inlineass }{
6095   type      .str_set_x:N = \sassertiontype,
6096   id        .str_set_x:N = \sassertionid,
6097   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
6098   name      .str_set_x:N = \sassertionname
6099 }
6100 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
6101   \str_clear:N \sassertiontype
6102   \str_clear:N \sassertionid
6103   \str_clear:N \sassertionname
6104   \clist_clear:N \l__stex_statements_sassertion_for_clist
6105   \keys_set:nn { stex / inlineass }{ #1 }
6106 }
6107 \NewDocumentCommand \inlineass { 0{} m } {
6108   \begin{group}
6109     \stex_reactivate_macro:N \premise
6110     \stex_reactivate_macro:N \conclusion
6111     \stex_reactivate_macro:N \varbindforall
6112     \__stex_statements_inlineass_args:n{ #1 }
6113     \str_if_empty:NTF \sassertionid {
6114       \str_if_empty:NF \sassertionname {
6115         \stex_ref_new_doc_target:n {}
6116       }
6117     } {
6118       \stex_ref_new_doc_target:n \sassertionid
6119     }
6120
6121     \stex_if_smsmode:TF{
6122       \str_if_empty:NF \sassertionname {
6123         \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
6124       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6125     }
6126   }{
6127     \seq_clear:N \l_tmpb_seq
6128     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6129       \tl_if_empty:nF{ ##1 }{
6130         \stex_get_symbol:n { ##1 }
6131         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6132           \l_stex_get_symbol_uri_str
6133         }
6134       }
6135     }
6136     \ifvmode\noindent\fi

```

```

6137 \exp_args:Nnx
6138 \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{
6139   \str_if_empty:NF \sassertiontype {
6140     \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
6141   }
6142   #2
6143   \str_if_empty:NF \sassertionname {
6144     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6145     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6146     \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
6147   }
6148 }
6149 }
6150 \endgroup
6151 \stex_smsmode_do:
6152 }

```

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

## 34.3 Examples

`sexample (env.)`

```

6153
6154 \keys_define:nn {stex / sexample }{
6155   type      .str_set_x:N = \exampletype,
6156   id        .str_set_x:N = \sexampleid,
6157   title     .tl_set:N     = \sexampletitle,
6158   name      .str_set_x:N = \sexamplename ,
6159   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
6160 }
6161 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
6162   \str_clear:N \sexampletype
6163   \str_clear:N \sexampleid
6164   \str_clear:N \sexamplename
6165   \tl_clear:N \sexampletitle
6166   \clist_clear:N \l__stex_statements_sexample_for_clist
6167   \keys_set:nn { stex / sexample }{ #1 }
6168 }
6169
6170 \NewDocumentEnvironment{sexample}{0{}}{
6171   \__stex_statements_sexample_args:n{ #1 }
6172   \stex_reactivate_macro:N \premise
6173   \stex_reactivate_macro:N \conclusion
6174   \stex_if_smsmode:F {
6175     \seq_clear:N \l_tmpb_seq
6176     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6177       \tl_if_empty:nF{ ##1 }{
6178         \stex_get_symbol:n { ##1 }
6179         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6180           \l_stex_get_symbol_uri_str
6181         }
6182       }
6183     }

```

```

6184 \exp_args:Nnnx
6185 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
6186 \str_if_empty:NF \sexamplotype {
6187   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{\}
6188 }
6189 \str_if_empty:NF \sexamplename {
6190   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{\}
6191 }
6192 \clist_set:No \l_tmpa_clist \sexamplotype
6193 \tl_clear:N \l_tmpa_tl
6194 \clist_map_inline:Nn \l_tmpa_clist {
6195   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6196     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
6197   }
6198 }
6199 \tl_if_empty:NTF \l_tmpa_tl {
6200   \__stex_statements_sexample_start:
6201 }{
6202   \l_tmpa_tl
6203 }
6204 }
6205 \str_if_empty:NF \sexampleid {
6206   \stex_ref_new_doc_target:n \sexampleid
6207 }
6208 \stex_smsmode_do:
6209 }{
6210   \str_if_empty:NF \sexamplename {
6211     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6212   }
6213   \stex_if_smsmode:F {
6214     \clist_set:No \l_tmpa_clist \sexamplotype
6215     \tl_clear:N \l_tmpa_tl
6216     \clist_map_inline:Nn \l_tmpa_clist {
6217       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6218         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6219       }
6220     }
6221     \tl_if_empty:NTF \l_tmpa_tl {
6222       \__stex_statements_sexample_end:
6223     }{
6224       \l_tmpa_tl
6225     }
6226     \end{stex_annotate_env}
6227   }
6228 }

```

**\stexpatchexample**

```

6229
6230 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6231   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
6232     (\sexampltitle)
6233   }~}
6234 }
6235 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}

```

```

6236
6237 \newcommand\stexpatchexample[3][] {
6238   \str_set:Nx \l_tmpa_str{ #1 }
6239   \str_if_empty:NTF \l_tmpa_str {
6240     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6241     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6242   }{
6243     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6244     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6245   }
6246 }

```

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

\inlineex inline:

```

6247 \keys_define:nn {stex / inlineex }{
6248   type      .str_set_x:N = \sexamplotype,
6249   id        .str_set_x:N = \sexampleid,
6250   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
6251   name      .str_set_x:N = \sexamplename
6252 }
6253 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6254   \str_clear:N \sexamplotype
6255   \str_clear:N \sexampleid
6256   \str_clear:N \sexamplename
6257   \clist_clear:N \l__stex_statements_sexample_for_clist
6258   \keys_set:nn { stex / inlineex }{ #1 }
6259 }
6260 \NewDocumentCommand \inlineex { 0{} m } {
6261   \begingroup
6262   \stex_reactivate_macro:N \premise
6263   \stex_reactivate_macro:N \conclusion
6264   \__stex_statements_inlineex_args:n{ #1 }
6265   \str_if_empty:NF \sexampleid {
6266     \stex_ref_new_doc_target:n \sexampleid
6267   }
6268   \stex_if_smsmode:TF{
6269     \str_if_empty:NF \sexamplename {
6270       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6271     }
6272   }{
6273     \seq_clear:N \l_tmpb_seq
6274     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6275       \tl_if_empty:NF{ ##1 }{
6276         \stex_get_symbol:n { ##1 }
6277         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6278           \l_stex_get_symbol_uri_str
6279         }
6280       }
6281     }
6282     \ifvmode\noindent\fi
6283     \exp_args:Nnx
6284     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6285       \str_if_empty:NF \sexamplotype {

```



```

6286 \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
6287 }
6288 #2
6289 \str_if_empty:NF \sexamplename {
6290   \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
6291   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
6292 }
6293 }
6294 }
6295 \endgroup
6296 \stex_smsmode_do:
6297 }

```

## 34.4 Logical Paragraphs

```

6298 \keys_define:nn { stex / sparagraph} {
6299   id      .str_set:x:N = \sparagraphid ,
6300   title   .tl_set:N     = \l_stex_sparagraph_title_tl ,
6301   type    .str_set:x:N = \sparagraphtype ,
6302   for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
6303   from    .tl_set:N     = \sparagraphfrom ,
6304   to      .tl_set:N     = \sparagraphto ,
6305   start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
6306   name    .str_set:N    = \sparagraphname ,
6307   imports .tl_set:N     = \l__stex_statements_sparagraph_imports_tl
6308 }
6309
6310 \cs_new_protected:Nn \stex_sparagraph_args:n {
6311   \tl_clear:N \l_stex_sparagraph_title_tl
6312   \tl_clear:N \sparagraphfrom
6313   \tl_clear:N \sparagraphto
6314   \tl_clear:N \l_stex_sparagraph_start_tl
6315   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6316   \str_clear:N \sparagraphid
6317   \str_clear:N \sparagraphtype
6318   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6319   \str_clear:N \sparagraphname
6320   \keys_set:nn { stex / sparagraph }{ #1 }
6321 }
6322 \newif\if@in@omtext\@in@omtextfalse
6323
6324 \NewDocumentEnvironment {sparagraph} { 0{} } {
6325   \stex_sparagraph_args:n { #1 }
6326   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6327     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6328   }{
6329     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6330   }
6331   \@in@omtexttrue
6332   \stex_if_smsmode:F {

```

```

6333 \seq_clear:N \l_tmpb_seq
6334 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6335   \tl_if_empty:nF{ ##1 }{
6336     \stex_get_symbol:n { ##1 }
6337     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6338       \l_stex_get_symbol_uri_str
6339     }
6340   }
6341 }
6342 \exp_args:Nnnx
6343 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6344 \str_if_empty:NF \sparagraphtype {
6345   \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6346 }
6347 \str_if_empty:NF \sparagraphfrom {
6348   \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6349 }
6350 \str_if_empty:NF \sparagraphto {
6351   \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6352 }
6353 \str_if_empty:NF \sparagraphname {
6354   \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6355 }
6356 \clist_set:No \l_tmpa_clist \sparagraphtype
6357 \tl_clear:N \l_tmpa_tl
6358 \clist_map_inline:Nn \sparagraphtype {
6359   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6360     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
6361   }
6362 }
6363 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6364 \tl_if_empty:NTF \l_tmpa_tl {
6365   \__stex_statements_sparagraph_start:
6366 }{
6367   \l_tmpa_tl
6368 }
6369 }
6370 \clist_set:No \l_tmpa_clist \sparagraphtype
6371 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6372 {
6373   \stex_reactivate_macro:N \definiendum
6374   \stex_reactivate_macro:N \definame
6375   \stex_reactivate_macro:N \Definame
6376   \stex_reactivate_macro:N \premise
6377   \stex_reactivate_macro:N \definiens
6378 }
6379 \str_if_empty:NTF \sparagraphid {
6380   \str_if_empty:NTF \sparagraphname {
6381     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6382       \stex_ref_new_doc_target:n {}
6383     }
6384   } {
6385     \stex_ref_new_doc_target:n {}
6386   }

```

```

6387 } {
6388   \stex_ref_new_doc_target:n \sparagraphid
6389 }
6390 \exp_args:NNx
6391 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6392   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6393     \tl_if_empty:nF{ ##1 }{
6394       \stex_get_symbol:n { ##1 }
6395       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6396     }
6397   }
6398 }
6399 \stex_smsmode_do:
6400 \ignorespacesandpars
6401 }{
6402   \str_if_empty:NF \sparagraphname {
6403     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6404     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6405   }
6406   \stex_if_smsmode:F {
6407     \clist_set:No \l_tmpa_clist \sparagraphtype
6408     \tl_clear:N \l_tmpa_tl
6409     \clist_map_inline:Nn \l_tmpa_clist {
6410       \tl_if_exist:cT {\__stex_statements_sparagraph_##1_end:}{
6411         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sparagraph_##1_end:}}
6412       }
6413     }
6414     \tl_if_empty:NTF \l_tmpa_tl {
6415       \__stex_statements_sparagraph_end:
6416     }{
6417       \l_tmpa_tl
6418     }
6419     \end{stex_annotate_env}
6420   }
6421 }

```

## \stexpatchparagraph

```

6422
6423 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6424   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6425     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6426       \titleemph{\l_stex_sparagraph_title_tl}:~
6427     }
6428   }{
6429     \titleemph{\l_stex_sparagraph_start_tl}~
6430   }
6431 }
6432 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6433
6434 \newcommand\stexpatchparagraph[3]{} {
6435   \str_set:Nx \l_tmpa_str{ #1 }
6436   \str_if_empty:NTF \l_tmpa_str {
6437     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6438     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }

```

```

6439     }{
6440         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
6441         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
6442     }
6443 }
6444
6445 \keys_define:nn { stex / inlinepara } {
6446     id      .str_set_x:N = \sparagraphid ,
6447     type    .str_set_x:N = \sparagraphtype ,
6448     for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6449     from    .tl_set:N    = \sparagraphfrom ,
6450     to      .tl_set:N    = \sparagraphto ,
6451     name    .str_set:N   = \sparagraphname
6452 }
6453 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6454     \tl_clear:N \sparagraphfrom
6455     \tl_clear:N \sparagraphto
6456     \str_clear:N \sparagraphid
6457     \str_clear:N \sparagraphtype
6458     \clist_clear:N \l__stex_statements_sparagraph_for_clist
6459     \str_clear:N \sparagraphname
6460     \keys_set:nn { stex / inlinepara }{ #1 }
6461 }
6462 \NewDocumentCommand \inlinepara { 0{} m } {
6463     \begingroup
6464     \__stex_statements_inlinepara_args:n{ #1 }
6465     \clist_set:Nn \l_tmpa_clist \sparagraphtype
6466     \str_if_empty:NTF \sparagraphid {
6467         \str_if_empty:NTF \sparagraphname {
6468             \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6469                 \stex_ref_new_doc_target:n {}
6470             }
6471         } {
6472             \stex_ref_new_doc_target:n {}
6473         }
6474     } {
6475         \stex_ref_new_doc_target:n \sparagraphid
6476     }
6477     \stex_if_smsmode:TF{
6478         \str_if_empty:NF \sparagraphname {
6479             \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6480             \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6481         }
6482     }{
6483         \seq_clear:N \l_tmpb_seq
6484         \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6485             \tl_if_empty:nF{ ##1 }{
6486                 \stex_get_symbol:n { ##1 }
6487                 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6488                     \l_stex_get_symbol_uri_str
6489                 }
6490             }
6491         }
6492         \ifvmode\noindent\fi

```

```

6493 \exp_args:Nnx
6494 \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6495   \str_if_empty:NF \sparagraphtype {
6496     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6497   }
6498   \str_if_empty:NF \sparagraphfrom {
6499     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6500   }
6501   \str_if_empty:NF \sparagraphto {
6502     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6503   }
6504   \str_if_empty:NF \sparagraphname {
6505     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6506     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6507     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6508   }
6509   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6510     \clist_map_inline:Nn \l_tmpb_seq {
6511       \stex_ref_new_sym_target:n {##1}
6512     }
6513   }
6514   #2
6515 }
6516 }
6517 \endgroup
6518 \stex_smsmode_do:
6519 }
6520

```

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

```

6521 </package>

```

## Chapter 35

# The Implementation

```
6522 <*package>
6523 <@@=stex_sproof>
6524
6525 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
6526
```

### 35.1 Proofs

We first define some keys for the proof environment.

```
6527 \keys_define:nn { stex / spf } {
6528   id          .str_set_x:N = \spfid,
6529   for          .clist_set:N = \l__stex_sproof_spf_for_clist ,
6530   from         .tl_set:N    = \l__stex_sproof_spf_from_tl ,
6531   proofend     .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
6532   type         .str_set_x:N = \spftype,
6533   title        .tl_set:N    = \spftitle,
6534   continues    .tl_set:N    = \l__stex_sproof_spf_continues_tl,
6535   functions    .tl_set:N    = \l__stex_sproof_spf_functions_tl,
6536   term         .tl_set:N    = \l__stex_sproof_spf_term_tl,
6537   method       .tl_set:N    = \l__stex_sproof_spf_method_tl,
6538   hide         .bool_set:N  = \l__stex_sproof_spf_hide_bool
6539 }
6540 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
6541   \str_clear:N \spfid
6542   \tl_clear:N \l__stex_sproof_spf_for_tl
6543   \tl_clear:N \l__stex_sproof_spf_from_tl
6544   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6545   \str_clear:N \spftype
6546   \tl_clear:N \spftitle
6547   \tl_clear:N \l__stex_sproof_spf_continues_tl
6548   \tl_clear:N \l__stex_sproof_spf_term_tl
6549   \tl_clear:N \l__stex_sproof_spf_functions_tl
6550   \tl_clear:N \l__stex_sproof_spf_method_tl
6551   \bool_set_false:N \l__stex_sproof_spf_hide_bool
6552   \keys_set:nn { stex / spf }{ #1 }
6553 }
6554 \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`

```
6555 \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.

```
6556 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6557 \cs_new_protected:Npn \sproofnumber {
6558   \int_set:Nn \l_tmpa_int {1}
6559   \bool_while_do:nn {
6560     \int_compare_p:nNn {
6561       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6562     } > 0
6563   }{
6564     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6565     \int_incr:N \l_tmpa_int
6566   }
6567 }
6568 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
6569   \int_set:Nn \l_tmpa_int {1}
6570   \bool_while_do:nn {
6571     \int_compare_p:nNn {
6572       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6573     } > 0
6574   }{
6575     \int_incr:N \l_tmpa_int
6576   }
6577   \int_compare:nNnF \l_tmpa_int = 1 {
6578     \int_decr:N \l_tmpa_int
6579   }
6580   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6581     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6582   }
6583 }
6584
6585 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6586   \int_set:Nn \l_tmpa_int {1}
6587   \bool_while_do:nn {
6588     \int_compare_p:nNn {
6589       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6590     } > 0
6591   }{
6592     \int_incr:N \l_tmpa_int
6593   }
6594   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6595 }
6596
```

```

6597 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6598   \int_set:Nn \l_tmpa_int {1}
6599   \bool_while_do:nn {
6600     \int_compare_p:nNn {
6601       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6602     } > 0
6603   }{
6604     \int_incr:N \l_tmpa_int
6605   }
6606   \int_decr:N \l_tmpa_int
6607   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6608 }

```

**\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

```

6609 \def\sproof@box{
6610   \ltx@ifpackageloaded{amssymb}{\square$}{
6611     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6612   }
6613 }
6614 \def\sproofend{
6615   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6616     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6617   }
6618 }

```

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

**spf@\*kw**

```

6619 \def\spf@proofsketch@kw{Proof~Sketch}
6620 \def\spf@proof@kw{Proof}
6621 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6622 \AddToHook{begindocument}{
6623   \ltx@ifpackageloaded{babel}{
6624     \makeatletter
6625     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6626     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6627       \input{sproof-ngerman.ldf}
6628     }
6629     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6630       \input{sproof-finnish.ldf}
6631     }
6632     \clist_if_in:NnT \l_tmpa_clist {french}{
6633       \input{sproof-french.ldf}
6634     }
6635     \clist_if_in:NnT \l_tmpa_clist {russian}{
6636       \input{sproof-russian.ldf}
6637     }
6638     \makeatother
6639   }{}
6640 }

```



**spfsketch**

```

6641 \newcommand\spfsketch[2] [] {
6642   \begin{group}
6643   \let \premise \stex_proof_premise:
6644   \_stex_sproof_spf_args:n{#1}
6645   \stex_if_smsmode:TF {
6646     \str_if_empty:NF \spfid {
6647       \stex_ref_new_doc_target:n \spfid
6648     }
6649   }{
6650     \seq_clear:N \l_tmpa_seq
6651     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6652       \tl_if_empty:nF{ ##1 }{
6653         \stex_get_symbol:n { ##1 }
6654         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6655           \l_stex_get_symbol_uri_str
6656         }
6657       }
6658     }
6659     \exp_args:Nnx
6660     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6661       \str_if_empty:NF \spftype {
6662         \stex_annotate_invisible:nnn{type}{\spftype}{ }
6663       }
6664       \clist_set:Nn \l_tmpa_clist \spftype
6665       \tl_set:Nn \l_tmpa_tl {
6666         \titleemph{
6667           \tl_if_empty:NTF \spftitle {
6668             \spf@proofsketch@kw
6669           }{
6670             \spftitle
6671           }
6672         }:-
6673       }
6674       \clist_map_inline:Nn \l_tmpa_clist {
6675         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6676           \tl_clear:N \l_tmpa_tl
6677         }
6678       }
6679       \str_if_empty:NF \spfid {
6680         \stex_ref_new_doc_target:n \spfid
6681       }
6682       \l_tmpa_tl #2 \sproofend
6683     }
6684   }
6685   \endgroup
6686   \stex_smsmode_do:
6687 }
6688

```

(End definition for *spfsketch*. This function is documented on page 54.)

```

\_stex_sproof_maybe_comment:
\_stex_sproof_maybe_comment_end:
\_stex_sproof_start_comment:
6689 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6690
6691 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6692   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6693     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6694   }
6695 }
6696 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6697   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6698 }
6699 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6700   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6701 }
6702

```

(End definition for \\_\_stex\_sproof\_maybe\_comment:, \\_\_stex\_sproof\_maybe\_comment\_end:, and \\_\_stex\_sproof\_start\_comment:.)

\stexcommentfont

```

6703 \cs_new_protected:Npn \stexcommentfont {
6704   \small\itshape
6705 }

```

(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.

```

6706 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6707   \seq_clear:N \l_tmpa_seq
6708   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6709     \tl_if_empty:NF{ ##1 }{
6710       \stex_get_symbol:n { ##1 }
6711       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6712         \l_stex_get_symbol_uri_str
6713       }
6714     }
6715   }
6716   \exp_args:Nnnx
6717   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6718   \str_if_empty:NF \spftype {
6719     \stex_annotate_invisible:nnn{type}{\spftype}{}
6720   }
6721   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6722   \str_if_empty:NF \spfid {
6723     \stex_ref_new_doc_target:n \spfid
6724   }
6725   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6726   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6727     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6728   }
6729   \begin{list}{}{
6730     \setlength\topsep{0pt}
6731     \setlength\parsep{0pt}
6732     \setlength\rightmargin{0pt}

```

```

6733 } \__stex_sproof_maybe_comment:
6734 }
6735 \cs_new_protected:Nn \__stex_sproof_end_env:n {
6736   \stex_if_smsmode:F{
6737     \__stex_sproof_maybe_comment_end:
6738     \end{list}
6739     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6740       \stex_html_backend:F{\egroup}
6741     }
6742     \clist_set:No \l_tmpa_clist \spftype
6743     #1
6744     \end{stex_annotate_env}
6745     \end{stex_annotate_env}
6746   }
6747 }
6748 }
6749 \NewDocumentEnvironment{sproof}{s O{} m}{
6750   \intarray_gzero:N \l__stex_sproof_counter_intarray
6751   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6752   \stex_reactivate_macro:N \yield
6753   \stex_reactivate_macro:N \eqstep
6754   \stex_reactivate_macro:N \assumption
6755   \stex_reactivate_macro:N \conclude
6756   \stex_reactivate_macro:N \spfstep
6757   \__stex_sproof_spf_args:n{#2}
6758   \stex_if_smsmode:TF {
6759     \str_if_empty:NF \spfid {
6760       \stex_ref_new_doc_target:n \spfid
6761     }
6762   }{
6763     \__stex_sproof_start_env:nnn{sproof}{#3}{
6764       \clist_set:No \l_tmpa_clist \spftype
6765       \tl_clear:N \l_tmpa_tl
6766       \clist_map_inline:Nn \l_tmpa_clist {
6767         \tl_if_exist:cT {\__stex_sproof_sproof_##1_start:}{
6768           \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_start:}}
6769         }
6770         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6771           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6772         }
6773       }
6774       \tl_if_empty:NTF \l_tmpa_tl {
6775         \__stex_sproof_sproof_start:
6776       }{
6777         \l_tmpa_tl
6778       }
6779     }
6780   }
6781   \stex_smsmode_do:
6782 }{\__stex_sproof_end_env:n{
6783   \tl_clear:N \l_tmpa_tl
6784   \clist_map_inline:Nn \l_tmpa_clist {
6785     \tl_if_exist:cT {\__stex_sproof_sproof_##1_end:}{
6786       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_end:}}

```

```

6787     }
6788   }
6789   \tl_if_empty:NTF \l_tmpa_tl {
6790     \__stex_sproof_sproof_end:
6791   }{
6792     \l_tmpa_tl
6793   }
6794 }
6795 \NewDocumentEnvironment{subproof}{s O{} m}{
6796   \__stex_sproof_spf_args:n{#2}
6797   \stex_if_smsmode:TF {
6798     \str_if_empty:NF \spfid {
6799       \stex_ref_new_doc_target:n \spfid
6800     }
6801   }{
6802     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6803   }
6804   \__stex_sproof_add_counter:
6805   \stex_smsmode_do:
6806 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6807   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6808     \__stex_sproof_inc_counter:
6809   }
6810   \aftergroup\__stex_sproof_maybe_comment:
6811 }
6812 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6813
6814 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6815   \par\noindent\titleemph{
6816     \tl_if_empty:NTF \spftype {
6817       \spf@proof@kw
6818     }{
6819       \spftype
6820     }
6821   }:
6822 }
6823 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6824
6825 \newcommand\stexpatchproof[3] [] {
6826   \str_set:Nx \l_tmpa_str{ #1 }
6827   \str_if_empty:NTF \l_tmpa_str {
6828     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6829     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6830   }{
6831     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6832     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6833   }
6834 }

\pstep
\conclude
\assumption
\have
\eqstep

```

```

6835 \keys_define:nn { stex / spfsteps } {
6836   id          .str_set_x:N = \spfstepid,
6837   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,

```

```

6839 type .str_set_x:N = \spftype,
6840 title .tl_set:N = \spftitle,
6841 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6842 term .tl_set:N = \l__stex_sproof_spf_term_tl
6843 }
6844 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6845 \str_clear:N \spfstepid
6846 \clist_clear:N \l__stex_sproof_spf_for_clist
6847 \str_clear:N \spftype
6848 \tl_clear:N \l__stex_sproof_spf_method_tl
6849 \tl_clear:N \l__stex_sproof_spf_term_tl
6850 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6851 \keys_set:nn { stex / spfsteps }{ #1 }
6852 }
6853
6854 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6855 \NewDocumentCommand #1 {s O{} +m} {
6856 \__stex_sproof_maybe_comment_end:
6857
6858 \__stex_sproof_spfstep_args:n{##2}
6859 \stex_annotate:nnn{spfstep}{#2}{
6860 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6861 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6862 }
6863 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6864 #4
6865 }{
6866 \item[\IfBooleanTF ##1 {}{#3}]
6867 }
6868 \ignorespacesandpars ##3
6869 }
6870 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6871 \__stex_sproof_maybe_comment:
6872 }
6873 \stex_deactivate_macro:Nn #1 {sproof~environments}
6874 }
6875
6876 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6877 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6878 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6879
6880 \NewDocumentCommand \eqstep {s m}{
6881 \__stex_sproof_maybe_comment_end:
6882 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6883 $=$
6884 }{
6885 \item[$=$]
6886 }
6887 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6888 \__stex_sproof_maybe_comment:
6889 }
6890 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6891
6892 \NewDocumentCommand \yield {+m}{

```

```

6893 \stex_annotate:nnn{spfyield}{\}{ #1 }
6894 }
6895 \stex_deactivate_macro:Nn \yield {sproof~environments}
6896
6897 \NewDocumentEnvironment{spfblock}{\}{
6898   \item[]
6899   \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6900 }{
6901   \aftergroup\__stex_sproof_maybe_comment:
6902 }
6903 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6904

```

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

### **\spfidea**

```

6905 \NewDocumentCommand\spfidea{0{} +m}{
6906   \__stex_sproof_spf_args:n{#1}
6907   \titleemph{
6908     \tl_if_empty:NTF \spftype {Proof-Idea}{
6909       \spftype
6910     }
6911   }~#2
6912   \sproofend
6913 }

```

*(End definition for \spfidea. This function is documented on page 54.)*

```

6914 \newcommand\spfjust[1]{
6915   #1
6916 }
6917 \</package>

```

Some auxiliary code, and clean up to be executed at the end of the package.

## Chapter 36

# STEX -Others Implementation

```
6918 <*package>
6919
6920 %%%%%%%%%% others.dtx %%%%%%%%%%
6921
6922 <@@=stex_others>
        Warnings and error messages
6923 % None

\MSC Math subject classifier

6924 \NewDocumentCommand \MSC {m} {
6925 % TODO
6926 }

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

6927 \@ifpackageloaded{tikzinput}{
6928 \RequirePackage{stex-tikzinput}
6929 }{}
6930
6931 \bool_if:NT \c_stex_persist_mode_bool {
6932 \let__stex_notation_restore_notation_old:nnnnn
6933 \__stex_notation_restore_notation:nnnnn
6934 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6935 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6936 \ExplSyntaxOn
6937 }
6938 \def__stex_notation_restore_notation:nnnnn{
6939 \ExplSyntaxOff
6940 \catcode'\sim10
6941 \__stex_notation_restore_notation_new:nnnnn
6942 }
6943 \input{\jobname.sms}
6944 \let__stex_notation_restore_notation:nnnnn
6945 \__stex_notation_restore_notation_old:nnnnn
6946 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```

6947 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6948 \l_tmpa_str
6949 \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6950 \c_stex_mathhub_main_manifest_prop
6951 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6952 }
6953 }
6954
6955 \stex_get_document_uri:
6956 </package>

```



## Chapter 37

# STEX -Metatheory Implementation

```
6957 <*package>
6958 <@@=stex_modules>
6959
6960 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6961
6962 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6963 \begingroup
6964 \stex_module_setup:nn{
6965   ns=\c_stex_metatheory_ns_str,
6966   meta=NONE
6967 }{Metatheory}
6968 \stex_reactivate_macro:N \symdecl
6969 \stex_reactivate_macro:N \notation
6970 \stex_reactivate_macro:N \symdef
6971 \ExplSyntaxOff
6972 \csname stex_suppress_html:n\endcsname{
6973   % is-a (a:A, a \in A, a is an A, etc.)
6974   \symdecl{isa}[args=ai]
6975   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6976   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6977   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6978
6979   % bind (\forall, \Pi, \lambda etc.)
6980   \symdecl{bind}[args=Bi,assoc=pre]
6981   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6982   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6983   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6984
6985   % implicit bind
6986   \symdecl{implicitbind}[args=Bi,assoc=pre]
6987   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6988     \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6989     \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6990
6991   % dummy variable
```

```

6992 \symdecl{dummyvar}
6993 \notation{dummyvar}[underscore]{\comp\_}
6994 \notation{dummyvar}[dot]{\comp\cdot}
6995 \notation{dummyvar}[dash]{\comp{\rm --}}
6996
6997 %fromto (function space, Hom-set, implication etc.)
6998 \symdecl{fromto}[args=ai]
6999 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
7000 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
7001
7002 % mapto (lambda etc.)
7003 \symdecl{mapto}[args=Bi]
7004 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
7005 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
7006 %\notation{mapto}[lambdau]{\comp\lambda_#1 \comp.\; #2}{#1 \comp, #2}
7007
7008 % function/operator application
7009 \symdecl{apply}[args=ia]
7010 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
7011 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
7012
7013 % collection of propositions/booleans/truth values
7014 \symdecl{prop}[name=proposition]
7015 \notation{prop}[prop]{\comp{\rm prop}}
7016 \notation{prop}[BOOL]{\comp{\rm BOOL}}
7017
7018 \symdecl{judgmentholds}[args=1]
7019 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
7020
7021 % sequences
7022 \symdecl{seqtype}[args=1]
7023 \notation{seqtype}[kleene]{#1^{\comp\ast}}
7024
7025 \symdecl{seqexpr}[args=a]
7026 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
7027
7028 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
7029 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
7030 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
7031 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
7032 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp
7033 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
7034 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7035 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
7036 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7037
7038 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\{#1\}_#2}
7039 \notation{sequence-index}[ui,prec=nobrackets]{\{#1\}^#2}
7040
7041 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
7042 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
7043 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#
7044
7045 % nat literals

```

```

7046 \symdef{natliteral}{\comp{\mathtt{Ord}}}
7047
7048 % letin (''let'', local definitions, variable substitution)
7049 \symdecl{letin}[args=bii]
7050 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
7051 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
7052 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
7053
7054 % structures
7055 \symdecl*{module-type}[args=1]
7056 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
7057 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
7058 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
7059
7060 % objects
7061 \symdecl{object}
7062 \notation{object}{\comp{\mathtt{OBJECT}}}
7063
7064 }
7065
7066 % The following are abbreviations in the sTeX corpus that are left over from earlier
7067 % developments. They will eventually be phased out.
7068
7069 \ExplSyntaxOn
7070 \stex_add_to_current_module:n{
7071   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
7072   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
7073   \def\livar{\csname sequence-index\endcsname[li]}
7074   \def\uivar{\csname sequence-index\endcsname[ui]}
7075   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
7076   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
7077 }
7078 \__stex_modules_end_module:
7079 \endgroup
7080 </package>

```

## Chapter 38

# Tikzinput Implementation

```
7081 <@@=tikzinput>
7082 <*package>
7083
7084 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
7085
7086 \ProvidesExplPackage{tikzinput}{2022/09/14}{3.2.0}{tikzinput package}
7087 \RequirePackage{l3keys2e}
7088
7089 \keys_define:nn { tikzinput } {
7090   image .bool_set:N = \c_tikzinput_image_bool,
7091   image .default:n = false ,
7092   unknown .code:n = {}
7093 }
7094
7095 \ProcessKeysOptions { tikzinput }
7096
7097 \bool_if:NTF \c_tikzinput_image_bool {
7098   \RequirePackage{graphicx}
7099
7100   \providecommand\usetikzlibrary[]{}
7101   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
7102 }{
7103   \RequirePackage{tikz}
7104   \RequirePackage{standalone}
7105
7106   \newcommand \tikzinput [2] [] {
7107     \setkeys{Gin}{#1}
7108     \ifx \Gin@ewidth \Gin@exclamation
7109       \ifx \Gin@eheight \Gin@exclamation
7110         \input { #2 }
7111       \else
7112         \resizebox{!}{ \Gin@eheight }{
7113           \input { #2 }
7114         }
7115       \fi
7116     \else
7117       \ifx \Gin@eheight \Gin@exclamation
7118         \resizebox{ \Gin@ewidth }{!}{
```

```

7119         \input { #2 }
7120     }
7121     \else
7122         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
7123             \input { #2 }
7124         }
7125     \fi
7126 \fi
7127 }
7128 }
7129
7130 \newcommand \ctikzinput [2] [] {
7131     \begin{center}
7132         \tikzinput [#1] {#2}
7133     \end{center}
7134 }
7135
7136 \@ifpackageloaded{stex}{
7137     \RequirePackage{stex-tikzinput}
7138 }{}
7139
7140 </package>
7141 <*stex>
7142 \ProvidesExplPackage{stex-tikzinput}{2022/09/14}{3.2.0}{stex-tikzinput}
7143 \RequirePackage{stex}
7144 \RequirePackage{tikzinput}
7145
7146 \newcommand\mhtikzinput[2] [] {%
7147     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
7148     \stex_in_repository:nn\Gin@mhrepos{
7149         \tikzinput[#1]{\mhp\path{##1}{#2}}
7150     }
7151 }
7152 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
7153
7154 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
7155     \pgfkeys@spdef\pgf@temp{#1}
7156     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
7157     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
7158     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
7159     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
7160     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
7161     \catcode'\@=11
7162     \catcode'\|=12
7163     \catcode'\$=3
7164     \pgfutil@InputIfFileExists{#2}{-}{-}
7165     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
7166     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
7167     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
7168 }
7169
7170
7171 \newcommand\libusetikzlibrary[1]{

```

```

7172 \prop_if_exist:NF \l_stex_current_repository_prop {
7173   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7174 }
7175 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
7176   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7177 }
7178 \seq_clear:N \l__tikzinput_libinput_files_seq
7179 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
7180 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
7181
7182 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
7183   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
7184   \IfFileExists{ \l_tmpa_str }{
7185     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7186   }{}
7187   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
7188   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
7189 }
7190
7191 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
7192 \IfFileExists{ \l_tmpa_str }{
7193   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7194 }{}
7195
7196 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
7197   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
7198 }{
7199   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7200     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7201       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7202     }
7203   }{
7204     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7205   }
7206 }
7207 }
7208 </stex>

```

## Chapter 39

# document-structure.sty Implementation

```
7209 <*package>
7210 <@@=document_structure>
7211 \ProvidesExplPackage{document-structure}{2022/09/14}{3.2.0}{Modular Document Structure}
7212 \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).

```
7213
7214 \keys_define:nn{ document-structure }{
7215   class      .str_set_x:N = \c_document_structure_class_str,
7216   topsect    .str_set_x:N = \c_document_structure_topsect_str,
7217   unknown    .code:n      = {
7218     \PassOptionsToClass{\CurrentOption}{stex}
7219     \PassOptionsToClass{\CurrentOption}{tikzinput}
7220   }
7221   % showignores .bool_set:N = \c_document_structure_showignores_bool,
7222 }
7223 \ProcessKeysOptions{ document-structure }
7224 \str_if_empty:NT \c_document_structure_class_str {
7225   \str_set:Nn \c_document_structure_class_str {article}
7226 }
7227 \str_if_empty:NT \c_document_structure_topsect_str {
7228   \str_set:Nn \c_document_structure_topsect_str {section}
7229 }
```

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

```
7230 \RequirePackage{xspace}
7231 \RequirePackage{comment}
7232 \RequirePackage{stex}
7233 \AddToHook{begindocument}{
```

```

7234 \ltx@ifpackageloaded{babel}{
7235     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7236     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7237         \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7238     }
7239 }{}
7240 }

```

`\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.

```

7241 \int_new:N \l_document_structure_section_level_int
7242 \str_case:NnF \c_document_structure_topsect_str {
7243     {part}}{
7244     \int_set:Nn \l_document_structure_section_level_int {0}
7245 }
7246 {chapter}}{
7247     \int_set:Nn \l_document_structure_section_level_int {1}
7248 }
7249 }{
7250     \str_case:NnF \c_document_structure_class_str {
7251         {book}}{
7252             \int_set:Nn \l_document_structure_section_level_int {0}
7253         }
7254         {report}}{
7255             \int_set:Nn \l_document_structure_section_level_int {0}
7256         }
7257     }{
7258         \int_set:Nn \l_document_structure_section_level_int {2}
7259     }
7260 }

```

## 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

```

7261 \def\current@section@level{document}%
7262 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
7263 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

7264 \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.



```

7265 \ifcase\l_document_structure_section_level_int
7266 \or\stepcounter{part}
7267 \or\stepcounter{chapter}
7268 \or\stepcounter{section}
7269 \or\stepcounter{subsection}
7270 \or\stepcounter{subsubsection}
7271 \or\stepcounter{paragraph}
7272 \or\stepcounter{subparagraph}
7273 \fi
7274 }

```

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

`blindfragment (env.)`

```

7275 \newcommand\at@begin@blindsfragment[1]{
7276 \newenvironment{blindfragment}
7277 {
7278 \int_incr:N\l_document_structure_section_level_int
7279 \at@begin@blindsfragment\l_document_structure_section_level_int
7280 }{}

```

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

```

7281 \newcommand\sfragment@nonum[2]{
7282 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7283 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7284 }

```

(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.

```

7285 \newcommand\sfragment@num[2]{
7286 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7287 \@nameuse{#1}{#2}
7288 }{
7289 \cs_if_exist:NTF\rdfmata@sectioning{
7290 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7291 }{
7292 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7293 }
7294 }
7295 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7296 }

```

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

`sfragment (env.)`

```

7297 \keys_define:nn { document-structure / sfragment }{
7298 id .str_set_x:N = \l__document_structure_sfragment_id_str,
7299 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

7300 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7301 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7302 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
7303 type          .tl_set:N     = \l__document_structure_sfragment_type_tl,
7304 short         .tl_set:N     = \l__document_structure_sfragment_short_tl,
7305 intro         .tl_set:N     = \l__document_structure_sfragment_intro_tl,
7306 imports       .tl_set:N     = \l__document_structure_sfragment_imports_tl,
7307 loadmodules   .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
7308 }
7309 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
7310   \str_clear:N \l__document_structure_sfragment_id_str
7311   \str_clear:N \l__document_structure_sfragment_date_str
7312   \clist_clear:N \l__document_structure_sfragment_creators_clist
7313   \clist_clear:N \l__document_structure_sfragment_contributors_clist
7314   \tl_clear:N \l__document_structure_sfragment_srccite_tl
7315   \tl_clear:N \l__document_structure_sfragment_type_tl
7316   \tl_clear:N \l__document_structure_sfragment_short_tl
7317   \tl_clear:N \l__document_structure_sfragment_imports_tl
7318   \tl_clear:N \l__document_structure_sfragment_intro_tl
7319   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7320   \keys_set:nn { document-structure / sfragment } { #1 }
7321 }

```

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.

```

7322 \newif\if@mainmatter\@mainmattertrue
7323 \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.

```

7324 \keys_define:nn { document-structure / sectioning }{
7325   name      .str_set_x:N = \l__document_structure_sect_name_str ,
7326   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
7327   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
7328   clear     .default:n   = {true} ,
7329   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
7330   num       .default:n   = {true}
7331 }
7332 \cs_new_protected:Nn \l__document_structure_sect_args:n {
7333   \str_clear:N \l__document_structure_sect_name_str
7334   \str_clear:N \l__document_structure_sect_ref_str
7335   \bool_set_false:N \l__document_structure_sect_clear_bool
7336   \bool_set_false:N \l__document_structure_sect_num_bool
7337   \keys_set:nn { document-structure / sectioning } { #1 }
7338 }
7339 \newcommand\omdoc@sectioning[3][]{
7340   \l__document_structure_sect_args:n {#1 }
7341   \let\omdoc@sect@name\l__document_structure_sect_name_str
7342   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7343   \if@mainmatter% numbering not overridden by frontmatter, etc.
7344     \bool_if:NTF \l__document_structure_sect_num_bool {
7345       \sfragment@num{#2}{#3}
7346     }{

```

```

7347     \sfragment@nonum{#2}{#3}
7348   }
7349   \def\current@section@level{\omdoc@sect@name}
7350   \else
7351     \sfragment@nonum{#2}{#3}
7352   \fi
7353 }% 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.

```

7354 \newcommand\sfragment@redefine@addtocontents[1]{%
7355 %\edef\__document_structureimport{#1}%
7356 %\@for\@I:=\__document_structureimport\do{%
7357 %\edef\@path{\csname module@\@I @path\endcsname}%
7358 %\@ifundefined{tf@toc}\relax%
7359 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
7360 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7361 %\def\addcontentsline##1##2##3{%
7362 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7363 %\else% hyperref.sty not loaded
7364 %\def\addcontentsline##1##2##3{%
7365 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7366 %\fi
7367 }% 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.

```

7368 \newenvironment{sfragment}[2][ ]% keys, title
7369 {
7370   \__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.

```

7371   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7372
7373   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7374     \sfragment@redefine@addtocontents{
7375       %\@ifundefined{module@id}\used@modules%
7376       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7377     }
7378   }

```

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

```

7379
7380   \stex_document_title:n { #2 }
7381
7382   \int_incr:N\l__document_structure_section_level_int
7383   \ifcase\l__document_structure_section_level_int
7384     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7385     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7386     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
7387     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

7388 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7389 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
7390 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragraph}{#1}
7391 \fi
7392 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
7393 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7394   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7395 }
7396 }% for customization
7397 {}

```

and finally, we localize the sections

```

7398 \newcommand\omdoc@part@kw{Part}
7399 \newcommand\omdoc@chapter@kw{Chapter}
7400 \newcommand\omdoc@section@kw{Section}
7401 \newcommand\omdoc@subsection@kw{Subsection}
7402 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7403 \newcommand\omdoc@paragraph@kw{paragraph}
7404 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

### 39.3 Front and Backmatter

Index markup is provided by the `omtext` package [Kohlhase:smmtf:git], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

7405 \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).

```

7406 \cs_if_exist:NTF\frontmatter{
7407   \let\__document_structure_orig_frontmatter\frontmatter
7408   \let\frontmatter\relax
7409 }{
7410   \tl_set:Nn\__document_structure_orig_frontmatter{
7411     \clearpage
7412     \@mainmatterfalse
7413     \pagenumbering{roman}
7414   }
7415 }
7416 \cs_if_exist:NTF\backmatter{
7417   \let\__document_structure_orig_backmatter\backmatter
7418   \let\backmatter\relax
7419 }{
7420   \tl_set:Nn\__document_structure_orig_backmatter{
7421     \clearpage
7422     \@mainmatterfalse
7423     \pagenumbering{roman}
7424   }

```

7425 }

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.

```
7426 \newenvironment{frontmatter}{
7427   \__document_structure_orig_frontmatter
7428 }{
7429   \cs_if_exist:NTF\mainmatter{
7430     \mainmatter
7431   }{
7432     \clearpage
7433     \@mainmattertrue
7434     \pagenumbering{arabic}
7435   }
7436 }
```

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

```
7437 \newenvironment{backmatter}{
7438   \__document_structure_orig_backmatter
7439 }{
7440   \cs_if_exist:NTF\mainmatter{
7441     \mainmatter
7442   }{
7443     \clearpage
7444     \@mainmattertrue
7445     \pagenumbering{arabic}
7446   }
7447 }
```

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

```
7448 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7449 \def \c__document_structure_document_str{document}
7450 \newcommand\afterprematurestop{}
7451 \def\prematurestop@endsfragment{
7452   \unless\ifx\@currenvir\c__document_structure_document_str
7453     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7454       \expandafter\prematurestop@endsfragment
7455     }
7456   }
7457 \providecommand\prematurestop{
7458   \message{Stopping~sTeX~processing~prematurely}
7459   \prematurestop@endsfragment
7460   \afterprematurestop
7461   \end{document}
7462 }
```

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

## 39.4 Global Variables

**\setSGvar** set a global variable

```
7463 \RequirePackage{etoolbox}
7464 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

**\useSGvar** use a global variable

```
7465 \newrobustcmd\useSGvar[1]{%
7466   \@ifundefined{sTeX@Gvar@#1}
7467   {\PackageError{document-structure}
7468    {The sTeX Global variable #1 is undefined}
7469    {set it with \protect\setSGvar}}
7470   \@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.

```
7471 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7472   \@ifundefined{sTeX@Gvar@#1}
7473   {\PackageError{document-structure}
7474    {The sTeX Global variable #1 is undefined}
7475    {set it with \protect\setSGvar}}
7476   {\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.

```
7477 \*cls)
7478 \@@=notesslides)
7479 \ProvidesExplClass{notesslides}{2022/09/14}{3.2.0}{notesslides Class}
7480 \RequirePackage{13keys2e}
7481
7482 \keys_define:nn{notesslides / cls}{
7483   class .str_set_x:N = \c__notesslides_class_str,
7484   notes .bool_set:N = \c__notesslides_notes_bool ,
7485   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7486   docopt .str_set_x:N = \c__notesslides_docopt_str,
7487   unknown .code:n = {
7488     \PassOptionsToPackage{\CurrentOption}{document-structure}
7489     \PassOptionsToClass{\CurrentOption}{beamer}
7490     \PassOptionsToPackage{\CurrentOption}{notesslides}
7491     \PassOptionsToPackage{\CurrentOption}{stex}
7492   }
7493 }
7494 \ProcessKeysOptions{ notesslides / cls }
7495
7496 \str_if_empty:NF \c__notesslides_class_str {
7497   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7498 }
7499
7500 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7501   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7502 }
7503 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7504   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7505 }
7506
7507 \RequirePackage{stex}
```

```

7508 \stex_html_backend:T {
7509   \bool_set_true:N\c__notesslides_notes_bool
7510 }
7511
7512 \bool_if:NTF \c__notesslides_notes_bool {
7513   \PassOptionsToPackage{notes=true}{notesslides}
7514   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7515 }{
7516   \PassOptionsToPackage{notes=false}{notesslides}
7517   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7518 }
7519 \</cls>

```

now we do the same for the notesslides package.

```

7520 <*package>
7521 \ProvidesExplPackage{notesslides}{2022/09/14}{3.2.0}{notesslides Package}
7522 \RequirePackage{l3keys2e}
7523
7524 \keys_define:nn{notesslides / pkg}{
7525   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7526   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7527   notes            .bool_set:N = \c__notesslides_notes_bool ,
7528   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7529   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7530   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7531   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7532   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
7533   unknown          .code:n      = {
7534     \PassOptionsToClass{\CurrentOption}{stex}
7535     \PassOptionsToClass{\CurrentOption}{tikzinput}
7536   }
7537 }
7538 \ProcessKeysOptions{ notesslides / pkg }
7539
7540 \RequirePackage{stex}
7541 \stex_html_backend:T {
7542   \bool_set_true:N\c__notesslides_notes_bool
7543 }
7544
7545 \newif\ifnotes
7546 \bool_if:NTF \c__notesslides_notes_bool {
7547   \notesttrue
7548 }{
7549   \notesfalse
7550 }
7551

```

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

```

7552 \str_if_empty:NTF \c__notesslides_topsect_str {
7553   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7554 }{
7555   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7556 }
7557 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```



```
7558 </package>
```

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

```
7559 <*cls>
7560 \bool_if:NTF \c__notesslides_notes_bool {
7561   \str_if_empty:NT \c__notesslides_class_str {
7562     \str_set:Nn \c__notesslides_class_str {article}
7563   }
7564   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
7565     {\c__notesslides_class_str}
7566 }{
7567   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7568   \newcounter{Item}
7569   \newcounter{paragraph}
7570   \newcounter{subparagraph}
7571   \newcounter{Hfootnote}
7572 }
7573 \RequirePackage{document-structure}
```

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

```
7574 \RequirePackage{notesslides}
7575 </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  $\text{\LaTeX}$  packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the  $\text{\LaTeX}$ -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.

```
7576 <*package>
7577 \bool_if:NT \c__notesslides_notes_bool {
7578   \RequirePackage{a4wide}
7579   \RequirePackage{marginnote}
7580   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7581   \RequirePackage{mdframed}
7582   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7583   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7584 }
7585 \RequirePackage{stex-tikzinput}
7586 \RequirePackage{comment}
7587 \RequirePackage{url}
7588 \RequirePackage{graphicx}
7589 \RequirePackage{pgf}
7590 \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.

```
7591 \bool_if:NT \c__notesslides_notes_bool {
7592   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7593 }
```

```

7594 \NewDocumentCommand \libusetheme {0{} m} {
7595   \libusepackage[#1]{beamertheme#2}
7596 }
7597

```

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

```

7598 \newcounter{slide}
7599 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7600 \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.

```

7601 \bool_if:NTF \c__notesslides_notes_bool {
7602   \renewenvironment{note}{\ignorespaces}{}
7603 }{
7604   \excludecomment{note}
7605 }

```

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.

```

7606 \bool_if:NT \c__notesslides_notes_bool {
7607   \newlength{\slideframewidth}
7608   \setlength{\slideframewidth}{1.5pt}

```

**frame** (*env.*) We first define the keys.

```

7609 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7610   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7611     \bool_set_true:N #1
7612   }{
7613     \bool_set_false:N #1
7614   }
7615 }
7616 \keys_define:nn{notesslides / frame}{
7617   label .str_set_x:N = \l__notesslides_frame_label_str,
7618   allowframebreaks .code:n = {
7619     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7620   },
7621   allowdisplaybreaks .code:n = {
7622     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7623   },
7624   fragile .code:n = {
7625     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7626   },
7627   shrink .code:n = {
7628     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7629   },
7630   squeeze .code:n = {
7631     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7632   },
7633   t .code:n = {

```

```

7634     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7635   },
7636   unknown    .code:n      = {}
7637 }
7638 \cs_new_protected:Nn \__notesslides_frame_args:n {
7639   \str_clear:N \l__notesslides_frame_label_str
7640   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7641   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7642   \bool_set_true:N \l__notesslides_frame_fragile_bool
7643   \bool_set_true:N \l__notesslides_frame_shrink_bool
7644   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7645   \bool_set_true:N \l__notesslides_frame_t_bool
7646   \keys_set:nn { notesslides / frame }{ #1 }
7647 }

```

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

```

7648 \renewenvironment{frame}[1][]{
7649   \__notesslides_frame_args:n{#1}
7650   \sffamily
7651   \stepcounter{slide}
7652   \def\@currentlabel{\theslide}
7653   \str_if_empty:NF \l__notesslides_frame_label_str {
7654     \label{\l__notesslides_frame_label_str}
7655   }

```

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

```

7656   \def\itemize@level{outer}
7657   \def\itemize@outer{outer}
7658   \def\itemize@inner{inner}
7659   \renewcommand\newpage{\addtocounter{framenum}{1}}
7660   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7661   \renewenvironment{itemize}{
7662     \ifx\itemize@level\itemize@outer
7663       \def\itemize@label{$\rhd$}
7664     \fi
7665     \ifx\itemize@level\itemize@inner
7666       \def\itemize@label{$\scriptstyle\rhd$}
7667     \fi
7668     \begin{list}
7669       {\itemize@label}
7670       {\setlength{\labelsep}{.3em}
7671        \setlength{\labelwidth}{.5em}
7672        \setlength{\leftmargin}{1.5em}
7673       }
7674     \edef\itemize@level{\itemize@inner}
7675   }{
7676     \end{list}
7677   }

```

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

```

7678   \stex_html_backend:TF {
7679     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7680     \mdf@patchamsthm
7681   }{
7682     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7683     }
7684   }{
7685     \stex_html_backend:TF {
7686       \miko@slidelabel\egroup\end{stex_annotate_env}
7687     }\medskip\miko@slidelabel\end{mdframed}}
7688   }

```

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

`\frametitle`

```

7689   \renewcommand{\frametitle}[1]{
7690     \stex_document_title:n { #1 }
7691     {\Large\bf\sf\color{blue}{#1}}\medskip
7692   }
7693 }

```

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

EdN:10

`\pause` 10

```

7694 \bool_if:NT \c__notesslides_notes_bool {
7695   \newcommand\pause{
7696   }

```

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

`nparagraph (env.)`

```

7697 \bool_if:NTF \c__notesslides_notes_bool {
7698   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7699 }{
7700   \excludecomment{nparagraph}
7701 }

```

`nfragment (env.)`

```

7702 \bool_if:NTF \c__notesslides_notes_bool {
7703   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1][#2]}\end{sfragment}}
7704 }{
7705   \excludecomment{nfragment}
7706 }

```

`ndefinition (env.)`

```

7707 \bool_if:NTF \c__notesslides_notes_bool {
7708   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7709 }{
7710   \excludecomment{ndefinition}
7711 }

```

`nassertion (env.)`

```

7712 \bool_if:NTF \c__notesslides_notes_bool {
7713   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7714 }{
7715   \excludecomment{nassertion}
7716 }

```

---

<sup>10</sup>EDNOTE: MK: fake it in notes mode for now

`nsproof (env.)`

```
7717 \bool_if:NTF \c__notesslides_notes_bool {
7718   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7719 }{
7720   \excludecomment{nproof}
7721 }
```

`nexample (env.)`

```
7722 \bool_if:NTF \c__notesslides_notes_bool {
7723   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7724 }{
7725   \excludecomment{nexample}
7726 }
```

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

```
7727 \def\inputref@preskip{\smallskip}
7728 \def\inputref@postskip{\medskip}
```

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

**`\inputref*`**

```
7729 \let\orig@inputref\inputref
7730 \def\inputref{\@ifstar\ninputref\orig@inputref}
7731 \newcommand\ninputref[2] []{
7732   \bool_if:NT \c__notesslides_notes_bool {
7733     \orig@inputref[#1]{#2}
7734   }
7735 }
```

*(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>}`.

```
7736 \newlength{\slidelogoheight}
7737
7738 \RequirePackage{graphicx}
7739
7740 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7741 \providecommand\mhgraphics[2] []{
7742   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7743   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7744 }
7745
7746 \bool_if:NTF \c__notesslides_notes_bool {
7747   \setlength{\slidelogoheight}{.4cm}
7748 }{
7749   \setlength{\slidelogoheight}{.25cm}
7750 }
```

```

7751 \ifcsname slidelogo\endcsname\else
7752 \newsavebox{\slidelogo}
7753 \sbox{\slidelogo}{\sTeX}
7754 \fi
7755 \newrobustcmd{\setslidelogo}[2][]{
7756 \tl_if_empty:nTF{#1}{
7757 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7758 }{
7759 \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7760 }
7761 }

```

(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.

```

7762 \bool_if:NT \c__notesslides_notes_bool {
7763 \def\author{\@dblarg\@ns@author}
7764 \long\def\@ns@author[#1]#2{%
7765 \def\c__notesslides_shortauthor{#1}%
7766 \def\@author{#2}
7767 }
7768 }

```

(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.

```

7769 \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.

```

7770 \def\copyrightnotice{%
7771 \footnotesize\copyright : \hspace{.3ex}%
7772 \ifcsname source\endcsname\source\else%
7773 \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7774 \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7775 ?source/author?\fi%
7776 \fi}
7777 \newsavebox{\cclogo}
7778 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7779 \newif\ifcchref\cchreffalse
7780 \AtBeginDocument{
7781 \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7782 }
7783 \def\licensing{
7784 \ifcchref
7785 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7786 \else
7787 {\usebox{\cclogo}}

```

```

7788 \fi
7789 }
7790 \newrobustcmd{\setlicensing}[2][]{
7791   \def\@url{#1}
7792   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7793   \ifx\@url\@empty
7794     \def\licensing{\usebox{\cclogo}}
7795   \else
7796     \def\licensing{
7797       \ifcchref
7798         \href{#1}{\usebox{\cclogo}}
7799       \else
7800         {\usebox{\cclogo}}
7801       \fi
7802     }
7803   \fi
7804 }

```

(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
7805 \newrobustcmd\miko@slidelabel{
7806   \vbox to \slidelogoheight{
7807     \vss\hbox to \slidewidth
7808       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7809   }
7810 }

```

(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.

```

7811 \def\Gin@mhrepos{}
7812 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7813 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7814 \newrobustcmd\frameimage[2][]{
7815   \stepcounter{slide}
7816   \bool_if:NT \c__notesslides_frameimages_bool {
7817     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7818     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7819     \begin{center}
7820       \bool_if:NTF \c__notesslides_fiboxed_bool {
7821         \fbox{
7822           \ifx\Gin@ewidth\@empty
7823             \ifx\Gin@mhrepos\@empty
7824               \mhgraphics[width=\slidewidth,#1]{#2}
7825             \else
7826               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7827             \fi
7828           \else% Gin@ewidth empty

```

<sup>11</sup>EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

7829         \ifx\Gin@mhrepos\@empty
7830         \mhgraphics[#1]{#2}
7831     \else
7832         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7833     \fi
7834 \fi% Gin@ewidth empty
7835 }
7836 }{
7837     \ifx\Gin@ewidth\@empty
7838     \ifx\Gin@mhrepos\@empty
7839         \mhgraphics[width=\slidewidth,#1]{#2}
7840     \else
7841         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7842     \fi
7843     \ifx\Gin@mhrepos\@empty
7844         \mhgraphics[#1]{#2}
7845     \else
7846         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7847     \fi
7848     \fi% Gin@ewidth empty
7849 }
7850 \end{center}
7851 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7852 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7853 }
7854 } % 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`.

```

7855 \stex_html_backend:F {
7856     \bool_if:NT \c__notesslides_sectocframes_bool {
7857         \str_if_eq:VnTF \__notesslidestopsect{part}{
7858             \newcounter{chapter}\counterwithin*{section}{chapter}
7859         }{
7860             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7861                 \newcounter{chapter}\counterwithin*{section}{chapter}
7862             }
7863         }
7864     }
7865 }

```

`\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`

```

7866 \def\part@prefix{}
7867 \@ifpackageloaded{document-structure}{}{
7868     \str_case:VnF \__notesslidestopsect {

```



```

7869 {part}{
7870   \int_set:Nn \l_document_structure_section_level_int {0}
7871   \def\thesection{\arabic{chapter}.\arabic{section}}
7872   \def\part@prefix{\arabic{chapter}.}
7873 }
7874 {chapter}{
7875   \int_set:Nn \l_document_structure_section_level_int {1}
7876   \def\thesection{\arabic{chapter}.\arabic{section}}
7877   \def\part@prefix{\arabic{chapter}.}
7878 }
7879 }{
7880   \int_set:Nn \l_document_structure_section_level_int {2}
7881   \def\part@prefix{}
7882 }
7883 }
7884
7885 \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.)`

```

7886 \renewenvironment{sfragment}[2][]{
7887   \__document_structure_sfragment_args:n { #1 }
7888   \int_incr:N \l_document_structure_section_level_int
7889   \bool_if:NT \c__notesslides_sectocframes_bool {
7890     \stepcounter{slide}
7891     \begin{frame}[noframenumbering]
7892     \vfill\Large\centering
7893     \red{
7894       \ifcase\l_document_structure_section_level_int\or
7895         \stepcounter{part}
7896         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7897         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7898         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7899         \def\currentsectionlevel{\omdoc@part@kw}
7900       \or
7901         \stepcounter{chapter}
7902         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7903         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7904         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7905         \def\currentsectionlevel{\omdoc@chapter@kw}
7906       \or
7907         \stepcounter{section}
7908         \def\__notesslideslabel{\part@prefix\arabic{section}}
7909         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7910         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7911         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7912         \def\currentsectionlevel{\omdoc@section@kw}
7913       \or
7914         \stepcounter{subsection}
7915         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7916         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

7917         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7918         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7919         \def\currentsectionlevel{\omdoc@subsection@kw}
7920     \or
7921         \stepcounter{subsubsection}
7922         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7923         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7924         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\thesubsubsection}
7925         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7926         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7927     \or
7928         \stepcounter{paragraph}
7929         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7930         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7931         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\theparagraph}
7932         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\theparagraph}
7933         \def\currentsectionlevel{\omdoc@paragraph@kw}
7934     \else
7935         \def\__notesslideslabel{}
7936         \def\currentsectionlevel{\omdoc@paragraph@kw}
7937     \fi% end ifcase
7938     \__notesslideslabel\quad #2%
7939 }%
7940 \vfill%
7941 \end{frame}%
7942 }
7943 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7944     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7945 }
7946 }{}
7947 }

```

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

```

7948 \def\inserttheorembodyfont{\normalfont}
7949 %\bool_if:NF \c__notesslides_notes_bool {
7950 % \defbeamertemplate{theorem begin}{miko}
7951 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7952 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7953 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7954 % \defbeamertemplate{theorem end}{miko}{%

```

and we set it as the default one.

```

7955 % \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.

```

7956 % \expandafter\def\csname Parent2\endcsname{}
7957 %}
7958
7959 \AddToHook{begindocument}{% this does not work for some reason
7960     \setbeamertemplate{theorems}[ams style]
7961 }
7962 \bool_if:NT \c__notesslides_notes_bool {
7963     \renewenvironment{columns}[1][{}]{%

```

```

7964     \par\noindent%
7965     \begin{minipage}%
7966     \slidewidth\centering\leavevmode%
7967   }{%
7968     \end{minipage}\par\noindent%
7969   }%
7970   \newsavebox\columnbox%
7971   \renewenvironment<>{column}[2][]{%
7972     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7973   }{%
7974     \end{minipage}\end{lrbox}\usebox\columnbox%
7975   }%
7976 }

7977 \bool_if:NTF \c__notesslides_noproblems_bool {
7978   \newenvironment{problems}{}{}
7979 }{
7980   \excludacomment{problems}
7981 }

```

## 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.

```

7982 \gdef\printexcursions{}
7983 \newcommand\excursionref[2]{% label, text
7984   \bool_if:NT \c__notesslides_notes_bool {
7985     \begin{sparagraph}[title=Excursion]
7986       #2 \sref[fallback=the appendix]{#1}.
7987     \end{sparagraph}
7988   }
7989 }
7990 \newcommand\activate@excursion[2][{}{
7991   \gappto\printexcursions{\inputref{#1}{#2}}
7992 }
7993 \newcommand\excursion[4][{}{ repos, label, path, text
7994   \bool_if:NT \c__notesslides_notes_bool {
7995     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7996   }
7997 }

```

(End definition for `\excursion`. This function is documented on page 66.)

**\excursiongroup**

```

7998 \keys_define:nn{notesslides / excursiongroup }{
7999   id          .str_set_x:N = \l__notesslides_excursion_id_str,
8000   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
8001   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
8002 }
8003 \cs_new_protected:Nn \__notesslides_excursion_args:n {
8004   \tl_clear:N \l__notesslides_excursion_intro_tl
8005   \str_clear:N \l__notesslides_excursion_id_str

```

```

8006 \str_clear:N \l__notesslides_excursion_mhrepos_str
8007 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
8008 }
8009 \newcommand\excursionsgroup[1][ ]{
8010 \__notesslides_excursion_args:n{ #1 }
8011 \ifdefempty\printexcursions{ }% only if there are excursions
8012 {\begin{note}
8013 \begin{sfragment}[#1]{Excursions}%
8014 \ifdefempty\l__notesslides_excursion_intro_tl}{
8015 \inputref[\l__notesslides_excursion_mhrepos_str]{
8016 \l__notesslides_excursion_intro_tl
8017 }
8018 }
8019 \printexcursions%
8020 \end{sfragment}
8021 \end{note}}
8022 }
8023 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
8024 \</package>

```

(End definition for \excursionsgroup. 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.

```
8025 <*package>
8026 <@@=problems>
8027 \ProvidesExplPackage{problem}{2022/09/14}{3.2.0}{Semantic Markup for Problems}
8028 \RequirePackage{13keys2e}
8029 \RequirePackage{amssymb}% for \Box
8030
8031 \keys_define:nn { problem / pkg }{
8032   notes      .default:n    = { true },
8033   notes      .bool_set:N   = \c__problems_notes_bool,
8034   gnotes     .default:n    = { true },
8035   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
8036   hints      .default:n    = { true },
8037   hints      .bool_set:N   = \c__problems_hints_bool,
8038   solutions  .default:n    = { true },
8039   solutions  .bool_set:N   = \c__problems_solutions_bool,
8040   pts        .default:n    = { true },
8041   pts        .bool_set:N   = \c__problems_pts_bool,
8042   min        .default:n    = { true },
8043   min        .bool_set:N   = \c__problems_min_bool,
8044   boxed      .default:n    = { true },
8045   boxed      .bool_set:N   = \c__problems_boxed_bool,
8046   test       .default:n    = { true },
8047   test       .bool_set:N   = \c__problems_test_bool,
8048   unknown    .code:n       = {
8049     \PassOptionsToPackage{\CurrentOption}{stex}
8050   }
8051 }
8052 \newif\ifsolutions
8053
8054 \ProcessKeysOptions{ problem / pkg }
8055 \bool_if:NTF \c__problems_solutions_bool {
8056   \solutionstrue
```

```

8057 }{
8058   \solutionsfalse
8059 }
8060 \RequirePackage{stex}

```

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

```

8061 \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.

```

8062 \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.

```

8063 \def\prob@problem@kw{Problem}
8064 \def\prob@solution@kw{Solution}
8065 \def\prob@hint@kw{Hint}
8066 \def\prob@note@kw{Note}
8067 \def\prob@grade@kw{Grading}
8068 \def\prob@pt@kw{pt}
8069 \def\prob@min@kw{min}
8070 \def\prob@correct@kw{Correct}
8071 \def\prob@wrong@kw{Wrong}

```

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

For the other languages, we set up triggers

```

8072 \AddToHook{begindocument}{
8073   \ltx@ifpackageloaded{babel}{
8074     \makeatletter
8075     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8076     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8077       \input{problem-ngerman.ldf}
8078     }
8079     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8080       \input{problem-finnish.ldf}
8081     }
8082     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8083       \input{problem-french.ldf}
8084     }
8085     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8086       \input{problem-russian.ldf}
8087     }
8088     \makeatother
8089   }{ }
8090 }

```

## 41.2 Problems and Solutions

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

```

8091 \keys_define:nn{ problem / problem }{
8092   id .str_set_x:N = \l__problems_prob_id_str,

```

```

8093 pts      .tl_set:N      = \l__problems_prob_pts_tl,
8094 min      .tl_set:N      = \l__problems_prob_min_tl,
8095 title    .tl_set:N      = \l__problems_prob_title_tl,
8096 type     .tl_set:N      = \l__problems_prob_type_tl,
8097 imports  .tl_set:N      = \l__problems_prob_imports_tl,
8098 name     .str_set_x:N     = \l__problems_prob_name_str,
8099 refnum   .int_set:N      = \l__problems_prob_refnum_int
8100 }
8101 \cs_new_protected:Nn \__problems_prob_args:n {
8102   \str_clear:N \l__problems_prob_id_str
8103   \str_clear:N \l__problems_prob_name_str
8104   \tl_clear:N \l__problems_prob_pts_tl
8105   \tl_clear:N \l__problems_prob_min_tl
8106   \tl_clear:N \l__problems_prob_title_tl
8107   \tl_clear:N \l__problems_prob_type_tl
8108   \tl_clear:N \l__problems_prob_imports_tl
8109   \int_zero_new:N \l__problems_prob_refnum_int
8110   \keys_set:nn { problem / problem }{ #1 }
8111   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
8112     \let\l__problems_prob_refnum_int\undefined
8113   }
8114 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

8115 \newcounter{problem}[section]
8116 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
8117 \def\theplainsproblem{\arabic{problem}}
8118 \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.

```

8119 \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

```

8120 \newcommand\prob@number{
8121   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
8122     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
8123   }{
8124     \int_if_exist:NTF \l__problems_prob_refnum_int {
8125       \prob@label{\int_use:N \l__problems_prob_refnum_int }
8126     }{
8127       \prob@label\theplainsproblem
8128     }
8129   }
8130 }
8131 \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.

```

8132 \newcommand\prob@title[3]{%
8133   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
8134     #2 \l__problems_inclprob_title_tl #3
8135   }{
8136     \tl_if_empty:NTF \l__problems_prob_title_tl {
8137       #1
8138     }{
8139       #2 \l__problems_prob_title_tl #3
8140     }
8141   }
8142 }

```

(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.

```

8143 \def\prob@heading{
8144   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
8145   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
8146 }

```

(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.)`

```

8147 \newenvironment{sproblem}[1][]{
8148   \__problems_prob_args:n{#1}%\sref@target%
8149   \@in@omtexttrue% we are in a statement (for inline definitions)
8150   \refstepcounter{sproblem}\record@problem
8151   \def\current@section@level{\prob@problem@kw}
8152
8153   \str_if_empty:NT \l__problems_prob_name_str {
8154     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
8155     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
8156     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
8157   }
8158
8159   \stex_if_do_html:T{
8160     \tl_if_empty:NF \l__problems_prob_title_tl {
8161       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
8162     }
8163   }
8164
8165   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
8166
8167   \stex_reactivate_macro:N \STEXexport
8168   \stex_reactivate_macro:N \importmodule

```



```

8169 \stex_reactivate_macro:N \symdecl
8170 \stex_reactivate_macro:N \notation
8171 \stex_reactivate_macro:N \symdef
8172
8173 \stex_if_do_html:T{
8174   \begin{stex_annotate_env} {problem} {
8175     \l_stex_module_ns_str ? \l_stex_module_name_str
8176   }
8177
8178   \stex_annotate_invisible:nnn{header}{} {
8179     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
8180     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
8181     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
8182       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
8183     }
8184   }
8185 }
8186
8187 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
8188
8189
8190 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
8191   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
8192 }{
8193   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
8194 }
8195 \str_if_exist:NTF \l__problems_inclprob_id_str {
8196   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
8197 }{
8198   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
8199 }
8200
8201
8202 \stex_if_smsmode:F {
8203   \clist_set:No \l_tmpa_clist \sproblemtype
8204   \tl_clear:N \l_tmpa_tl
8205   \clist_map_inline:Nn \l_tmpa_clist {
8206     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8207       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8208     }
8209   }
8210   \tl_if_empty:NTF \l_tmpa_tl {
8211     \__problems_sproblem_start:
8212   }{
8213     \l_tmpa_tl
8214   }
8215 }
8216 \stex_ref_new_doc_target:n \sproblemid
8217 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8218 }{
8219   \__stex_modules_end_module:
8220   \stex_if_smsmode:F{
8221     \clist_set:No \l_tmpa_clist \sproblemtype
8222     \tl_clear:N \l_tmpa_tl

```

```

8223 \clist_map_inline:Nn \l_tmpa_clist {
8224   \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8225     \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
8226   }
8227 }
8228 \tl_if_empty:NTF \l_tmpa_tl {
8229   \__problems_sproblem_end:
8230 }{
8231   \l_tmpa_tl
8232 }
8233 }
8234 \stex_if_do_html:T{
8235   \end{stex_annotate_env}
8236 }
8237
8238 \smallskip
8239 }
8240
8241 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8242
8243
8244
8245 \cs_new_protected:Nn \__problems_sproblem_start: {
8246   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
8247 }
8248 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
8249
8250 \newcommand\stexpatchproblem[3]{} {
8251   \str_set:Nx \l_tmpa_str{ #1 }
8252   \str_if_empty:NTF \l_tmpa_str {
8253     \tl_set:Nn \__problems_sproblem_start: { #2 }
8254     \tl_set:Nn \__problems_sproblem_end: { #3 }
8255   }{
8256     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8257     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8258   }
8259 }
8260
8261
8262 \bool_if:NT \c__problems_boxed_bool {
8263   \surroundwithmdframed{problem}
8264 }

```

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

```

8265 \def\record@problem{
8266   \protected@write\@auxout{}
8267   {
8268     \string\@problem{\prob@number}
8269     {
8270       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8271         \l__problems_inclprob_pts_tl
8272       }{
8273         \l__problems_prob_pts_tl
8274       }

```

```

8275 }%
8276 {
8277   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8278     \l__problems_inclprob_min_tl
8279   }{
8280     \l__problems_prob_min_tl
8281   }
8282 }
8283 }
8284 }

```

(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).

```

8285 \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.

```

8286 \keys_define:nn { problem / solution }{
8287   id          .str_set_x:N = \l__problems_solution_id_str ,
8288   for         .str_set_x:N = \l__problems_solution_for_str ,
8289   type        .str_set_x:N = \l__problems_solution_type_str ,
8290   title       .tl_set:N     = \l__problems_solution_title_tl
8291 }
8292 \cs_new_protected:Nn \__problems_solution_args:n {
8293   \str_clear:N \l__problems_solution_id_str
8294   \str_clear:N \l__problems_solution_type_str
8295   \str_clear:N \l__problems_solution_for_str
8296   \tl_clear:N \l__problems_solution_title_tl
8297   \keys_set:nn { problem / solution }{ #1 }
8298 }

```

**`\startsolutions`** for the `\startsolutions` macro we use the `\specialcomment` macro from the `comment` package. Note that we use the `\@startsolution` macro in the start codes, that parses the optional argument.

```

8299 \box_new:N \l__problems_solution_box
8300 \newenvironment{solution}[1][{}]{
8301   \__problems_solution_args:n{#1}
8302   \stex_html_backend:TF{
8303     \stex_if_do_html:T{
8304       \begin{stex_annotate_env}{solution}{}
8305       \str_if_empty:NF \l__problems_solution_type_str {
8306         \par\noindent
8307         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
8308       }
8309       \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8310     }
8311   }{
8312     \setbox\l__problems_solution_box\vbox\bgroup
8313     \par\smallskip\hrule\smallskip
8314     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
8315   }

```

```

8316 }{
8317   \stex_html_backend:TF{
8318     \stex_if_do_html:T{
8319       \end{stex_annotate_env}
8320     }
8321   }{
8322     \smallskip\hrule
8323     \egroup
8324     \bool_if:NT \c__problems_solutions_bool {
8325       \strut\par\noindent
8326       \box\l__problems_solution_box
8327     }
8328   }
8329 }
8330
8331 \newcommand\startsolutions{
8332   \bool_set_true:N \c__problems_solutions_bool
8333   \solutionstrue
8334   % \specialcomment{solution}{\@startsolution}{
8335   %   \bool_if:NF \c__problems_boxed_bool {
8336   %     \hrule\medskip
8337   %   }
8338   %   \end{small}%
8339   % }
8340   % \bool_if:NT \c__problems_boxed_bool {
8341   %   \surroundwithmdframed{solution}
8342   % }
8343 }

```

(End definition for \startsolutions. This function is documented on page 68.)

### \stopsolutions

```

8344 \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.)

```

8345 \bool_if:NTF \c__problems_notes_bool {
8346   \newenvironment{exnote}[1][]{
8347     \par\smallskip\hrule\smallskip
8348     \noindent\textbf{\prob@note@kw :~ }\small
8349   }{
8350     \smallskip\hrule
8351   }
8352 }{
8353   \excludacomment{exnote}
8354 }

```

### hint (env.)

```

8355 \bool_if:NTF \c__problems_notes_bool {
8356   \newenvironment{hint}[1][]{
8357     \par\smallskip\hrule\smallskip
8358     \noindent\textbf{\prob@hint@kw :~ }\small
8359   }{

```

```

8360     \smallskip\hrule
8361   }
8362   \newenvironment{exhint}[1][ ]{
8363     \par\smallskip\hrule\smallskip
8364     \noindent\textbf{\prob@hint@kw :~ }\small
8365   }{
8366     \smallskip\hrule
8367   }
8368   }{
8369     \excludecomment{hint}
8370     \excludecomment{exhint}
8371   }

```

**gnote** (*env.*)

```

8372 \bool_if:NTF \c__problems_notes_bool {
8373   \newenvironment{gnote}[1][ ]{
8374     \par\smallskip\hrule\smallskip
8375     \noindent\textbf{\prob@gnote@kw :~ }\small
8376   }{
8377     \smallskip\hrule
8378   }
8379   }{
8380     \excludecomment{gnote}
8381   }

```

## 41.3 Markup for Added Value Services

## 41.4 Multiple Choice Blocks

EdN:12

**mcb** (*env.*)<sup>12</sup>

```

8382 \newenvironment{mcb}{
8383   \begin{enumerate}
8384 }{
8385   \end{enumerate}
8386 }

```

we define the keys for the mcb macro

```

8387 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8388   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8389     \bool_set_true:N #1
8390   }{
8391     \bool_set_false:N #1
8392   }
8393 }
8394 \keys_define:nn { problem / mcb }{
8395   id      .str_set_x:N = \l__problems_mcc_id_str ,
8396   feedback .tl_set:N   = \l__problems_mcc_feedback_tl ,
8397   T       .default:n   = { false } ,
8398   T       .bool_set:N  = \l__problems_mcc_t_bool ,
8399   F       .default:n   = { false } ,

```

<sup>12</sup>EdNOTE: MK: maybe import something better here from a dedicated MC package

```

8400 F      .bool_set:N    = \l__problems_mcc_f_bool ,
8401 Ttext   .tl_set:N     = \l__problems_mcc_Ttext_tl ,
8402 Ftext   .tl_set:N     = \l__problems_mcc_Ftext_tl
8403 }
8404 \cs_new_protected:Nn \l__problems_mcc_args:n {
8405   \str_clear:N \l__problems_mcc_id_str
8406   \tl_clear:N \l__problems_mcc_feedback_tl
8407   \bool_set_false:N \l__problems_mcc_t_bool
8408   \bool_set_false:N \l__problems_mcc_f_bool
8409   \tl_clear:N \l__problems_mcc_Ttext_tl
8410   \tl_clear:N \l__problems_mcc_Ftext_tl
8411   \str_clear:N \l__problems_mcc_id_str
8412   \keys_set:nn { problem / mcc }{ #1 }
8413 }

```

**\mcc**

```

8414 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8415 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8416 \newcommand\mcc[2][] {
8417   \l__problems_mcc_args:n{ #1 }
8418   \item[{$\Box$}] #2
8419   \bool_if:NT \c__problems_solutions_bool {
8420     \
8421     \bool_if:NT \l__problems_mcc_t_bool {
8422       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8423     }
8424     \bool_if:NT \l__problems_mcc_f_bool {
8425       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8426     }
8427     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8428       \emph{\l__problems_mcc_feedback_tl}
8429     }
8430   }
8431 } %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.

```

8432 \newcommand\fillinsol[2][] {%
8433   \def\@test{#1}
8434   \quad%
8435   \ifsolutions\textcolor{red}{\@test!}\else%
8436   \fbox{\ifx\@test\empty\phantom{\huge{21}}\else\hspace{#1}\fi}%
8437   \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.

```

8438
8439 \keys_define:nn{ problem / inclproblem }{
8440   id      .str_set:x:N = \l__problems_inclprob_id_str,
8441   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8442   min     .tl_set:N    = \l__problems_inclprob_min_tl,
8443   title   .tl_set:N    = \l__problems_inclprob_title_tl,
8444   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
8445   type    .tl_set:N    = \l__problems_inclprob_type_tl,
8446   mhrepos .str_set:x:N = \l__problems_inclprob_mhrepos_str
8447 }
8448 \cs_new_protected:Nn \l__problems_inclprob_args:n {
8449   \str_clear:N \l__problems_prob_id_str
8450   \tl_clear:N \l__problems_inclprob_pts_tl
8451   \tl_clear:N \l__problems_inclprob_min_tl
8452   \tl_clear:N \l__problems_inclprob_title_tl
8453   \tl_clear:N \l__problems_inclprob_type_tl
8454   \int_zero_new:N \l__problems_inclprob_refnum_int
8455   \str_clear:N \l__problems_inclprob_mhrepos_str
8456   \keys_set:nn { problem / inclproblem }{ #1 }
8457   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8458     \let\l__problems_inclprob_pts_tl\undefined
8459   }
8460   \tl_if_empty:NT \l__problems_inclprob_min_tl {
8461     \let\l__problems_inclprob_min_tl\undefined
8462   }
8463   \tl_if_empty:NT \l__problems_inclprob_title_tl {
8464     \let\l__problems_inclprob_title_tl\undefined
8465   }
8466   \tl_if_empty:NT \l__problems_inclprob_type_tl {
8467     \let\l__problems_inclprob_type_tl\undefined
8468   }
8469   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8470     \let\l__problems_inclprob_refnum_int\undefined
8471   }
8472 }
8473
8474 \cs_new_protected:Nn \l__problems_inclprob_clear: {
8475   \let\l__problems_inclprob_id_str\undefined
8476   \let\l__problems_inclprob_pts_tl\undefined
8477   \let\l__problems_inclprob_min_tl\undefined
8478   \let\l__problems_inclprob_title_tl\undefined
8479   \let\l__problems_inclprob_type_tl\undefined
8480   \let\l__problems_inclprob_refnum_int\undefined
8481   \let\l__problems_inclprob_mhrepos_str\undefined
8482 }
8483 \l__problems_inclprob_clear:
8484
8485 \newcommand\includeproblem[2][ ]{
8486   \l__problems_inclprob_args:n{ #1 }

```

```

8487 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8488   \stex_html_backend:TF {
8489     \str_clear:N \l_tmpa_str
8490     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8491       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8492     }
8493     \stex_annotate_invisible:nnn{includeproblem}{
8494       \l_tmpa_str / #2
8495     }{}
8496   }{
8497     \begingroup
8498     \inputreftrue
8499     \tl_if_empty:nTF{ ##1 }{
8500       \input{#2}
8501     }{
8502       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8503     }
8504     \endgroup
8505   }
8506 }
8507 \__problems_inclprob_clear:
8508 }

```

(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.

```

8509 \AddToHook{enddocument}{
8510   \bool_if:NT \c__problems_pts_bool {
8511     \message{Total:~\arabic{pts}~points}
8512   }
8513   \bool_if:NT \c__problems_min_bool {
8514     \message{Total:~\arabic{min}~minutes}
8515   }
8516 }

```

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

```

8517 \def\pts#1{
8518   \bool_if:NT \c__problems_pts_bool {
8519     \marginpar{#1~\prob@pt@kw}
8520   }
8521 }
8522 \def\min#1{
8523   \bool_if:NT \c__problems_min_bool {
8524     \marginpar{#1~\prob@min@kw}
8525   }
8526 }

```

`\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.



```

8527 \newcounter{pts}
8528 \def\show@pts{
8529   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8530     \bool_if:NT \c__problems_pts_bool {
8531       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8532       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8533     }
8534   }{
8535     \tl_if_exist:NT \l__problems_prob_pts_tl {
8536       \bool_if:NT \c__problems_pts_bool {
8537         \tl_if_empty:NT\l__problems_prob_pts_tl{
8538           \tl_set:Nn \l__problems_prob_pts_tl {0}
8539         }
8540         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8541         \addtocounter{pts}{\l__problems_prob_pts_tl}
8542       }
8543     }
8544   }
8545 }

```

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

\show@min

```

8546 \newcounter{min}
8547 \def\show@min{
8548   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8549     \bool_if:NT \c__problems_min_bool {
8550       \marginpar{\l__problems_inclprob_min_tl\ min}
8551       \addtocounter{min}{\l__problems_inclprob_min_tl}
8552     }
8553   }{
8554     \tl_if_exist:NT \l__problems_prob_min_tl {
8555       \bool_if:NT \c__problems_min_bool {
8556         \tl_if_empty:NT\l__problems_prob_min_tl{
8557           \tl_set:Nn \l__problems_prob_min_tl {0}
8558         }
8559         \marginpar{\l__problems_prob_min_tl\ min}
8560         \addtocounter{min}{\l__problems_prob_min_tl}
8561       }
8562     }
8563   }
8564 }
8565 \</package>

```

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

## 41.8 Testing and Spacing

\testspace

```

8566 \newcommand\testspace[1]{\bool_if:NT \c__problems_boxed_bool {\vspace*{#1}}}

```

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

`\testnewpage`

```
8567 \newcommand\testnewpage{\bool_if:NT \c__problems_boxed_bool {\newpage}}
```

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

`\testemptypage`

```
8568 \newcommand\testemptypage[1][\%
```

```
8569 \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`

```
8570 \newcommand\testsmallspace{\testspace{1cm}}
```

```
8571 \newcommand\testmedspace{\testspace{2cm}}
```

```
8572 \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.

```
8573 {*package}
8574 \ProvidesExplPackage{hwexam}{2022/09/14}{3.2.0}{homework assignments and exams}
8575 \RequirePackage{13keys2e}
8576
8577 \newif\iftest\testfalse
8578 \DeclareOption{test}{\testtrue\PassOptionsToPackage{\CurrentOption}{problem}}
8579 \newif\ifmultiple\multiplefalse
8580 \DeclareOption{multiple}{\multipletrue}
8581 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8582 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8583 \ProcessOptions
```

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

```
8584 \RequirePackage{keyval}[1997/11/10]
8585 \RequirePackage{problem}
```

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

```
8586 \newcommand\hwexam@assignment@kw{Assignment}
8587 \newcommand\hwexam@given@kw{Given}
8588 \newcommand\hwexam@due@kw{Due}
8589 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8590 \newcommand\hwexam@minutes@kw{minutes}
8591 \newcommand\correction@probs@kw{prob.}
8592 \newcommand\correction@pts@kw{total}
8593 \newcommand\correction@reached@kw{reached}
8594 \newcommand\correction@sum@kw{Sum}
8595 \newcommand\correction@grade@kw{grade}
8596 \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

```

8597 \AddToHook{begindocument}{
8598 \ltx@ifpackageloaded{babel}{
8599 \makeatletter
8600 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8601 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8602 \input{hwexam-ngerman.ldf}
8603 }
8604 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8605 \input{hwexam-finnish.ldf}
8606 }
8607 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8608 \input{hwexam-french.ldf}
8609 }
8610 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8611 \input{hwexam-russian.ldf}
8612 }
8613 \makeatother
8614 }{}
8615 }
8616

```

## 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.

```

8617 \newcounter{assignment}
8618 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8619 \keys_define:nn { hwexam / assignment } {
8620 id .str_set:N = \l_@@_assign_id_str,
8621 number .int_set:N = \l_@@_assign_number_int,
8622 title .tl_set:N = \l_@@_assign_title_tl,
8623 type .tl_set:N = \l_@@_assign_type_tl,
8624 given .tl_set:N = \l_@@_assign_given_tl,
8625 due .tl_set:N = \l_@@_assign_due_tl,
8626 loadmodules .code:n = {
8627 \bool_set_true:N \l_@@_assign_loadmodules_bool
8628 }
8629 }
8630 \cs_new_protected:Nn \_@@_assignment_args:n {
8631 \str_clear:N \l_@@_assign_id_str
8632 \int_set:Nn \l_@@_assign_number_int {-1}
8633 \tl_clear:N \l_@@_assign_title_tl
8634 \tl_clear:N \l_@@_assign_type_tl
8635 \tl_clear:N \l_@@_assign_given_tl
8636 \tl_clear:N \l_@@_assign_due_tl
8637 \bool_set_false:N \l_@@_assign_loadmodules_bool
8638 \keys_set:nn { hwexam / assignment }{ #1 }
8639 }

```

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.

```

8640 \newcommand\given@due[2]{
8641 \bool_lazy_all:nF {
8642 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8643 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8644 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8645 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8646 }{ #1 }
8647
8648 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8649 \tl_if_empty:NF \l_@@_assign_given_tl {
8650 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8651 }
8652 }{
8653 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8654 }
8655
8656 \bool_lazy_or:nnF {
8657 \bool_lazy_and_p:nn {
8658 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8659 }{
8660 \tl_if_empty_p:V \l_@@_assign_due_tl
8661 }
8662 }{
8663 \bool_lazy_and_p:nn {
8664 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8665 }{
8666 \tl_if_empty_p:V \l_@@_assign_due_tl
8667 }
8668 }{ ,~ }
8669
8670 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8671 \tl_if_empty:NF \l_@@_assign_due_tl {
8672 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8673 }
8674 }{
8675 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8676 }
8677
8678 \bool_lazy_all:nF {
8679 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8680 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8681 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8682 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8683 }{ #2 }
8684 }

```

`\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.

```

8685 \newcommand\assignment@title[3]{
8686 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8687 \tl_if_empty:NTF \l_@@_assign_title_tl {
8688 #1
8689 }{
8690 #2\l_@@_assign_title_tl#3
8691 }
8692 }{
8693 #2\l_@@_inclasssign_title_tl#3
8694 }
8695 }

```

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

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

```

8696 \newcommand\assignment@number{
8697 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8698 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8699 \arabic{assignment}
8700 } {
8701 \int_use:N \l_@@_assign_number_int
8702 }
8703 }{
8704 \int_use:N \l_@@_inclasssign_number_int
8705 }
8706 }

```

(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.

```

8707 \newenvironment{assignment}[1][]{
8708 \_@@_assignment_args:n { #1 }
8709 %\sref@target
8710 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8711 \global\stepcounter{assignment}
8712 }{
8713 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8714 }
8715 \setcounter{sproblem}{0}
8716 \renewcommand\prob@label[1]{\assignment@number.##1}
8717 \def\current@section@level{\document@hwexamtype}
8718 %\sref@label{id{\document@hwexamtype \thesection}
8719 \begin{@assignment}
8720 }{
8721 \end{@assignment}
8722 }

```

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

```

8723 \def\ass@title{
8724 {\protect\document@hwexamtype}\arabic{assignment}
8725 \assignment@title{}\;\;{}{}\; -- \given@due{}\;}
8726 }
8727 \ifmultiple
8728 \newenvironment{@assignment}{
8729 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8730 \begin{sfragment}[loadmodules]{\ass@title}
8731 }{
8732 \begin{sfragment}{\ass@title}
8733 }
8734 }{
8735 \end{sfragment}
8736 }

```

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

```

8737 \else
8738 \newenvironment{@assignment}{
8739 \begin{center}\bf
8740 \Large@title\strut\
8741 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;\;{}{}\;
8742 \large\given@due{--\;\;{}{}\;}\;
8743 \end{center}
8744 }{}
8745 \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 `inclasssign` keys after the input.

```

8746 \keys_define:nn { hwexam / inclasssign } {
8747 %id .str_set_x:N = \l_@@_assign_id_str,
8748 number .int_set:N = \l_@@_inclasssign_number_int,
8749 title .tl_set:N = \l_@@_inclasssign_title_tl,
8750 type .tl_set:N = \l_@@_inclasssign_type_tl,
8751 given .tl_set:N = \l_@@_inclasssign_given_tl,
8752 due .tl_set:N = \l_@@_inclasssign_due_tl,
8753 mhrepos .str_set_x:N = \l_@@_inclasssign_mhrepos_str
8754 }
8755 \cs_new_protected:Nn \l_@@_inclasssign_args:n {
8756 \int_set:Nn \l_@@_inclasssign_number_int {-1}
8757 \tl_clear:N \l_@@_inclasssign_title_tl
8758 \tl_clear:N \l_@@_inclasssign_type_tl
8759 \tl_clear:N \l_@@_inclasssign_given_tl
8760 \tl_clear:N \l_@@_inclasssign_due_tl
8761 \str_clear:N \l_@@_inclasssign_mhrepos_str
8762 \keys_set:nn { hwexam / inclasssign } { #1 }
8763 }
8764 \l_@@_inclasssign_args:n {}
8765
8766 \newcommand\inputassignment[2][]{

```

```

8767 \_@@_inclassassignment_args:n { #1 }
8768 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8769 \input{#2}
8770 }{
8771 \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8772 \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8773 }
8774 }
8775 \_@@_inclassassignment_args:n {}
8776 }
8777 \newcommand\includeassignment[2][]{
8778 \newpage
8779 \inputassignment[#1]{#2}
8780 }

```

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

## 42.4 Typesetting Exams

\quizheading

```

8781 \ExplSyntaxOff
8782 \newcommand\quizheading[1]{%
8783 \def\@tas{#1}%
8784 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8785 \ifx\@tas\@empty\else%
8786 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8787 \fi%
8788 }
8789 \ExplSyntaxOn

```

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

\testheading

```

8790
8791 \def\hwexamheader{\input{hwexam-default.header}}
8792
8793 \def\hwexamminutes{
8794 \tl_if_empty:NTF \testheading@duration {
8795 {\testheading@min}~\hwexam@minutes@kw
8796 }{
8797 \testheading@duration
8798 }
8799 }
8800
8801 \keys_define:nn { hwexam / testheading } {
8802 min .tl_set:N = \testheading@min,
8803 duration .tl_set:N = \testheading@duration,
8804 reqpts .tl_set:N = \testheading@reqpts,
8805 tools .tl_set:N = \testheading@tools
8806 }
8807 \cs_new_protected:Nn \_@@_testheading_args:n {
8808 \tl_clear:N \testheading@min
8809 \tl_clear:N \testheading@duration

```



```

8810 \tl_clear:N \testheading@reqpts
8811 \tl_clear:N \testheading@tools
8812 \keys_set:nn { hwexam / testheading }{ #1 }
8813 }
8814 \newenvironment{testheading}[1][ ]{
8815 \_@@_testheading_args:n{ #1 }
8816 \newcount\check@time\check@time=\testheading@min
8817 \advance\check@time by -\theassignment@totalmin
8818 \newif\if@bonuspoints
8819 \tl_if_empty:NTF \testheading@reqpts {
8820 \@bonuspointsfalse
8821 }{
8822 \newcount\bonus@pts
8823 \bonus@pts=\theassignment@totalpts
8824 \advance\bonus@pts by -\testheading@reqpts
8825 \edef\bonus@pts{\the\bonus@pts}
8826 \@bonuspointstrue
8827 }
8828 \edef\check@time{\the\check@time}
8829
8830 \makeatletter\hwexamheader\makeatother
8831 }{
8832 \newpage
8833 }

```

(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.

```

8834 <@@=problems>
8835 \renewcommand\@problem[3]{
8836 \stepcounter{assignment@probs}
8837 \def\__problemspts{#2}
8838 \ifx\__problemspts\empty\else
8839 \addtocounter{assignment@totalpts}{#2}
8840 \fi
8841 \def\__problemsmin{#3}\ifx\__problemsmin\empty\else\addtocounter{assignment@totalmin}{#3}\fi
8842 \xdef\correction@probs{\correction@probs & #1}%
8843 \xdef\correction@pts{\correction@pts & #2}
8844 \xdef\correction@reached{\correction@reached &}
8845 }
8846 <@@=hwexam>

```

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

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

```

8847 \newcounter{assignment@probs}
8848 \newcounter{assignment@totalpts}
8849 \newcounter{assignment@totalmin}
8850 \def\correction@probs{\correction@probs@kw}
8851 \def\correction@pts{\correction@pts@kw}
8852 \def\correction@reached{\correction@reached@kw}
8853 \stepcounter{assignment@probs}
8854 \newcommand\correction@table{

```

```

8855 \resizebox{\textwidth}{!}{%
8856 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8857 &\multicolumn{\theassignment@probs}{c|}{%|
8858 {\footnotesize\correction@forgrading@kw} &\\ \hline
8859 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8860 \correction@pts & \theassignment@totalpts & \\ \hline
8861 \correction@reached & & \[.7cm]\hline
8862 \end{tabular}}
8863 \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).