

# The $\text{\LaTeX}$ 3 Package Collection\*

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

2023-10-13

## Contents

<b>1</b>	<b>Introduction &amp; Setup</b>	<b>7</b>
1.1	What is $\text{\LaTeX}$ ?	7
1.2	The $\text{\LaTeX}$ package	8
1.3	What is MMT?	8
1.4	Math archives and the MathHub Directory	8
1.5	Setting Up the $\text{\LaTeX}$ IDE	9
<b>I</b>	<b>Tutorial</b>	<b>11</b>
<b>2</b>	<b>The Basics</b>	<b>12</b>
2.1	Text symbols	15
2.1.1	Using Modules & Search in the IDE	15
2.2	Symbol References	18
2.3	Modules and Simple Symbol Declarations	21
2.4	Documenting Symbols	24
2.5	Sectioning and Reusing Document Fragments	26
2.6	Building and Exporting HTML	28
<b>3</b>	<b>Mathematical Concepts</b>	<b>31</b>
3.1	Simple Symbol Declarations	31
3.1.1	Semantic Macros and Notations	31
3.1.2	Types and Variables	34
3.1.3	Flexary Macros and Argument Modes	39
3.1.4	Precedences	41
3.1.5	Implicit Arguments	41
3.1.6	Finishing Equality	43
3.1.7	Variable Sequences	43
3.2	Statements	44
3.2.1	Definitions	44
	Semantic Macros in Text Mode	46

---

\*Version 3.4.0 (last revised 2023-10-13)

	Definientia . . . . .	47
	Using Symbols Without Semantic Macros and Exporting Code in Modules . . . . .	48
3.2.2	Assertions . . . . .	49
3.2.3	Proofs . . . . .	52
3.3	Mathematical Structures . . . . .	52
3.3.1	Declaring and Using Structures Instantiating Structures . . . . .	52
3.3.2	Extending Structures and Axioms Conservative Extensions . . . . .	55
3.3.3	Nesting Structures and <code>\this</code> . . . . .	57
3.4	Complex Inheritance and Theory Morphisms . . . . .	59
3.4.1	Glueing Structures Together . . . . .	61
3.4.2	Realizations . . . . .	63
<b>4</b>	<b>Extensions for Education</b>	<b>65</b>
4.1	Slides and Course Notes . . . . .	65
4.2	Problems and Exercises . . . . .	65
4.2.1	Background . . . . .	65
4.2.2	The Package, Options, and Configuration . . . . .	66
4.2.3	(Open) Problems . . . . .	67
4.2.4	Structured Problems . . . . .	68
4.2.5	Single/Multiple Choice Blocks . . . . .	69
4.2.6	Filling-In Concrete Solutions . . . . .	71
4.2.7	Answer Classes . . . . .	72
4.2.8	Including Problems . . . . .	72
4.2.9	Testing and Spacing . . . . .	72
4.3	Homework Assignments and Exams . . . . .	73
4.3.1	Introduction . . . . .	73
4.3.2	Package Options . . . . .	73
4.3.3	Assignments . . . . .	73
4.3.4	Including Assignments . . . . .	74
4.3.5	Typesetting Exams . . . . .	74
<b>II</b>	<b>User Manual</b>	<b>76</b>
<b>5</b>	<b>Basics</b>	<b>77</b>
5.1	Package and Class Options . . . . .	77
5.2	Math Archives and the MathHub Directory . . . . .	78
5.2.1	The Structure of Math Archives . . . . .	79
5.2.2	MANIFEST.MF-Files . . . . .	79
5.3	The <code>lib</code> -Directory . . . . .	80
5.4	Basic Macros . . . . .	81
<b>6</b>	<b>Document Features</b>	<b>82</b>
6.1	Document Fragments . . . . .	82
6.2	Using and Referencing Document Fragments . . . . .	83
6.3	Cross-Document References . . . . .	83

<b>7</b>	<b>Modules and Symbols</b>	<b>86</b>
7.1	Modules	86
7.1.1	Signature Modules, Languages, and Multilinguality	87
7.2	Symbol Declarations	87
7.2.1	Returns	88
7.3	Referencing Symbols	88
7.4	Notations and Semantic Macros	90
7.4.1	Precedences and Bracketing	91
7.4.2	Notations for Argument Sequences	92
7.4.3	Semantic Macros	92
7.5	Simple Inheritance	93
7.6	Variables and Sequences	95
7.7	Structures	96
7.7.1	Semantic Macros for Structures	96
<b>8</b>	<b>Statements</b>	<b>99</b>
8.1	More on Definitions	100
8.2	More on Assertions	101
<b>9</b>	<b>Customizing Typesetting</b>	<b>102</b>
9.1	Highlighting Symbol References	102
9.2	Styling Environments and Macros	103
9.3	Custom CSS for Environments	104
<b>10</b>	<b>Additional Packages</b>	<b>105</b>
10.1	NotesSlides Manual	105
10.1.1	Introduction	105
10.1.2	Package Options	105
10.1.3	Notes and Slides	106
10.1.4	Customizing Header and Footer Lines	107
10.1.5	Frame Images	107
10.1.6	Ending Documents Prematurely	108
10.1.7	Global Document Variables	108
10.1.8	Excursions	109
10.2	Problem Manual	110
10.2.1	Introduction	110
10.2.2	Problems and Solutions	110
10.2.3	Markup for Added-Value Services	112
	Multiple Choice Blocks	112
	Filling-In Concrete Solutions	113
10.2.4	Including Problems	114
10.2.5	Testing and Spacing	115
10.3	HWExam Manual	115
10.3.1	Introduction	115
10.3.2	Package Options	115
10.3.3	Assignments	116
10.3.4	Including Assignments	116
10.3.5	Typesetting Exams	116
10.4	Tikzinput Manual	117

<b>III</b>	<b>Documentation</b>	<b>119</b>
<b>11</b>	<b>STEX Developer Manual</b>	<b>120</b>
11.1	Documents	121
11.2	Modules	121
11.3	Symbols	123
11.4	Notations	125
11.5	Structural Features	128
11.6	Imports and Morphisms	130
11.7	Expressions and Semantic Macros	131
11.8	Optional (Key-Value) Argument Handling	132
11.9	Stylable Commands and Environments	133
11.10	Math Archives	134
11.11	SMS-Mode	135
11.11.1	Second Pass	135
11.11.2	First Pass	136
11.12	Strings, File Paths, URIs	136
11.12.1	File Paths	137
	File Path Constants and Variables	138
11.12.2	URIs	138
	URI Constants and Variables	139
11.13	Language Handling	139
11.14	Inserting Annotations	140
11.14.1	Backend macros	140
11.15	Persisting Content from Math Archives in sms-Files	141
11.16	Utility Methods	141
11.16.1	Group-like Behaviours	142
<b>12</b>	<b>Additional Packages</b>	<b>144</b>
12.1	NotesSlides Documentation	144
12.2	Problem Documentation	144
12.3	HWExam Documentation	144
12.4	Tikzinput Documentation	144
<b>IV</b>	<b>Implementation</b>	<b>145</b>
<b>13</b>	<b>The STEX Implementation</b>	<b>146</b>
13.1	Setting up	146
13.2	Utilities	147
13.2.1	Calling kpsewhich and Environment Variables	147
13.2.2	Logging	148
13.2.3	Languages	149
13.2.4	Group-like Behaviours	151
13.2.5	HTML Annotations	152
13.2.6	Auxiliary Methods	154
13.2.7	Persistence	160
13.2.8	Files, Paths and URIs	161
13.2.9	File Hooks	169
13.3	Math Archives	170

13.4	Documents	174
13.4.1	Title	174
13.4.2	Sectioning	175
13.4.3	References	179
13.4.4	Inputs	187
13.5	SMS Mode	192
13.6	Modules	196
13.6.1	The smodule-environment	196
13.6.2	Structural Features	207
13.7	Inheritance	212
13.7.1	<code>\importmodule/\usemodule</code>	212
13.7.2	Theory Morphisms	218
13.8	Symbols	224
13.8.1	Declarations	224
13.8.2	Notations	233
	a/B-mode argument handling	244
13.8.3	Variables	246
13.8.4	Sequences	250
13.8.5	Expressions	254
	Invoking Semantic Macros	255
	Argument Handling and Annotating	261
	Term HTML Annotations	266
	Automated Bracketing	268
	Symname and Variants	269
	Highlighting	272
13.9	Mathematical Structures	273
13.10	Statements	286
13.11	Proofs	292
13.12	Metatheory	300
13.13	MMT Interfaces	302
<b>14</b>	<b>Additional Packages</b>	<b>306</b>
14.1	Implementation: The notesslides Package	306
14.1.1	Class and Package Options	306
14.1.2	Notes and Slides	310
14.1.3	Environment and Macro Patches	313
14.1.4	Styling Across Notes/Slides	315
14.1.5	Beamer Compatibility	315
14.1.6	TODO Excursions	316
14.2	Implementation: The problem Package	317
14.2.1	Package Options	317
14.2.2	Problems and Solutions	318
14.3	Implementation: The hwexam Package	333
14.3.1	Package Options	333
14.3.2	Assignments	334
14.3.3	Leftovers	337
14.4	Tikzinput Implementation	337



$\text{sTeX}$  is – by now – relatively stable and ready to use for the general public. However, it is also actively being developed further and subject to ongoing research. Some of the features described in here might not fully work as expected, some are still experimental, there might occasionally be cryptic error messages, and in general bugs are expected.

We welcome all kinds of issues you might encounter at <https://github.com/slatex/sTeX>.

*If you have questions or problems with  $\text{sTeX}$ , you can talk to us directly at <https://matrix.to/#/#stex:fau.de>.*

# Chapter 1

## Introduction & Setup

### 1.1 What is sTeX?

sTeX is a system for generating human-oriented documents in either PDF or HTML format, augmented with computer-actionable semantic information (conceptually) based on the OMDoc format and ontology.

At its core is the sTeX package for L<sup>A</sup>T<sub>E</sub>X, that allows for semantically marking up document fragments; in particular concepts, formulae and mathematical statements (such as definitions, theorems and proofs). Running pdf<sub>l</sub>atex over sTeX-annotated documents formats them into normal-looking PDF.

But sTeX also comes with a conversion pipeline into semantically annotated HTML, which can host semantic added-value services that make the documents *active* (i.e. interactive and user-adaptive) – essentially turning L<sup>A</sup>T<sub>E</sub>X into a document format for (mathematical) knowledge management (MKM).

The sTeX system consists of the following components:

- The sTeX package,
- the RuSTeX system for converting L<sup>A</sup>T<sub>E</sub>X documents to (semantically enriched) HTML,
- the MMT system for advanced knowledge management service and semantically informed backend for hosting the HTML with integrated added-value services,
- a collection of math archives of sTeX document fragments and symbols for reuse, available of mathhub.info, and
- the sTeX IDE: A VS Code extension that, besides the usual IDE functionalities like syntax highlighting, integrates RuSTeX and MMT and allows for accessing the public math archives on mathhub.info to make the entire toolchain easily accessible to authors.

If PDF is all you are interested in, the sTeX package is all you *need*. Either way, however, we recommend using the sTeX IDE – it very much helps with semantically annotating your L<sup>A</sup>T<sub>E</sub>X documents.

## 1.2 The `sTeX` package

The `sTeX` package extends `LATEX` with:

- A mechanism to declare `symbols` – concepts, functions, relations, variables, etc., which can be used and referenced in text or via `semantic macros` in mathematical formulae,
- a `module system` based on logical identifiers – `modules` bundle declarations, definitions, theorems, document snippets, and `symbols` for reuse, and
- an analogous organizational structure for developing documents modularly from individual fragments and sections.

The `sTeX` package has been designed to have minimal impact on other `packages` and `document classes`, or the actual document layout – formatting of semantic `environments`, `symbol` references and `semantic macros` can be fully customized.

## 1.3 What is `Mmt`?

`MMT` is a `software system` and `Scala` API for generic knowledge management services. It is based on a version of the `OMDOC` ontology and `document format` (`MMT/OMDoc`).

Among the services `MMT` provides are compiling, building, converting and managing libraries, a built-in web-server for browsing content, various algorithms for generic computation, checking and translating expressions, and querying.

## 1.4 `Math archives` and the `MathHub` Directory

To make the most of `sTeX`, it is strongly encouraged to follow a workflow of small document fragments and `modules` to maximize reuse.

One considerable weakness of `LATEX` is the way source files are referenced: they need to be either in a `texmf` directory, or else be referenced via file paths relative to the main `.tex`-file being compiled. This is highly inconvenient if we want to collaboratively develop many highly interrelated document fragments.

`sTeX` therefore adds an organizational layer on top of `LATEX`'s: `math archives` stored in a fixed `MathHub` directory anywhere on your hard drive. Referencing source files and `modules` is then done relative to the containing `math archive`, and is thus *independent* of user's individual setups or the current `.tex`-file.

The drawback of this approach is that `sTeX` needs to know the location of your `MathHub` directory. There are multiple ways to achieve that, but the simplest and recommended approach is to set an environment variable: Simply create a new directory `<path>/MathHub` somewhere on your hard drive and set the environment variable `MATHHUB` as the path to this new directory.

Alternatively, you can let the `sTeX IDE` do the work for you (see [section 1.5](#)).

For more on `math archives`, see [section 5.2](#).



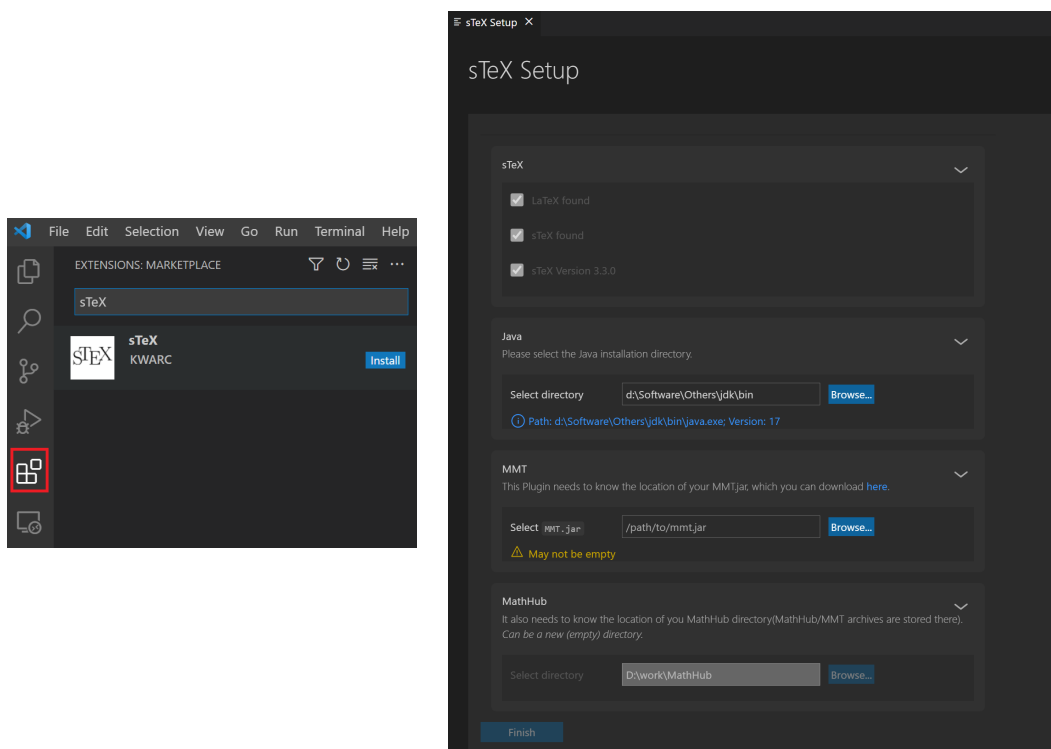


Figure 1: Installing the  $sTeX$  IDE

## 1.5 Setting Up the $sTeX$ IDE

$sTeX$  is based on  $L^ATeX$ , and adds additional layers of presentational and functional markup to it. As a consequence the source files of  $sTeX$  documents look quite different from the resulting  $XHTML$  and  $PDF$  documents. Thus the best way of interacting the  $sTeX$  document collections is via an *integrated development environment (IDE)*. In this tutorial we will use the  $sTeX$  plugin for the  $VS$  Code, which you should set up as a first step (this also sets up the necessary auxiliary software).

Setting up  $sTeX$  with the dedicated *IDE* is easy:

1. Download and install  $VS$  Code here: <https://code.visualstudio.com/download>
2. Start  $VS$  Code and navigate to the *Extensions*-tab on the left. Here you can search for Extensions in the  $VS$  Code marketplace. Look for the  $sTeX$  extension by *KWARC*, as in **Figure 1** on the left.
3. Having done so, upon opening any folder in  $VS$  Code containing a  $.tex$ -file the setup window will pop up, as in **Figure 1** on the right.

The *IDE* will attempt to determine your  $Java$  installation and your  $MathHub$  directory (if set via an environment variable). Alternatively, you can set the latter now.

4. Download the  $MMT$   $.jar$ -file at the link provided in the setup and select it. The *IDE* should then be able to determine your  $MMT$  version.

And that's it. Click on *Finish* and your setup is finished. The extension will start and download [R<sub>U</sub>S<sub>T</sub>E<sub>X</sub>](#) and some fundamental [math archives](#) for you automatically (an internet connection is required when finishing the setup).

# Part I

## Tutorial

*The dynamic [HTML](#) version of this part can be found at*  
*<https://stexmt.mathhub.info/sTeX/fullhtml?archive=sTeX/Documentation&filepath=tutorial.en.xhtml>*

[sTeX](#) is a system for generating human-oriented documents in either [PDF](#) or [HTML](#) format, augmented with computer-actionable semantic information (conceptually) based on the [OMDoc](#) format and ontology.

In this part, we will give a broad but shallow introduction to [sTeX](#), and what you can get out of it. Additionally, this serves as an introduction to the [sTeX IDE](#), and we consequently assume that you have that one set up, as described in [section 1.5](#).

Note that in [PDFs](#), the specific highlighting of semantically annotated text is fully customizable (see [chapter 9](#)). In this document, we use [this highlighting](#) for [notation](#) components, [this highlighting](#) for [symbol](#) references, [this highlighting](#) for (local) [variables](#) and [this highlighting](#) for definienda; i.e. new concepts being introduced.

## Chapter 2

# The Basics

This document itself uses [S<sub>T</sub>E<sub>X</sub>](#) and serves as a direct example for the following. You can download its source files, the generated [PDF](#) files, and the generated [HTML](#) documents directly from within the [IDE](#), by navigating to the [S<sub>T</sub>E<sub>X</sub>](#) tab in the menu on the left and finding `sTeX/Documentation` in the list of [math archives](#) and clicking the small “Install”-button next to it, see the screenshot on the left of [Figure 2](#).

Once downloading is finished (this may take a while since dependencies are also downloaded), you can then browse the `.tex`-files in `sTeX/Documentation` directly from the [math archives](#) panel in the [S<sub>T</sub>E<sub>X</sub>](#) tab, as you can see in the right screenshot in [Figure 2](#).

For example, you can now navigate to the file `tutorial/intro.en` to see the sources of this very chapter.

As a first example, consider the following document fragment from [section 1.1](#):

[S<sub>T</sub>E<sub>X</sub>](#) is a system for generating human-oriented documents in either [PDF](#) or [HTML](#) format, augmented with computer-actionable semantic information (conceptually based on the [OMDoc](#) format and ontology).

If you were to look at the generated [HTML](#) from this fragment, you could hover over the highlighted words ([S<sub>T</sub>E<sub>X</sub>](#), [PDF](#), [HTML](#), [OMDoc](#)) and get a little popup with their definitions ([Figure 3](#)). Neat, huh?

Here, in the [PDF](#), hovering will only show you a unique identifier ([MMT-URI](#)) for the word, and link to a definition on the web. Still useful, but not quite as neat, of course.

A plain [L<sup>A</sup>T<sub>E</sub>X](#)-version of the above document fragment, without any [S<sub>T</sub>E<sub>X</sub>](#) markup, could look like this:

### **Example 1**

Input:

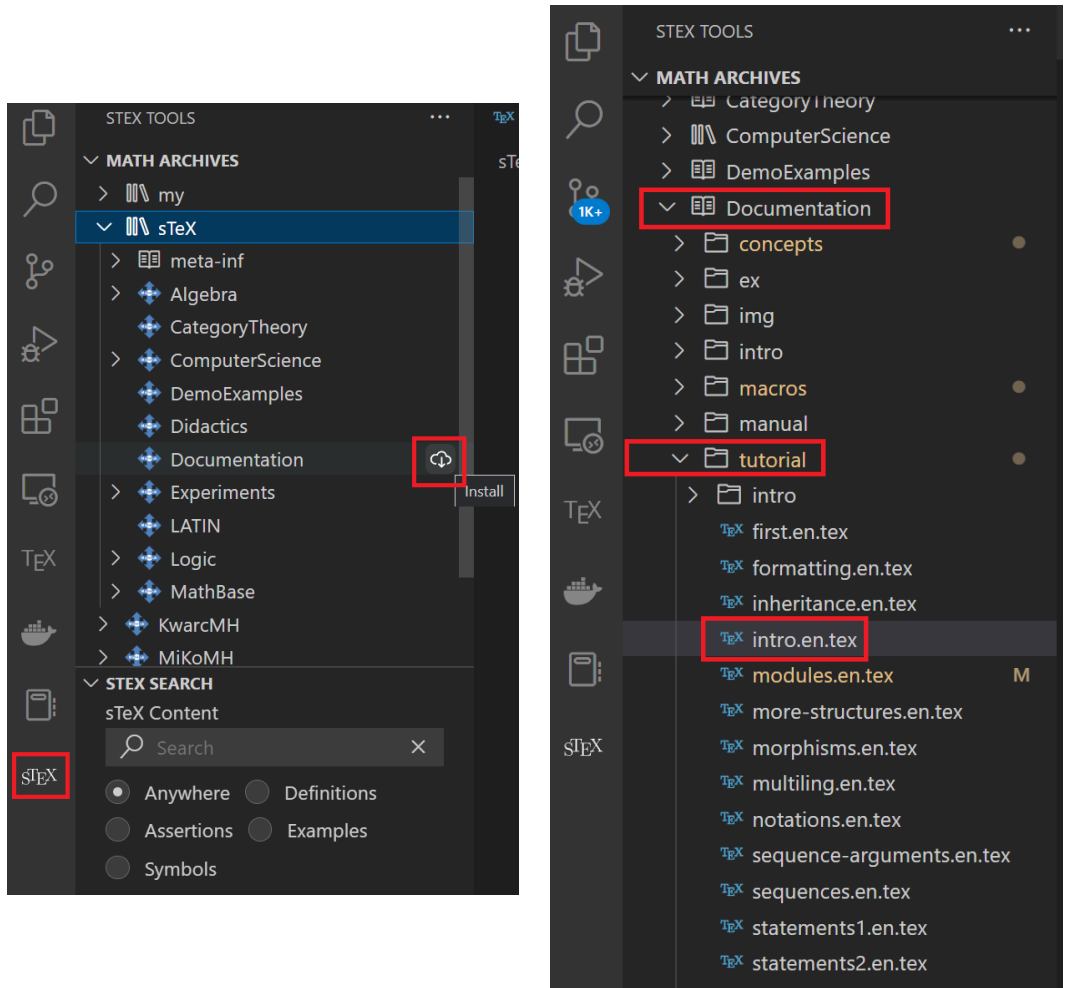


Figure 2: Installing Math Archives

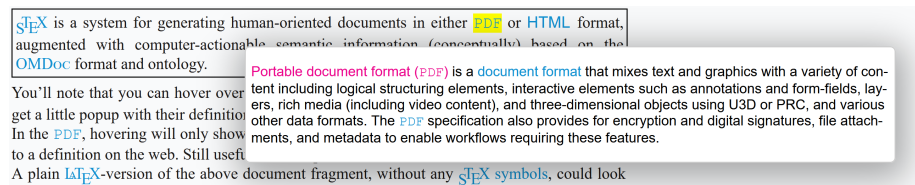


Figure 3: Definition on Hover

```

File [sTeX/Documentation]tutorial/intro/intro1plain.en.tex
1 \documentclass{article}
2 \usepackage{stex-logo}
3 \begin{document}
4
5   \sTeX{} is a system for generating human-oriented documents
6   in either \textsf{PDF} or \textsf{HTML} format, augmented
7   with computer-actionable semantic information (conceptually)
8   based on the \textsc{OMDoc} format and ontology.
9
10 \end{document}

```

Output:

```

\sTeX is a system for generating human-oriented documents in either PDF or HTML
format, augmented with computer-actionable semantic information (conceptually) based
on the OMDoc format and ontology.

```

(Examples like the one above always show the file the source code is in, so if you have downloaded the `sTeX/Documentation` [math archive](#) you can toy around with it yourself)

If you save a file in the IDE (regardless of whether it has unsaved changes), a preview window will pop up, showing you the `HTML` generated from the `.tex`-file; see (Figure 4).

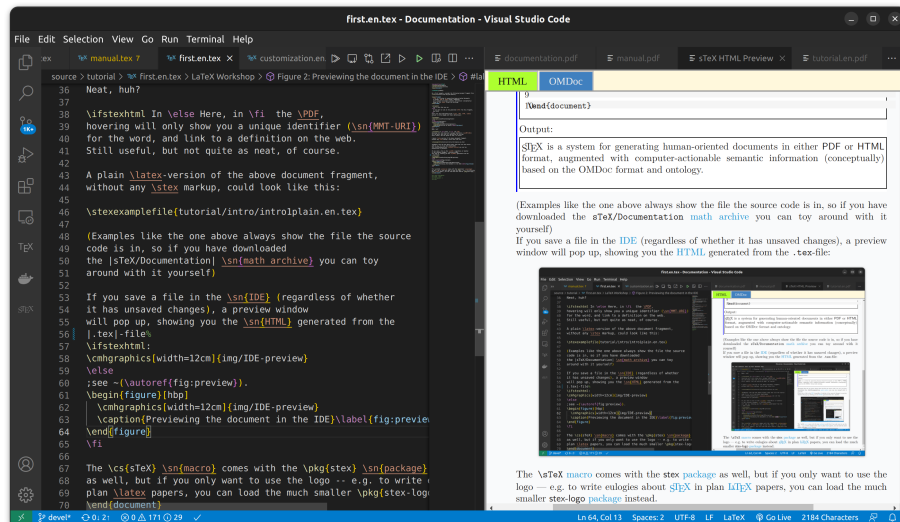


Figure 4: Previewing the document in the IDE

The `\sTeX` macro comes with the `stex` package as well, but if you only want to use the logo – e.g. to write eulogies about `sTeX` in plain `LATEX` papers, you can load the much smaller `stex-logo` package instead.

## 2.1 Text symbols

The most central concept behind  $\TeX$  is that of a *symbol*:

$\TeX$  A **symbol** is a *named* concept that can be defined, documented and referenced. Examples for **symbols** are mathematical constants, functions, theorems, statements, principles – anything that has a (somewhat) precise meaning and can be referenced by name can be a **symbol**.

Before we explain how we can declare new **symbols** and associate them with definitions, **notations** and all that, let’s assume an ideal world in which others have done that job already for us – after all,  $\TeX$  is all about *reuse*, and naturally, there are  $\TeX$  **symbols** for all of the above already. Let’s start with the one for  $\TeX$  itself:

### 2.1.1 Using Modules & Search in the IDE

In the **VS Code IDE**, navigate to the  $\TeX$ -tab on the left. In the search panel, select the “Symbols” radio button and search for “ $\TeX$ ”. The second search result should be what we’re looking for (**Figure 5**).

Search results are grouped into *local* and *remote* results. Local ones are the ones you already have in your local **MathHub** directory; remote ones you can download directly from within the **IDE**.

You can click the preview button to see the generated **HTML** for the document – the resulting window that pops up also has an **OMDoc** tab you can select, which (among other things) shows you the **semantic macros** provided by the respective **module**: In this case, it tells us that there is a *text symbol* named “ $\TeX$ ” with **semantic macro** `\stex` in the **module** `mod/systems/tex?stex` that is in the `\TeX/ComputerScience/Software` archive. It produces the presentation “ $\TeX$ ” as we want (**Figure 6**).

$\TeX$  A **text symbol** is a **symbol** `foo` with an associated **semantic macro** `\foo`. The **macro** `\foo` is allowed in text or math mode and produces a predefined piece of text output annotated with `foo`.  
The variant `\fooname` produces the same output without annotation.

If we want to use the  $\TeX$  **symbol** in a document – which we have open in the **IDE** – we simply click on the **use** button, and the **IDE** will automatically insert the line `\usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?stex}`, making all **symbols** in that **module** available to use – in particular, we can now use the `\stex` **semantic macro** instead of the plain, non-semantic `\TeX` **macro** – that is, of course, after we include the `stex` **package** first.

$\TeX$  The `\usemodule` **macro** takes as *optional* argument the name of a **math archive**, and as a regular argument the path to an  $\TeX$  **module** (see **section 7.5**).

Analogously, we can also search for the **PDF**, **HTML** and **OMDoc** **symbols**, all of which are also **text symbols** and have the associated **semantic macros** `\PDF`, `\HTML` and `\omdoc`; the document should thus look like this:

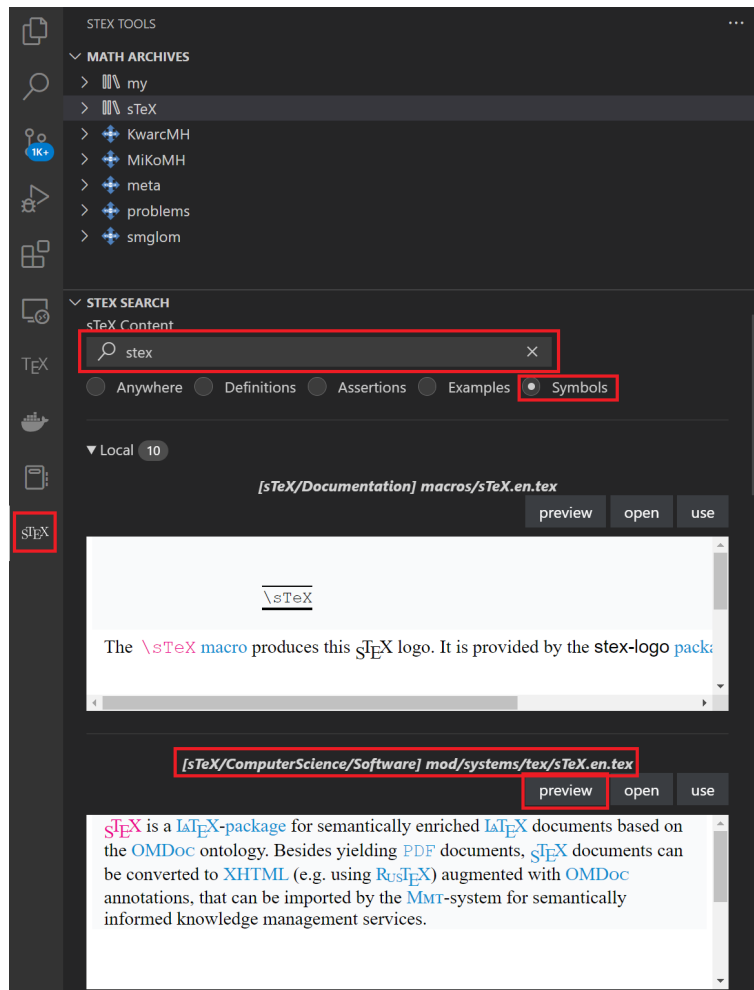


Figure 5: Search in the sTeX IDE



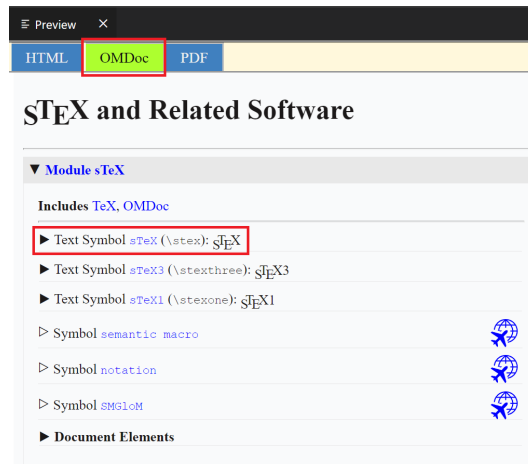


Figure 6: OMDoc Preview

## Example 2

Input:

```

File [sTeX/Documentation]tutorial/intro/intro1stex.en.tex
1 \documentclass{article}
2 \usepackage{stex}
3 \begin{document}
4   \usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}
5   \usemodule[sTeX/ComputerScience/Software]{mod/formats?PDF}
6   \usemodule[sTeX/ComputerScience/Software]{mod/formats?HTML}
7   \usemodule[sTeX/ComputerScience/Software]{mod/formats?OMDoc}
8
9   \stex is a system for generating human-oriented documents
10  in either \PDF or \HTML format, augmented
11  with computer-actionable semantic information (conceptually)
12  based on the \omdoc format and ontology.
13 \end{document}

```

Output:

```

sTeX is a system for generating human-oriented documents in either PDF or HTML
format, augmented with computer-actionable semantic information (conceptually) based
on the OMDoc format and ontology.

```

Now, our generated [HTML](#) looks a lot more interesting, with highlighting, pop-ups on hover and all that. Notably however, if we compile the file with `pdflatex`, it looks pretty much exactly as before.

That's because we haven't told [sTeX](#) what to do with semantic annotations yet – and by default, it does not do anything fancy, except for wrapping them in an `\emph`. We can customize how we want [sTeX](#) to highlight various semantic text fragments (see [chapter 9](#)). A default highlighting schema is provided in the `stex-highlighting` package – including that will

```
http://mathhub.info/sTeX/ComputerScience/Software]
\usemodule[sTeX/ComputerScience/Software]{mod/formats?OMDoc}
Redundant Import
View Problem (Alt+F8) No quick fixes available
```

Figure 7: Redundant Imports

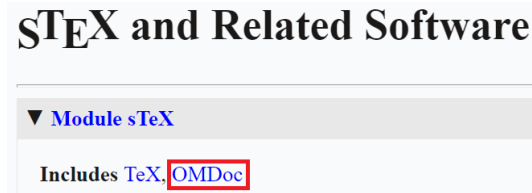


Figure 8: Includes in the OMDoc Preview

- highlight semantically annotated text in [this color](#),
- show the [MMT-URI](#) of the corresponding [symbol](#) in a tooltip on hovering over the text,
- make the text link to the place the [symbol](#) is being defined in the current document (if it is), or, alternatively,
- make it link to an external resource, if one is known. In our case, they link to [stexmt.mathhub.info/:sTeX](http://stexmt.mathhub.info/:sTeX), where the [HTML](#) for all the [symbols](#) we use in this document are hosted.

Note that in the IDE, the `\usemodule`-statement for `OMDoc` is underlined in blue (Figure 7) – VS Code is letting us know, that this `\usemodule` statement is *redundant*. That is because the `sTeX` module we imported earlier already imports the `OMDoc` module; as such we have all `macros` therein available already. If we look at the `sTeX` module in the VS Code preview window again, we can see that (Figure 8).

We can consequently safely delete the `\usemodule` again.

## 2.2 Symbol References

Let's continue with the next paragraph of [section 1.1](#); for now unannotated:

### Example 3

Input:

```

File [sTeX/Documentation]tutorial/intro/intro2plain.en.tex
1 \documentclass{article}
2 \usepackage{stex}
3 \begin{document}
4
5   At its core is the \sTeX{} package for \LaTeX, that allows for
6   semantically marking up document fragments; in particular
7   concepts, formulae and mathematical statements (such as
8   definitions, theorems and proofs). Running \texttt{pdflatex}
9   over \sTeX-annotated documents formats them into normal-looking
10  \textsf{PDF}.
11
12 \end{document}

```

Output:

```

At its core is the sTeX package for LATEX, that allows for semantically marking up
document fragments; in particular concepts, formulae and mathematical statements
(such as definitions, theorems and proofs). Running pdflatex over sTeX-annotated
documents formats them into normal-looking PDF.

```

We already know how to annotate “[sTeX](#)” and “[PDF](#)”; and if we use the search field in the [IDE](#) again, we can also find a [text symbol](#) for “[LATEX](#)”. But if we look at the documentation, we will note that *more* is highlighted:

```

At its core is the sTeX package for LATEX, that allows for semantically marking up
document fragments; in particular concepts, formulae and mathematical statements
(such as definitions, theorems and proofs). Running pdflatex over sTeX-annotated
documents formats them into normal-looking PDF.

```

The “[package](#)”-[symbol](#) can be found in the [LATEX](#) module too, and searching for the keywords “[formula](#)” and “[mathematics](#)” will yield the symbols “[well-formed formula](#)” and “[mathematics](#)”, but they are not *text symbols* and “[mathematics](#)” and “[package](#)” do not even have a [semantic macro](#) – and the one for “[well-formed formula](#)” would not work outside of math mode.

[Text symbols](#) are special in that way – they are intended for [symbols](#) that have a specific formatting associated (such as [LATEX](#), [OMDOC](#), or [HTML](#), which we prefer to typeset as sans serif). For those settings, it makes sense to associate that formatting with a [semantic macro](#) that does the typesetting for us.

[Symbols without a text macro](#) can be referenced with the [\symname](#) macro: [\symname{package}](#) prints the *name* of the “[package](#)”-[symbol](#) and annotates it accordingly, without any special formatting – in particular it is compatible with being in [\emph](#), [\textbf](#) and similar [macros](#). That takes care of *one* of the missing annotations.

More generally, the [\symref](#) macro can be used to annotate arbitrary text with a [symbol](#): [\symref{mathematics}{mathematical}](#) associates the text [mathematical](#) with the [symbol](#) “[mathematics](#)”; thus, we get “[mathematical](#)” and similarly “[formulae](#)”.

STEX

In general, any [macro](#) that expects a [symbol](#) name can be given either

1. the *name* of the [symbol](#),

2. the name of its [semantic macro](#),
3. or any suffix of its [MMT-URI](#) containing at least the [module](#) name.

**STEX**

The second option is often short – and therefore convenient to write; for example, to achieve “[formulae](#)”, we can also write `\symref{wff}{formulae}`, since `\wff` is the [semantic macro](#) for “[well-formed formula](#)”.

The third option allows for distinguishing between multiple [symbols](#) with the same name – the [IDE](#) can help in the latter case, by underlining ambiguous [symbol](#) references in yellow, and offering the [Quick Fix](#) functionality to let you select and autocomplete the specific [symbol](#) you want to reference.

Since `\symname` and `\symref` are a lot to type for something that should ideally be used as often as possible, the [macros](#) `\sn` and `\sr` exist as well and behave exactly the same way. We also provide some convenience abbreviations for `\sn`; namely `\Sn` (capitalizes the first letter of the [symbol](#) name), `\sns` (adds an “s” at the end, for the most common pluralization of a name), and `\Sns` (both).

Using all of the above, our annotated fragment now looks like this:

#### Example 4

Input:

```

File [sTeX/Documentation]tutorial/intro/intro2stex.en.tex
5 \usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}
6 \usemodule[sTeX/Logic/General]{mod/syntax?Formula}
7 \usemodule[sTeX/MathBase/General]{mod?Mathematics}
8 \usemodule[sTeX/ComputerScience/Software]{mod/formats?PDF}
9
10 At its core is the \stex \sn{package} for \latex, that allows for
11 semantically marking up document fragments; in particular
12 concepts, \sr{wff}{formulae} and \sr{mathematics}{mathematical}
13 statements (such as definitions, theorems and proofs). Running
14 \texttt{pdflatex} over \stex-annotated documents formats them
15 into normal-looking \PDF.

```

Output:

At its core is the [STEX](#) package for [L<sup>A</sup>T<sub>E</sub>X](#), that allows for semantically marking up document fragments; in particular concepts, [formulae](#) and [mathematical](#) statements (such as definitions, theorems and proofs). Running `pdflatex` over [STEX](#)-annotated documents formats them into normal-looking [PDF](#).

There’s only one problem: *the document does not compile*, with an error `Undefined control sequence`. The reason being that *some* [macro](#) in the [module](#) `Formula` uses the `\text` [macro](#). We can fix that by using the [amsfonts](#) package of course, but this points to a more general problem; namely that [modules](#) can make use of various [L<sup>A</sup>T<sub>E</sub>X](#) packages for typesetting [symbols](#).

Good practice suggests putting those packages into a *prelude* per [math archive](#), which we can then import from anywhere, using the `\libinput` [macro](#). For more on that, see [section 5.3](#).

For now, suffice it to say that we can import all [packages](#) required for the [module](#) `Formula` from the [math archive](#) `sTeX/Logic/General` by adding the line

```
\libinput[sTeX/Logic/General]{preamble}
```

before the `\begin{document}`.

## 2.3 Modules and Simple Symbol Declarations

Consider again the first two paragraphs of [section 1.1](#):

$\text{\sTeX}$  is a system for generating human-oriented documents in either [PDF](#) or [HTML](#) format, augmented with computer-actionable semantic information (conceptually) based on the [OMDoc](#) format and ontology.

At its core is the  [\$\text{\sTeX}\$  package](#) for  [\$\text{\LaTeX}\$](#) , that allows for semantically marking up document fragments; in particular concepts, [formulae](#) and [mathematical](#) statements (such as definitions, theorems and proofs). Running `pdflatex` over  $\text{\sTeX}$ -annotated documents formats them into normal-looking [PDF](#).

Firstly, note that the first paragraph would be perfectly suitable to serve as a pop-up definition on hover for the  $\text{\sTeX}$  symbol. Secondly, what if all the [symbols](#) used in the above *didn't* already exist?

In this section, we will describe how to make your own [symbols](#) and collect them as reusable fragments in [modules](#) and [math archives](#) from scratch.

We start by creating a new [math archive](#). In the [IDE](#), switch to the  $\text{\sTeX}$ -tab on the left and click the “[New  \$\text{\sTeX}\$  Archive](#)” button ([Figure 9](#)). You will then be asked for the

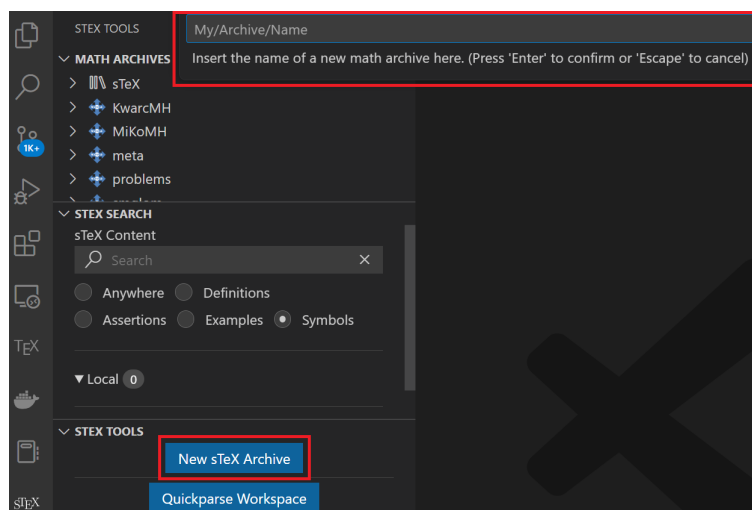


Figure 9: New [Math Archive](#) in the [IDE](#)

name of the [archive](#), a [namespace](#) for its content, and a url-base, where the content is supposedly going to end up online. You can safely keep the defaults for the latter two. In the following, we assume that your archive is named `my/archive`.

The [IDE](#) will then create the following files and directories in your MathHub directory:

```

- my
  - archive
    - lib
      - preamble.tex
    - META-INF
      - MANIFEST.MF
    - source
      - helloworld.tex

```

... and open the file `helloworld.tex` with the content

```

1 \documentclass{stex}
2 \libinput{preamble}
3 \begin{document}
4 % A first sTeX document
5 \end{document}

```

You can now reference any newly created content in you new `archive` using for example `\usemodule[my/archive]{...}`.

Let's start with the "`LATEX`" `symbol`. Rename the file `helloworld.tex` to something more meaningful, for example `latex.en.tex` – the `.en` will be picked up on by `sTeX` to signify that the fragment will be in *english* (see [subsection 7.1.1](#)).

What we want to achieve in this file is the following:

`TEX` is a document typesetting software developed by Donald Knuth, with a focus on mathematical formulae. It is based on a powerful and extensible `macro` expansion engine.

`LATEX` is a (nowadays) default collection of `TEX` `macros` developed by Leslie Lamport. Among other things, `LATEX` introduces `environments`, a distinction between preamble and document content, `packages` to bundle and distribute `macro` definitions, and `document classes`: special `packages` that govern the global layout of a document.

In particular, in the `HTML` the two paragraphs above should be shown when hovering over the `symbols` they define (as indicated by the magenta definiendum highlighting). So we need `symbols` and `semantic macros`, for: `TEX`, `macro`, `LATEX`, `environment`, `package` and `document class`.

`Symbol` declarations are only allowed within `modules`:

`STEX` A `module` is a *named* block that bundles `symbol` declarations for subsequent reuse. A `module` is introduced with the `smodule`-environment.

Let's name our `module` `LaTeX`. We then wrap the contents of our document in a `smodule` environment:

```

\begin{document}
  \begin{smodule}{LaTeX}
    ...
  \end{smodule}
\end{document}

```

Note, that the `IDE` immediately picks up on this and displays the full `MMT-URI` of our new `module` over the `\begin{smodule}{LaTeX}` ([Figure 10](#)) –

```
http://mathhub.info/my/archive/latex?LaTeX
\begin{smodule}{LaTeX}
```


Figure 10: VS Code Code Lense

From this, we can glimpse that the namespace of the module is `http://mathhub.info/my/archive/latex`. This implies, that to use the module somewhere else, we will have to type `\usemodule[my/archive]{latex?LaTeX}` – the `latex`-part pointing to the *file* and `LaTeX` referring to the actual module.

If we rename the file to `LaTeX.en.tex`, we notice that the namespace changes to `http://mathhub.info/my/archive`, allowing us to now use it with `\usemodule[my/archive]{LaTeX}` directly. That’s because the module name `LaTeX` and the file name `LaTeX` match now (see section 7.5, Figure 11).

```
http://mathhub.info/my/archive?LaTeX
\begin{smodule}{LaTeX}
```

Figure 11: VS Code Code Lense



Note that “`LaTeX`” and “`latex`” only differ in capitalization – if your file system is case-insensitive (as e.g. MacOS’s was until quite recently), this distinction gets murky, but remains very important especially if you want to share your [math archive](#) with others!  
It is therefore *highly recommended* to treat file names as case-sensitive either way.

Within the module, we can now declare new symbols using the `\symdecl`-macro. We start with those that are not text symbols:

```
\symdecl*{macro}
\symdecl*{environment}
\symdecl*{package}
\symdecl*{document class}
```

The `*` after the `\symdecl` indicates, that we do not want a semantic macro for the symbol – otherwise, it would generate one with the same name as the symbol itself and “pollute the macro space”, so to speak.

The symbols `\TeX` and `\LaTeX`, however, have a definite way of being typeset associated with them, which can be produced using the standard `\TeX` and `\LaTeX` macros. So let’s make them text symbols, using the `\textsymdecl` macro:

```
\textsymdecl{tex}{\TeX}
\textsymdecl{latex}{\LaTeX}
```

The first argument being the name of the generated macro (i.e. `\tex` and `\latex`) and the second one specifying the output to produce.

## 2.4 Documenting Symbols

We can now use the two new macros, `\symname/\sn`, `\symref/\sr` etc. to mark up the above two paragraphs. But the IDE also makes us aware of the symbols not yet being documented, via squiggly blue lines (Figure 12).

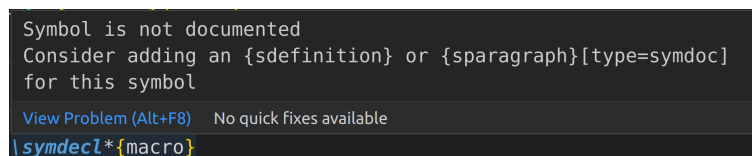


Figure 12: Undocumented Symbols

Among other things, this means that the system does not yet know what to show a reader when hovering over the symbol in the HTML. The IDE also recommends two ways to fix that: The `sdefinition` or `sparagraph` environments.

Ignoring the former for now, which is more useful for mathematical concepts, we can use the following to mark up the first paragraph:

```
\begin{sparagraph}[style=symdoc,for={tex,macro}]
  \tex is a document typesetting
  software developed by Donald Knuth, with a focus on
  mathematical formulae. It is based on a powerful
  and extensible \sn{macro} expansion engine.
\end{sparagraph}
```

In general, the `sparagraph` environment can be used to mark up arbitrary paragraphs semantically, but the `style=symdoc` option tells `STEX` to use this paragraph as a documentation for the symbols provided in the `for=` option. And indeed, doing so makes the squiggly blue lines in the IDE under `\textsymdecl{tex}{TeX}` and `\symdecl*{macro}` disappear.

We just used the semantic macro `\stex` and the `\sn` macro to mark up the fragment – but we can do better. Both concepts are being *introduced* in the above paragraph, and we can let `STEX` know that that is the case:

Within an `sparagraph` environment with `style=symdoc` (or an `sdefinition` environment), we can mark up *definienda*, meaning the terms *being defined*, explicitly. Analogously to `\symname` and `\symref`, we have the macros `\definame` and `\definiendum` for that purpose.

Note that the `\tex` macro induced by the `text` symbol above already marks up the “TeX” it produces, so wrapping it in another `\definiendum` would be redundant. However, every `text` symbol also generates a *second* macro with the suffix `name` that generates a non-marked-up version of the same presentation. In other words, we get the macro `\texname` for free, that produces “TeX” (of course, we could just as well use the `\TeX` macro, but that one you probably know already).

Furthermore, every `\definiendum` or `\definame` automatically adds the symbol being referenced to the internal `for=-` list of the `sparagraph` environment, obviating the need to list it explicitly.

As such, we can produce a better markup like this:

```
\begin{sparagraph}[style=symdoc]
  \definiendum{tex}{\texname} is a document typesetting
\end{sparagraph}
```



```
software developed by Donald Knuth, with a focus on
mathematical formulae. It is based on a powerful
and extensible \definame{macro} expansion engine.
\end{sparagraph}
```

### Exercise

In your archive my/archive, create additional files that produce the following outputs:

```
Mathematics.en.tex
```

To do **mathematics** is to be, at once, touched by fire and bound by reason. This is no contradiction. Logic forms a narrow channel through which intuition flows with vastly augmented force.

– Jordan Ellenberg

```
PDF.en.tex
```

**Portable Document Format (PDF)** is a document format that mixes text and graphics with a variety of content.

```
HTML.en.tex
```

The **HyperText Markup Language (HTML)** is a representation format for web-pages.

```
OMDoc.en.tex
```

**OMDoc** is a document format for representing **mathematical** documents with their flexiformal semantics.

such that the following file compiles and shows the above snippets on hover:

```
sTeX.en.tex
1 \documentclass{stex}
2 \libinput{preamble}
3 \begin{document}
4 \begin{smodule}{sTeX}
5   \usemodule{OMDoc}
6   \usemodule{PDF}
7   \usemodule{HTML}
8   \textsymdecl{stex}{\sTeX}
9   \begin{sparagraph}[style=symdoc]
10    \definiendum{stex}{\stexname} is a system for generating
11    documents in either \PDF or \HTML format, augmented with
12    computer-actionable semantic information (conceptually)
13    based on the \OMDoc format and ontology.
14  \end{sparagraph}
15 \end{smodule}
16 \end{document}
```

**sTeX** is a system for generating documents in either **PDF** or **HTML** format, augmented with computer-actionable semantic information (conceptually) based on the **OMDoc** format and ontology.

The preamble of every file should only be

```
\documentclass{stex}
\libinput{preamble}
```

and the macros `\OMDoc`, `\PDF`, `\HTML` should produce `\textsc{OMDoc}`, `\textsf{PDF}` and `\textsf{HTML}`, respectively (but with semantic annotations of course).

---

*Lösung:* Can be found in `[sTeX/Documentation]source/tutorial/solution`

---

## 2.5 Sectioning and Reusing Document Fragments

We know now how to import and reuse the *symbols* of some *module* (using `\usemodule`). What about the actual document *content*?

Assume we want to write a new article that includes all of the fragments in `my/archive` we made so far, in a file `all.en.tex` in the same *math archive*:

```
1 \documentclass{article}
2 \usepackage{stex}
3 \libinput{preamble}
4 \begin{document}
5   \author{Me}
6   \title{The \texttt{my/archive} Archive}
7   \maketitle
8   \tableofcontents
9   ...
10 \end{document}
```

In there, we want sections as follows:

```
- 1 Preliminaries
  (Mathematics)
- 1.1 Document Formats
  (PDF)
  (HTML)
  (OMDoc)
- 2 \TeX and Friends
  (LaTeX)
  (sTeX)
```

We could of course do the following:

```
\section{Preliminaries}
\input{Mathematics.en}
\subsection{Document Formats}
\input{PDF.en}
\input{HTML.en}
\input{OMDoc.en}
\section{\TeX and Friends}
\input{LaTeX.en}
\input{sTeX.en}
```

...but this approach has two drawbacks:

Firstly, we need to manually keep track of the section levels, by explicitly writing `\section`, `\subsection` etc. This is fine as long as we are just interested in this particular

article. But what if we want to *reuse* the article’s content in another document at some point? The section levels might be entirely different then – e.g. we might want the “Preliminaries” section to be a subsection instead.

Secondly, the `\input` macro considers the file name/path provided to be either *absolute* or relative to the *current tex file being compiled* – which means that the `\input{Mathematics.en}` only works for files in the same directory as `Mathematics.en.tex`.

In short: using `\section`, `\chapter` etc. explicitly, and `\input` to reuse fragments, breaks reusability.

Instead of using `\section` and `\subsection`,  $\text{\TeX}$  therefore provides the `sfragment` environment.

`\begin{sfragment}{Foo}... \end{sfragment}` inserts a sectioning header depending on the current section level and availability. These are: `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph` and `\subparagraph`. This allows us to do the following instead:

```
\begin{sfragment}{Preliminaries}
  \input{Mathematics.en}
  \begin{sfragment}{Document Formats}
    \input{PDF.en}
    \input{HTML.en}
    \input{OMDoc.en}
  \end{sfragment}
\end{sfragment}
\begin{sfragment}{\TeX and Friends}
  \input{LaTeX.en}
  \input{sTeX.en}
\end{sfragment}
```

The only problem remaining now is that if we do this,  $\text{\TeX}$  will insert a `\part` for the first `sfragment`. If we want the “top-level” sectioning level to be `\section` instead, we can insert a `\setsectionlevel{section}` in the preamble.

As a more reuse-friendly replacement of `\input`,  $\text{\TeX}$  provides the `\inputref` macro. Using that has two advantages: Firstly, its argument is relative to some (optionally provided, or the current) *math archive* and is thus independent of the specific location of the file relative to the currently being compiled `.tex`-file. Secondly, when converting to `HTML`, it will *not* “copy” the referenced file’s content in its entirety (as `\input` would), but instead dynamically insert the already existent (if so) `HTML` of the referenced file, avoiding content duplication and having to process the file all over again.

In general `\inputref[some/archive]{file/path}` inputs the file `file/path.tex` in the *archive* `some/archive`. As the `\input`-ed files in the example above are in the same *archive* anyway, we can simply substitute the `\inputs` by `\inputrefs` and call it a day.

Finally, we can make two more minor changes:

1. The *title* of our document is only supposed to be there, if we compile the document directly – if we were to `\inputref` our file into a “driver file” `all.en.tex`, the title and the table of contents should be omitted.

We can achieve this using the `\ifinputref` conditional: by wrapping the header in an `\ifinputref \else...\fi`, it will only be processed if the file is *not* being loaded using `\inputref`. `\ifinputref` is a “classic” `TEX` conditional and is treated as such in both `PDF` and `HTML` compilation. A smarter macro to use is `\IfInputref`, which takes two arguments for the *true* and *false* cases, respectively. Additionally, when compiling to `HTML`, *both* arguments to `\IfInputref` will be processed, and the backend will decide which of the two to present when serving a document.

2. The table of contents should also be omitted in `HTML` mode. To achieve that, we can use the `\ifstexhtml` conditional, which is *true* if the document is being compiled to `HTML`, and *false* if compiled to `PDF`.



Note, that since *both* arguments of `\IfInputref` are processed, they should *not* open `TEX` groups or environments!

In summary, we can modify our document to do the following:

```
\IfInputref{}{
  \author{Me}
  \title{The \texttt{my/archive} Archive}
  \maketitle
  \ifstexhtml \else \tableofcontents \fi
}
```

The final `all.en.tex` can be found in `[sTeX/Documentation]tutorial/solution/all.en.tex`.

## 2.6 Building and Exporting `HTML`

So far we know how to write `STEX` documents, (we assume) how to build `PDF` files from them (via `pdflatex` of course), and on saving documents the `IDE` will preview the generated `HTML`. But if we do that with our new `all.en.tex`, we get presented with [Figure 13](#) Where did all of our fragments go?



Figure 13: Missing Fragments in the `HTML` Preview

Well, they don’t exist yet as `HTML`. The `HTML` Preview window in the `IDE` is really just that: A *preview*. But when using `\inputref`, it has to find the `HTML` of the `\inputrefed` fragment *somewhere*. Meaning: we have to compile all of the fragments

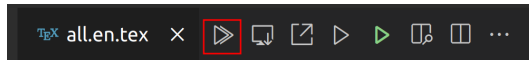


Figure 14: The Build PDF/XHTML/OMDoc Button

we used to [HTML](#) first. Individually, we can compile the currently open file in [VS Code](#) using the button in [Figure 14](#).

This will do the following:

1. Run `pdflatex` over the file three times.
2. Store the resulting `.pdf` in `[archive]/export/pdf/<filepath>.pdf`.
3. Convert the file to [HTML](#) and store it in `[archive]/xhtml/<filepath>.xhtml`.
4. Extract all the semantics and store them as [OMDoc](#) in `[archive]/content/...`, `[archive]/narration/...` and `[archive]/relational/...`
5. Construct a search index in `[archive]/export/lucene/...`

Doing all of this for every individual file *in hindsight* would of course be a huge hassle. We can therefore just compile the full [archive](#), folders in an [archive](#), or whole *groups* of [archives](#) via right-clicking an element in the [Math Archives](#) viewer in the `STEX` tab ([Figure 15](#)).

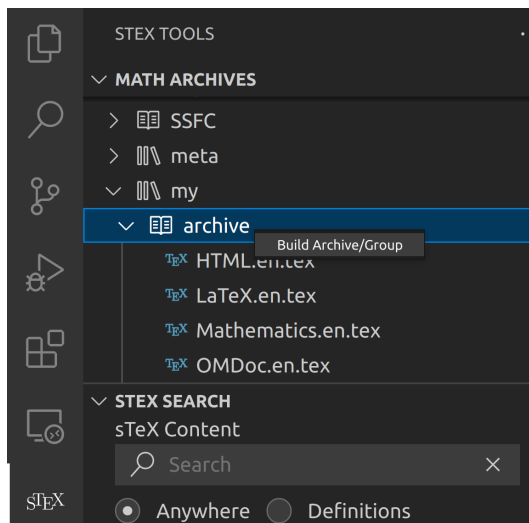


Figure 15: Building Archives in the IDE

Once that's done, saving `all.en.tex` again yields the correct [HTML](#) in the preview window.

At this point, it should be noted that you can't actually just open the [HTML](#) files exported to `[archive]/xhtml` in your browser and get all of the expected functionality – that shouldn't be too surprising. Features like the fancy pop-up windows require a semantically informed backend infrastructure, in the form of the [MMT](#) system. However, [MMT](#) can dump a standalone version for you. Let's do that now:

With our `all.en.tex` file open and everything built as above, click the **Export Standalone HTML** button in the [IDE](#) (see [Figure 16](#)).



Figure 16: Exporting [HTML](#) in the [IDE](#)

In the dialog box that opens now, select an **empty** directory and [MMT](#) will dump a standalone version of our `all.en.tex` document there. You will still not be able to open it in the browser directly, because most browser forbid javascript modules on the `file://` protocol, but opening the file via `http` will yield the desired result, and you can now upload the directory's content to wherever you might want to use it.

If you want to test this, a quick and easy way to do so is to use [VS Code](#): You can install the **Live Server** extension, open the directory and click the **Go Live** button on the lower right of the window, which will start a small web-server in the selected directory and open its `index.html` in the browser for you.

## Chapter 3

# Mathematical Concepts

So far, we have seen how to declare and reference [symbols](#) generate [semantic macros](#) for [text symbols](#), collect them in [modules](#) and document them properly.

But where [sTeX](#) really shines is when it comes to mathematics and related subject areas: [semantic macros](#) are significantly more useful when used for generating symbolic [notations](#) in math mode, and by associating [symbols](#) with (flexi-)formal semantics, [sTeX](#) can even *check* that your content is (to some degree) formally correct, or at least well-formed.

Alos [sTeX](#) provides specialized functionality for mathematical [statements](#): the text fragments marked as Definition, Theorem, Proof that are iconic to mathematical documents.

The example snippets in this chapter can be found in the [math archive sTeX/MathTutorial](#). If you downloaded the [sTeX/Documentation archive](#) in the [sTeX IDE](#), you already have that [archive](#). If not, you can download it from within the [IDE](#), as described in [chapter 2](#).

### 3.1 Simple [Symbol](#) Declarations

We will start with [symbols](#) and [semantic macros](#) for mathematical concepts and objects and their contribution to mathematical formulae.

#### 3.1.1 [Semantic Macros](#) and [Notations](#)

Let us start with a very fundamental concept; namely [equality](#). As you should by now know, declaring a new [symbol](#) requires a [module](#), so let's open a new one and use `\symdecl`:

```
\begin{smodule}{Equality}
  \symdecl{equal}
\end{smodule}
```

As mentioned in [section 2.3](#), the starred variant `\symdecl*` does not create a [semantic macro](#), so presumably, the variant without a *\** *does*. And indeed, we now have a macro `\equal`, which however will produce errors if we try to use it. That's because we haven't told [sTeX](#) what to do with it yet.

STEX

A **semantic macro** is a  $\LaTeX$ -macro that allows for referencing a **symbol** itself, or – in the case of e.g. a function – the *application* of a **symbol** to (one or multiple) *arguments*; primarily by invoking a **symbol**’s *notation* in *math mode*.

The command `\symdecl{macroname}` declares a new **symbol** with name `macroname` and a **semantic macro** `\macroname`. In the case where we want the name and the **semantic macro** to be distinct, the command `\symdecl{macroname}[name=some name]` declares the name of the **symbol** to be `some name` instead.

The starred variant `\symdecl*{name}` declares the concept with the given name, but does not generate a **semantic macro**.

So let’s provide equality with a **notation**. As a first step, we should let  $\TeX$  know that “equal” takes two arguments. We might also want to shorten the **semantic macro** to e.g. `\eq`, without changing the name. Hence:

```
\symdecl{eq}[name=equal, args=2]
```

Next, we add an infix notation with the **notation macro**:

```
\notation{eq}{#1 = #2}
```

That seems like a lot to write, so for the very common case where we want to declare a **symbol** with a **semantic macro** and a **notation** all at once, the `\symdef` macro does all three by combining the optional and mandatory argument of `\symdecl` and `\notation`:

```
\symdef{eq}[name=equal, args=2]{#1 = #2}
```

and indeed, we can now use the `\eq` macro in *math mode* to invoke our new **notation**: `\eq{a}{b}` now yields  $a = b$  – notably without any highlighting (and hover interaction in the **HTML**) though. Since our **semantic macro** takes *arguments*, which should be differently highlighted, we need to let our **notation** know which parts of the **notation** are highlightable components.

We can do so with the `\comp` and `\maincomp` macros:

STEX

The `\comp`-macro marks components to be highlighted in a **notation** for a **symbol** taking (one or more) arguments.

This is necessary because it is (nearly) impossible for  $\LaTeX$  to figure out, which parts of a **notation** to highlight and which not on its own – in particular, the highlighting should stop for the *arguments* of a **semantic macro**.

Additionally, the `\maincomp` macro can be used to mark (at most) one **notation** component to represent the *primary* component of the **notation**.

**Notations** that do not take arguments, as well as **operator notations**, are automatically wrapped in `\maincomp`.

In our case, this applies only to the “=”, symbol, so:

```
\symdef{eq}[name=equal, args=2]{#1 \mathrel{\maincomp{=}} #2}
```



You may be wondering about the role of the `\mathrel` macro in the example above:  $\TeX$  determines spacing/kerling in *math mode* by assigning a *class* to every character. Both individual characters and whole subexpressions can be assigned one of these classes using dedicated macros. These are:





class	TeX macro	examples
ordinary (default class)	<code>\mathord</code>	$\alpha i \diamond$
large operator	<code>\mathop</code>	$\sum \prod \int$
opening	<code>\mathopen</code>	$( [ \langle$
closing	<code>\mathclose</code>	$) ] \rangle$
binary relation	<code>\mathrel</code>	$\leq > =$
binary operator	<code>\mathbin</code>	$+ \cdot \circ$
punctuation	<code>\mathpunct</code>	$, ;$

TeX “forgets” the class of an expression if it is wrapped in a `\comp` macro. It is therefore a good idea to wrap any occurrence of a `\comp` in the corresponding TeX macro for the desired class (e.g. `\mathrel{\comp{\leq}}`).

Having done so, we can now type `\eq{a}{b}` to get  $a = b$ . Thanks to using `\maincomp`, we now also have an operator notation, which we can invoke using `\eq!`, yielding  $=$ .

What if we want to add more notations? Say we want to be able to invoke equality to get the variant notation  $a \equiv b$  (without changing the intended meaning). If we want to be able to choose one of several notations, we should give the notation an identifier.

Let’s again modify our earlier notation by adding the identifier `eq` to the optional arguments of `\symdef`, like so:

```
\symdef{eq}[name=equal, args=2, eq]{#1 \mathrel{\maincomp{=}} #2}
```

We can now invoke the specific notation provided here by writing `\eq[eq]{a}{b}` to the same effect. But we can also add more notations using the `\notation` macro:

```
\notation{eq}[equiv]{#1 \mathrel{\maincomp{\equiv}} #2}
```

which we can now invoke with `\eq[equiv]{a}{b}`, yielding  $a \equiv b$ .

By default, the first notation provided for a given symbol is considered the default notation, which is invoked if the semantic macro is used without an optional argument – hence, `\eq{a}{b}` still yields  $a = b$ .

If we use the starred variant of the `\notation` macro, the notation is set as the new default. Hence, had we done

```
\notation*{eq}[equiv]{#1 \mathrel{\maincomp{\equiv}} #2}
```

then `\eq{a}{b}` would now yield  $a \equiv b$ .

Any already existing notation can be set as default using the `\setnotation` macro; e.g. instead of using `\notation*`, we could also do

```
\notation{eq}[equiv]{#1 \mathrel{\maincomp{\equiv}} #2}
\setnotation{eq}{equiv}
```

### Exercise

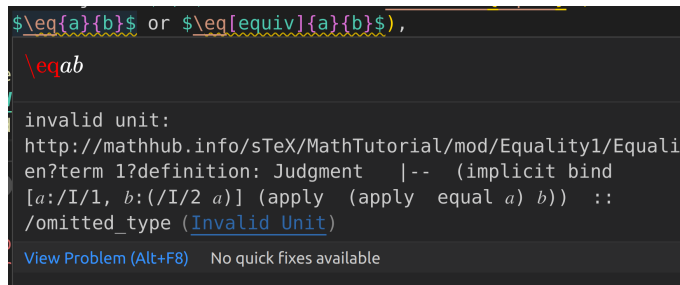
Implement the symbol “equal” as above in a new module “Equality” and add a documentation such that hovering over the symbol in the HTML yields the following snippet:

Two objects  $a, b$  are considered equal (written  $a = b$  or  $a \equiv b$ ), if there is no property that distinguishes them.

Lösung: Can be found in [sTeX/MathTutorial]/mod/Equality1.en.tex

### 3.1.2 Types and Variables

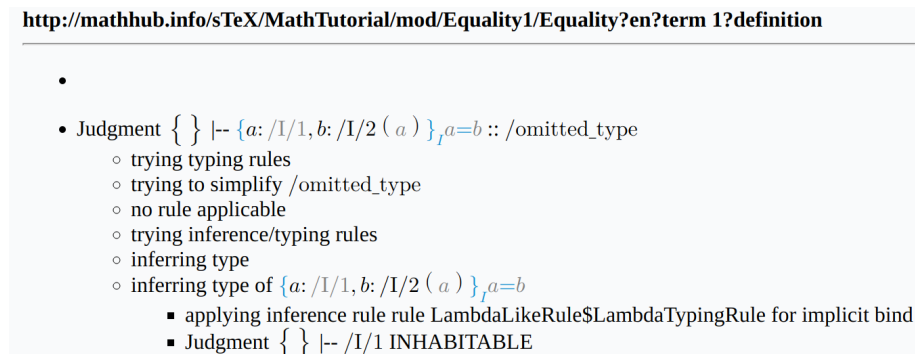
You might have noticed – after you save the file – that the expressions  $\backslash\text{eq}\{a\}\{b\}$  and  $\backslash\text{eq}[\text{equiv}]\{a\}\{b\}$  are underlined in yellow in the IDE and have a warning attached to them (Figure 17). If we click on the Invalid Unit link in the error message, we get



```
 $\backslash\text{eq}\{a\}\{b\}$  or  $\backslash\text{eq}[\text{equiv}]\{a\}\{b\}$ ,  
 $\backslash\text{eq}ab$   
invalid unit:  
http://mathhub.info/sTeX/MathTutorial/mod/Equality1/Equality1/Equality1/en?term 1?definition: Judgment |-- (implicit bind  
[a:/I/1, b:/I/2 a]) (apply (apply equal a) b)) ::  
/omitted_type (Invalid Unit)  
View Problem (Alt+F8) No quick fixes available
```

Figure 17: Type Checking Warning

a somewhat cryptic stacktrace-like window (Figure 18). The reason being, that MMT



```
http://mathhub.info/sTeX/MathTutorial/mod/Equality1/Equality1/Equality1/en?term 1?definition
```

- 
- Judgment  $\{ \} \text{ |-- } \{ a: /I/1, b: /I/2 ( a ) \}_{ a=b } :: /omitted\_type$ 
  - trying typing rules
  - trying to simplify /omitted\_type
  - no rule applicable
  - trying inference/typing rules
  - inferring type
  - inferring type of  $\{ a: /I/1, b: /I/2 ( a ) \}_{ a=b }$ 
    - applying inference rule rule LambdaLikeRule\$LambdaTypingRule for implicit bind
    - Judgment  $\{ \} \text{ |-- } /I/1 \text{ INHABITABLE}$

Figure 18: Type Checking Proof Tree

actually tries to formally verify *everything we write using semantic macros!* It does so, by attempting to infer the *type* of an expression – success implies that the expression is in fact well-typed.

If the former paragraph is difficult to comprehend for you, don’t worry – you’ll likely pick up on things as we go along. For now, suffice it to say that we can assign “*types*” to *symbols*, and the MMT system is smart enough to use those to check that what we’re writing actually “makes sense”; for example,  $a + b$  makes perfect sense if  $+$  is addition and  $a$  and  $b$  are numbers, or elements of a vector space, but not if  $a$  and  $b$  are, say, triangles.

**STEX** Every **symbol** or **variable** can be assigned a **type**, signifying what “kind of object” the **symbol** represents, or what (primary) set it is contained in.



In order to *formally verify* a mathematical statement, we have to rely on a set of *rules* that determine what is or isn’t a valid statement. There are many systems of such rules with very different flavours, called **(logical) foundations**.

The most commonly used **foundation** in (informal) mathematics is *set theory*, in particular *ZFC*; a set of axioms in (usually) *first-order logic*. However, in *computer proof assistants* and similar systems, *type theories* like *higher-order logic* or the *calculus of (inductive) constructions* are more popular, because they lend themselves better to computer implementations.

In as far as possible, we prefer to remain “foundationally agnostic”, or **foundation independent**: Every **foundation** has advantages and disadvantages, and which one is appropriate often depends on the particular setting one is working in. Nevertheless, certain “meta-principles” have proven themselves to be extremely effective in representing and checking mathematical content in software, and while we do not fix a particular **foundation** or specific checking rules, we will make use of those principles in general. These include e.g. the *Curry-Howard Correspondance*, or *Judgments-as-Types paradigm*, and *Higher-Order Abstract Syntax*.



Full formal verification of document content is an extremely lofty goal, and hardly realistic if you’re not willing to write your content in pretty specific ways, and informed by a decent amount of background knowledge in formal logic. Moreover, formally verifying content in **STEX** is an ongoing research project, so we will not go into the specifics in detail here.

While full formal verification is out of reach for now, annotating adequate **types** can strike a useful balance between the effort required and the benefit of automated meaning checking afforded by them. In this sense **STEX** is pragmatically similar to programming languages where adding types can raise the quality and correctness assurance in programs.



Keep in mind that getting `Invalid Unit` warnings does not impact at all what your document is going to look like – feel free to ignore them entirely.

**Types** are particularly useful for *variables*:



A **variable** represents a *generic* or *unspecified* object.

**Variables** can be declared using the `\vardef`-macro, whose syntax is analogous to `\symdef`.

Note that **variables** are local to the current `TeX`-group (e.g. environment).

Let’s leave our `equality-module` aside for now and turn our attention to something simpler: **natural numbers**. Consider the following module:

### Example 5

Input:

```
\begin{smodule}{Nat}
  \symdef{Nat}[name=natural numbers]{\mathbb N}
  \begin{sparagraph}[style=symdoc]
    The \definame{Nat} $\defnotation{\Nat}$ are the numbers
    $0,1,2,\dots$
  \end{sparagraph}
  \symdef{plus}[name=addition,args=2]{#1 \mathbin{\maincomp{+}} #2}
  \begin{sparagraph}[style=symdoc]
    \Definame{addition} $\defnotation{\plus{a}{b}}$
    refers to the process of adding two \sn{Nat}.
  \end{sparagraph}
\end{smodule}
```

Output:

The **natural numbers**  $\mathbb{N}$  are the numbers 0,1,2,...  
**Addition**  $a+b$  refers to the process of adding two **natural numbers**.

(like `\definame` and `\definiendum`, the `\defnotation` macro is only allowed in documenting environments like `sparagraph[style=symdoc]` or `sdefinition`, and highlights the `notation` components marked with `\comp` or `\maincomp` the same way as `\definame` and `\definiendum` do.)

Note, that as the `\Nat` semantic macro does not take any arguments, we do not need to wrap the `notation` in a `\comp` or `\maincomp`.

Note also, that the `\plus{a}{b}` is again underlined in the IDE with an Invalid Unit warning.

The above fragment uses two `variables`  $a$  and  $b$ . In fact, MMT will consider them `variables` even though they are not marked up as such – but since they are not marked up, we are missing out on useful functionality.

Let's change that by adding two `variable` definitions<sup>1</sup>:

### Example 6

Input:

```
\begin{sparagraph}[style=symdoc]
  \vardef{va}[name=a]{a}\vardef{vb}[name=b]{b}
  \Definame{addition} $\defnotation{\plus{\va}{\vb}}$
  refers to the process of adding two \sn{Nat}.
\end{sparagraph}
```

Output:

**Addition**  $a+b$  refers to the process of adding two **natural numbers**.

Okay, so now  $a$  and  $b$  are gray, but besides that, we haven't achieved much yet.

<sup>1</sup>Technically, this is called a *variable reservation*, for those in the know.

Let's change that by giving the variables the type  $\mathbb{N}$ :

### Example 7

Input:

```
\begin{sparagraph}[style=symdoc]
\vardef{va}[name=a,type=\Nat]{a}\vardef{vb}[name=b,type=\Nat]{b}
\Definame{addition} $\defnotation{\plus{va}{vb}}$
refers to the process of adding two \sn{\Nat}.
\end{sparagraph}
```

Output:

**Addition**  $a+b$  refers to the process of adding two natural numbers.

Now if we hover over the  $a$  and  $b$  (in the HTML), it will show us that their type is  $\mathbb{N}$ !

We can of course also assign types to symbols. In the IDE, find the symbol “function space” with semantic macro `\funspace` (in `[sTeX/MathBase/Functions]{mod?Function}`). The OMDoc preview window shows you how to use this symbol (Figure 19). This tells

▼ Symbol `function space` (`\funspace{a_1, \dots, a_n}{b}`)

Type	$(A : \text{SET}, B : \text{SET}) \rightarrow \text{SET}$	
Notations	id	notation
	<code>arrowtimes</code>	$a_1 \times \dots \times a_n \rightarrow b$
	<code>arrowcurry</code>	$a_1 \rightarrow \dots \rightarrow a_n \rightarrow b$
	<code>Arrowtimes</code>	$a_1 \times \dots \times a_n \Rightarrow b$
	<code>Arrowcurry</code>	$a_1 \Rightarrow \dots \Rightarrow a_n \Rightarrow b$

Figure 19: Syntax Preview

us that if we write `\funspace{a_1, \dots, a_n}{b}` (depending on which notation we use), we will get  $a_1 \times \dots \times a_n \rightarrow b$ .

We want `addition` to have type  $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ , hence we do:

```
\symdef{plus}[name=addition,args=2,
type=\funspace{\Nat,\Nat}{\Nat}
]{#1 \mathbin{\maincomp{+}} #2}
```



So far (and when using the use button in the IDE), we have been using the `\usemodule` macro to import content. `\usemodule` is allowed anywhere and imports the referenced module content local to the current TeX group. Now that we use imported symbols in types (and since we are in a module), we



need to make sure that the imported `modules` are also (transitively) *exported*, since our new `symbols` now *depend* on the imported `module`.

For that we use the `\importmodule` macro within the `module`; i.e. the file should now look something like this:

```
\begin{smodule}{Nat}
  \importmodule [sTeX/MathBase/Functions]{mod?Function}
  ...
```

Note that the `HTML` is aware of this now (after you save): *Clicking* on any occurrence of `addition` now yields [Figure 20](#).

Figure 20: On-Click Popup in the `HTML`

However, the squiggly yellow `Invalid Unit` warnings are still there – that’s because everything we did with `types` so far still depends on our `natural numbers` symbol, which does not have a `type` yet.

By virtue of using `[sTeX/MathBase/Functions]{mod?Function}`, we also imported `[sTeX/MathBase/Sets]{mod?Set}`, which gives us the “*collection*” symbol. Let’s use this as a `type` for the `natural numbers`:

```
\symdef{Nat}[name=natural numbers,type=\collection]{\mathbb N}
```

Now if we save the file, all the squiggly lines are gone. Moreover, if you look at the `OMDOC` tab in the preview window, you can find [Figure 21](#). The `Document Elements` block collects all semantically annotated expressions in a `module` or document; including `variables` and the `\plus{\va}{\vb}`. Here, it tells us that it has checked the expression  $a + b$  (in the context of  $a : \mathbb{N}$  and  $b : \mathbb{N}$ ), and inferred that it has `type`  `$\mathbb{N}$` .

Here’s what just happened:

1. The `MMT` system realized, that `\plus{\va}{\vb}` is the symbol “*addition*” applied to the two arguments  $a$  and  $b$ .
2. It knows, that “*addition*” has `type`  `$\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}^2$` .

<sup>2</sup>Do not worry that the IDE actually reports the type `{a :  $\mathbb{N}$ , b :  $\mathbb{N}$ }I $\mathbb{N}$` , this is an artefact of the underlying type system with dependent types used by `sTeX`; it just means  `$\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$`  in this special case, but would also allow  $a$  and  $b$  to appear in the range type in more complex situations; see ?? for details.

▼ Document Elements
▶ Variable <code>a</code> ( <code>\va</code> ) of type <code>N</code>
▶ Variable <code>b</code> ( <code>\vb</code> ) of type <code>N</code>
▼ <code>{a: N, b: N}_a+b</code>
Inferred Type: <code>{a: N, b: N}_N</code>

Figure 21: Inferred Type

- It knows, that this means that if the two arguments  $a$  and  $b$  both have `type N`, then the full expression has `type N`.

Here’s something you can now try: If we *remove* the `types` from the `variables`  $a$  and  $b$  again, the warnings are *still* gone. We lose the `type` information on hover, but `MMT` still doesn’t complain, because it now realizes that since  $a$  and  $b$  have no explicit `types` given, it should infer them. And by the same chain of reasoning as above, it can infer that since they are being used as arguments for `addition`, they need to have `type N`.

### 3.1.3 Flexary Macros and Argument Modes

Here is one thing you might wonder: Writing `\plus{a}{b}` is one thing, but what if we want to produce  $a + b + c + d + e$ ? Do we really need to write `\plus{a}{\plus{b}{\plus{c}{...}}}`?

Of course not. We can declare the `symbol` such that the `semantic macro` `\plus` expects a (comma-separated) *sequence* of arguments instead of two “normal” arguments.

STEX

The optional `args`-argument of `\symdecl` expects a string of characters indicating the `semantic macro`’s **argument modes**. There are four such **modes**:

- i a **simple argument**,
- a a – (*left or right*) **associative – sequence argument**, represented as a single TeX-argument `{a,b,...}`,
- b A **binding argument** that expects a variable that is bound by the `symbol` in its application, and
- B A **binding sequence argument** of arbitrarily many bound variables by the `symbol` (`{x,y,z,...}`).

If `args` is given as a number  $n$  instead, the `semantic macro` takes  $n$  arguments of **mode** `i`.

#### Example 8

- For `\plus{a,b,c}` yielding  $a + b + c$ , we do `\symdecl{plus}[args=a]`,
- for `\inset{a,b,c}{A}` yielding  $a, b, c \in A$ , we do `\symdecl{inset}[args=ai]`,
- in `\add{i}{1}{n}{f(i)}` yielding  $\sum_{i=1}^n f(i)$ , the variable  $i$  is **bound** in the ex-

pression, we hence do `\symdecl{add}[args=biii]`,

- in `\forall{x,y,z}{P(x,y,z)}` yielding  $\forall x,y,z. P(x,y,z)$ , the variables  $x,y,z$  are all **bound** by the  $\forall$ , we hence do `\symdecl{forall}[args=Bi]`.

So when we wrote `\symdecl{plus}[args=2]`, this was actually shorthand for `\symdecl{plus}[args=ii]`.

Let's revise our previous declaration and the syntax of the `\plus` macro:

```
\symdef{plus}[name=addition,args=a,
  type=\funspace{\Nat,\Nat}{\Nat}
]{#1 \mathbin{\maincomp{+}} #2}
\begin{spargraph}[style=symdoc]
  \vardef{va}[name=a]{a}\vardef{vb}[name=b]{b}
  \Definame{addition} $\defnotation{\plus{va,vb}}$
  refers to the process of adding two \sn{Nat}.
\end{spargraph}
```

Now we get new errors, that are easy to explain: Our **notation** `{#1 \mathbin{\maincomp{+}} #2}` refers to *two* arguments, but our **semantic macro** only takes *one* (albeit a **sequence argument**). We now need to let **TEX** know what to do with the **sequence argument** in our **notation**. Using the `\argsep` macro, we can tell **TEX** to insert the *separator* “+” between the individual elements of the **argument sequence** #1:

```
\symdef{plus}[name=addition,args=a,
  type=\funspace{\Nat,\Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{+}}}}
```

Now we can finally write `+\plus{a,b,c,d,e}` and get  $a + b + c + d + e$  – hooray! ...expect that our squiggly yellow **Invalid Unit** warnings are back. That's because the **type** of **addition** still corresponds to a binary operation, rather than a unary function on sequences.

We *could* change the **type** of course, but we shouldn't *want* to or *have* to: platonically, **addition** is *still* a *binary function*; we just introduced the **a-mode** argument for *our* convenience as authors.

Instead, we can tell **MMT** how to “resolve” the **sequence argument** into a nested application of **addition**. In the very common case we have here, where the **symbol** represents an *associative binary operator*, we can just add the argument `assoc=bin` to the `\symdecl` (or `\symdef`) **macro**:

```
\symdef{plus}[name=addition,args=a,assoc=bin,
  type=\funspace{\Nat,\Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{+}}}}
```

and the warnings are gone again. Formally/internally, **MMT** will now turn the term `addition(sequence(a,b,c))` into `addition(a,addition(b,c))`.

### Exercise

Analogously to the above, implement a **symbol** “multiplication” with **semantic macro** `\mult`, that takes a single **sequence argument** and has a default **notation** such that `\mult{a,b,c}` produces  $a \cdot b \cdot c$ .

---

*Lösung:* Can be found in `[sTeX/MathTutorial]mod/Nat.en.tex`

---



### 3.1.4 Precedences

If you have done the previous exercise, you now have `semantic macros` `\plus` and `\mult` at your disposal. We can of course nest them to produce e.g.  $a + b \cdot c$  (with `\plus{a, \mult{b, c}}`). If we do `\mult{a, \plus{b, c}}` however, we get  $a \cdot b + c$ . Annoying – we now have to insert parentheses: `\mult{a, (\plus{b, c})}`... or do we?

We do *not*. Instead, we can assign *precedences* to *notations* to have `TeX` insert parentheses automatically.

`\notation` (and hence `\symdef`) take an optional argument `prec=<opprec>;<argprec1>x...x<argprec n>` consisting of an **operator precedence** `<opprec>` and for each argument `k` an **argument precedence** `<argprec k>`.

All *precedences* are integers, e.g. 10 or -500. It is good practice to use *precedences* that leave enough room to smuggle values inbetween, so that we can fine-tune them later as more symbols may intervene.

The precise numbers used for *precedences* are arbitrary – what matters is which *precedence* is higher than which other *precedence* when used together.

By default, all *precedences* are 0, unless the *symbol* takes no arguments, in which case the *operator precedence* is `\neginfprec` (negative infinity).

If we only provide a single number in `prec=`, this is taken as both the *operator precedence* and all *argument precedences*.

The *lower* a *precedence*, the *stronger* a *notation* binds its arguments. In our particular case, we want *multiplication* to bind stronger than *addition*, so we can (arbitrarily) assign them *precedences* e.g. 10 and 20:

```
\symdef{plus}[name=addition, args=a, assoc=bin, prec=20,
  type=\funspace{\Nat, \Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{+}}}}
\symdef{mult}[name=multiplication, args=a, assoc=bin, prec=10,
  type=\funspace{\Nat, \Nat}{\Nat}
]{\argsep{#1}{\mathbin{\maincomp{\cdot}}}}
```

And now if we type `\mult{a, \plus{b, c}}`, `TeX` will automatically insert parentheses and yield  $a \cdot (b + c)$  – and conversely, if we do `\plus{a, \mult{b, c}}`, `TeX` will *not* insert parentheses and yield  $a + b \cdot c$ .

### 3.1.5 Implicit Arguments

Let us turn our attention back to *equality*. Here’s an almost philosophical question: *What is the type of “equality”?* Asking (the right kind of) mathematicians this question can cause fist fights to break out. As such, we will not give a definitive answer, *but* here is an informative approach that has proven to be quite effective in computational settings:

*Equality* is a *polymorphic binary relation* on an *implicit collection*  $A$ . And a *relation* is a function into a *type* of *propositions*.

We will see the advantage of this approach over time. For now, consider that given objects  $a$  and  $b$ , the expression “ $a = b$ ” is either true or false<sup>3</sup>, and “*equal*” takes two argu-

<sup>3</sup>Assuming classical logic – if you prefer to remain intuitionistic/constructive, note that `TeX`, being *foundation independent*, does not enforce the law of excluded middle!

ments, so if we have a `type` of “truth values”, it makes sense to model “`equal`” as a function taking two arguments and returning that `type`. So we do `type=\funSPACE{.....}`?

Here’s the idea with respect to *implicit arguments*. Let’s first declare a new `variable` of `type` “`collection`”:

```
\vardef{vA}[name=a,type=\collection]{A}
```

We now assign the `type`  $A \times A \rightarrow \text{Prop}$  to `equal`:

```
\symdef{eq}[name=equal,args=2,eq,
type=\funSPACE{\vA,\vA}{\prop}
]{#1 \mathrel{\maincomp{=}} #2}
```

(The `symbol` “`proposition`” with `semantic macro` `\prop` comes with `STEX` directly; we say that it is part of the `STEX`.)

Now our `type` has a free variable  $A$ . For `MMT`, this now means that `equal` actually takes *one more argument*, but one whose value is uniquely determined from the other arguments. Indeed, if you consider `equal` to take three arguments (the first one being some  $A$  of `type` `collection`), then the *next* two arguments *enforce* that the first argument has to be the `type` of the other two.

In other words:  $A$  is now an implicit argument that `MMT` is tasked with inferring whenever we use `equal`, and that we never explicitly provide in `STEX`.

Indeed, if we use our `module` `Nat` from before, and apply `\eq` to a variable of `type` `N`, `MMT` does not complain:

```
\usemodule{mod?Nat}
\vardef{vn}[name=n,type=\Nat]{n}
$\eq{\vn}{m}$
```

And if we inspect the `OMDOC` tab in the `HTML` preview, we can see exactly what `MMT` did (Figure 22). We can see

Figure 22: Implicit Arguments

1. (by the  $\{\cdot\}_I \dots$ ) that `MMT` considers  $A$  an implicit argument in the `type` of `equal`,

2. that the *inferred* type of  $n = m$  is `Prop`,
3. that `MMT` inferred the implicit argument of `equal` in  $n = m$  to be `N` (by the  $\underbrace{\dots}_N$ ),  
and
4. that it was enough to give `\vn` the explicit `type N` – `MMT` also inferred that hence  $m$  also has to have `type N`!

### 3.1.6 Finishing Equality

You might wonder if – as with `addition` – we can make “`equal`” take a `sequence argument` as well. Naturally, we can:

```
1 \symdef{eq}[name=equal,args=a,eq,
2   type=\funspace{\vA,\vA}{\prop}
3 ]{\argsep{#1}{\mathrel{\maincomp=}}}
4 \notation{eq}[equiv]{\argsep{#1}{\mathrel{\maincomp\equiv}}}
```

and as before, we now get `Invalid Unit` warnings. Unlike before, however, we can not just fix this with adding `assoc=bin`. As mentioned, `bin` instructs `MMT` to “fold” the `symbol` over the arguments, so when doing `\eq{a,b,c}`, `MMT` would turn this into `equal(a,equal(b,c))`, i.e. the claim that “ $a$  is equal to “ $b = c$ ” – but that’s not what  $a = b = c$  means. What we mean by  $a = b = c$  is really “ $a = b$  and  $b = c$ ”.

For that, we can use `assoc=conj` – however, that requires that some `symbol` that can be used for *conjunction* (i.e. “and”) is in the current scope.

If we search for `conjunction` in the `IDE`, we should find the `module` `[sTeX/Logic/General]{mod/syntax?Conjunction}`.

Using that, we can now write the following:

```
\usemodule{mod?Nat}
\usemodule[sTeX/Logic/General]{mod/syntax?Conjunction}
\vardef{vn}[name=n,type=\Nat]{n}
 $\eq{\vn,m,p}$ 
```

Upon saving, `MMT` does not complain; and if we inspect the `OMDoc` tab in the `HTML` window again, we now notice that `MMT` correctly resolved this as in [Figure 23](#).

### 3.1.7 Variable Sequences

There is a special kind of `variable` in `sTeX` for when we want to use *sequences* of `variables`.

We can use the `\varseq` macro to declare a new `sequence variable`; in the simplest case that looks something like the following:

```
\varseq{seqn}[name=n,type=\Nat]{1,\ellipses,k}{\maincomp{n}_{#1}}
```

We have just declared a new `variable sequence` of `type N`, that ranges over indices  $1, \dots, k$ , with `notation`  $n_i$  for some specific index  $i$ .

If we now do `\seqn{i}`, we get  $n_i$ , and if we do `\seqn!`, we get  $n_1, \dots, n_k$ .

We can also do multi-dimensional sequences, e.g.

```
\varseq{seqm}[name=m,type=\Nat,args=2]
  {{1}{1},\ellipses,{\ell}{k}}
  {\maincomp{m}_{#1}^{#2}}
```

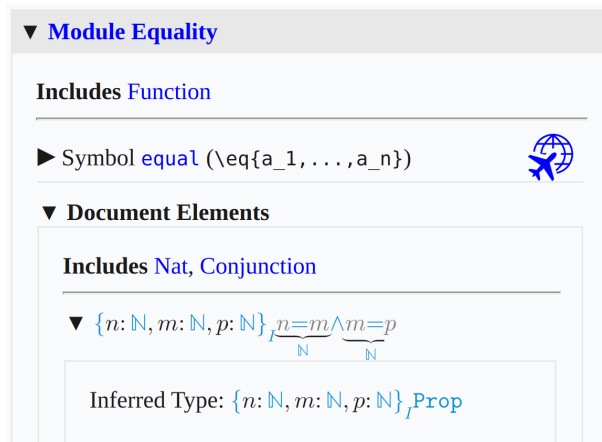


Figure 23: Conjunction of Equalities

Now `\seqm{i}{j}` produces  $m_i^j$ , and `\seqm!` produces  $m_1^1, \dots, m_\ell^k$ .

Of course, we can manually change the way `\seqn!` is typeset by providing an explicit operator notation using `op=`; e.g. if we do

```
\varseq{seqn}[name=n, type=\Nat, op={ (n_i)_{i=1}^k }
{1, \ellipses, k}{\maincomp{n}_{#1}}
```

then `\seqn!` produces  $(n_i)_{i=1}^k$ .

So far so nice, but sequence variables get especially useful in combination with [sequence arguments](#): Consider for example the `\plus semantic macro` for [addition](#). This expects one [sequence argument](#), or alternatively, a *sequence variable*: `\plus{\seqn}` now produces  $n_1 + \dots + n_k$ , and `\eq{\seqm}` now produces  $m_1^1 = \dots = m_\ell^k$ .

**TODO<sup>4</sup>**

## 3.2 Statements

Now that we have [equality](#), [natural numbers](#), [addition](#) and [multiplication](#) at our disposal, let's implement some *statements*. Both [addition](#) and [multiplication](#) are, for example, *associative* and *commutative*.

We could state these properties directly for the two operations, but we can also first define *associativity* and *commutativity* in general, and then assert them specifically for [addition](#) and [multiplication](#).

### 3.2.1 Definitions

Let's define what it means to be *associative*. This means, of course, declaring a new [symbol](#). Note that we don't need a [semantic macro](#) for [associativity](#), since there is no [notation](#) to attach to it. We will also for now ignore its [type](#). Note however, that [associativity](#) is still a property of (binary) operations, so it still makes sense to have the [symbol](#) take an *argument*; namely the operation it applies to.

<sup>4</sup>**TODO: seqmap**

We will also finally provide an actual (more or less) formal *definition* for the [symbol](#), so where we used the [sparagraph environment](#) with `style=symdoc` before, we will now use the [sdefinition environment](#), which also gives us `\definame`, `\definiendum`, `\defnotation` and all that.

A first variant of a corresponding [module](#) could look like this:

### Example 9

Input:

```

File [sTeX/MathTutorial]props/Associative1.en.tex
4 \begin{smodule}{Associative}
5   \importmodule{mod?Equality}
6
7   \symdecl*{associative}[args=1]
8   \begin{sdefinition}[for=associative]
9     \vardef{vA}[name=A,type=\collection]{A}
10    \vardef{vop}[name=op,type=\funspace{\vA,\vA}\vA,args=a,assoc=bin]
11      {\argsep{#1}{\mathbin{\maincomp{\circ}}}}
12    %
13    A binary operation  $\fun{\vop!}{\vA,\vA}\vA$  is called
14    \definame{associative}, if
15    $eq{
16      \vop{(\vop{a,b}),c},
17      \vop{a,(\vop{b,c})}
18    }$ for all  $\inset{a,b,c}\vA$ .
19  \end{sdefinition}
20 \end{smodule}

```

Output:

**Definition 3.2.1.** A binary operation  $\circ : A \times A \rightarrow A$  is called **associative**, if  $(a \circ b) \circ c = a \circ (b \circ c)$  for all  $a, b, c \in A$ .

Note, that the [semantic macros](#) `\fun` and `\inset` come from `[sTeX/MathBase/Functions]mod?Function` and `[sTeX/MathBase/Sets]mod?Set`, respectively. Also note, that the [variable](#) declaration for `\vop` makes use of all the [fun](#) features we already discussed for [addition](#).



Note that the above is more than good enough, if you merely want to produce nice-looking, “wikified” [HTML](#) and [PDF](#) documents. The rest of this subsection will cover how to add more flexiformal semantics to the above.

If this seems laborious and/or difficult, keep in mind that this is to some degree experimental still, and you are not forced to go overboard with semantic annotations!

But if you aim to create a “library of symbols” for mathematical concepts, then all of the possibilities that we discuss here will add value for the community. Generally, the higher the ratio of readers to authors the more any investment in semantization will pay off.

## Semantic Macros in Text Mode

The first thing we can do to further improve this document is marking up the “for all” in the definition – after all, there naturally is a [symbol](#) for the [universal quantifier](#), which can be found in `[sTeX/Logic/General]mod/syntax?UniversalQuantifier` and has the [semantic macro](#) `\forall` (as to not conflict with the [TeX primitive macro](#) `\forall`).

The naive approach would be to replace the “for all” by e.g. `\sr{forall}{for all}`. That would (correctly) associate and highlight the text fragment with the [symbol](#) “[universal quantifier](#)”, *but* we are not just referencing the [symbol](#) here – we are actually using it, by *applying* it to the [variables](#)  $a, b, c$  and the expression  $(a \circ b) \circ c = a \circ (b \circ c)$ .

In *math mode*, we can just use the [semantic macro](#) `\forall` – that will take two arguments (of [modes](#) B and i) and produce the corresponding [notation](#), so that

```
\forall{\inset{a,b,c}{\vA}}{
  \eq{ \vop{(\vop{a,b}),c} , \vop{a,(\vop{b,c})} }
}
```

will produce  $\forall a, b, c \in A. (a \circ b) \circ c = a \circ (b \circ c)$ .

In *text mode*, however, we don’t have a specific [notation](#) – instead, the specific “[notation](#)” is whatever sentence we want to mark up semantically. In text mode, [semantic macros](#) therefore behave differently:

1. They take *precisely* one argument, regardless of how many arguments the [macro](#) would take in math mode or (equivalently) the [args](#) property of the [symbol](#).
2. *Within* that argument, we can use `\comp` to highlight arbitrary text fragments, and
3. we can use the `\arg` [macro](#) to mark up the *actual* arguments that the [symbol](#) is supposed to be applied to.

`\arg` takes as optional argument the index of the argument that is being marked up; if not they are used consecutively. The starred variant `\arg*` produces no output.

So we could now do

```
\forall{\comp{For all} $\arg{\inset{a,b,c}{\vA}}$, we have
  $\arg{
    \eq{ \vop{(\vop{a,b}),c} , \vop{a,(\vop{b,c})} }
  }$
}
```

which produces “[For all](#)  $a, b, c \in A$ , we have  $(a \circ b) \circ c = a \circ (b \circ c)$ ”.

In our case though, we want to “switch the arguments around” – first comes the equation, then the [variables](#) to be bound. Hence:

```
\forall{
  $\arg[2]{
    \eq{ \vop{(\vop{a,b}),c} , \vop{a,(\vop{b,c})} }
  }$
  \comp{for all}
  $\arg[1]{ \inset{a,b,c}{\vA} }$
}
```

which produces “ $(a \circ b) \circ c = a \circ (b \circ c)$  [for all](#)  $a, b, c \in A$ ”.

## Definientia

Now we have a fully semantically annotated expression in the definition for “associative”. Can we let MMT know, that this expression really is *the* definition of the symbol?

Yes, we can. All we need to do is wrap the sentence in a `\definiens` macro (plural: *definientia*; like the word “*definiendum*” refers to “the term being defined”, “*definiens*” refers to “the thing the term is being defined as”).

The `\definiens` macro is only allowed within the `sdefinition` environment, and requires that the `environment` lists the `symbol` that gets the `definiens` attached explicitly in its `for=` argument. It is possible to attach `definientia` to multiple `symbols` within an `sdefinition` environment, in which case the symbol needs to be provided as an optional argument, e.g. we could do `\definiens[associative]{...}`. Since “associative” is the only `symbol` being defined in our definition, we can omit that argument.

Alternatively, as with `types` we can attach `definientia` to a `\symdecl` directly using the optional argument `def=...`

At this point, you might justifiably wonder, why we even still need to declare `associative` with `\symdecl*` before we define it. And indeed, we don’t – the `sdefinition` environment takes the same optional arguments as the `\symdecl` macro, and if we explicitly provide a `name=` (or a `macro=`), it will generate a `symbol` for us. We can hence get rid of the `\symdecl*` and instead do:

```
1 \begin{sdefinition}[name=associative,args=1]
2   ...
3 \end{sdefinition}
```

One more problem remains: We stated that `associative` is to take one argument – but we haven’t told `STEX` what it is yet. In our case, the argument is represented by the `variable` `\vop`. In fact, chances are that arguments to symbols in `types` or `definientia` are almost always represented by some `variable`.

We can use one of two ways to a `variable` as being an argument:

1. If the `variable` (e.g. `\vop` with name `op`) was already declared prior to the `sdefinition` environment, we can use the `\varbind` macro in the `environment`; e.g. by adding `\varbind{op}`.
2. We can move (or copy) the `\vardef` for the `variable` into the `environment` and add `bind` to its optional arguments.

In total, our fully annotated definition now looks like this:

### Example 10

Input:

```

File [sTeX/MathTutorial]props/Associative.en.tex
8 \begin{sdefinition}[name=associative,args=1]
9 \vardef{vA}[name=A,type=\collection]{A}
10 \vardef{vop}[name=op,type=\funspace{\vA,\vA}\vA,
11 args=a,assoc=bin,bind % <- argument for the symbol
12 ]{\argsep{#1}{\mathbin{\maincomp{\circ}}}}
13 \vardef{va}[name=a,type=\vA]{a}
14 \vardef{vb}[name=b,type=\vA]{b}
15 \vardef{vc}[name=c,type=\vA]{c}
16 %
17 A binary operation  $\fun{\vop!}{\vA,\vA}\vA$  is called
18 \definame{associative}, if
19 \definiens{\foral{\arg[2]{\eq{
20 \vop{\vop{va,vb}},vc},
21 \vop{va,(\vop{vb,vc})}}
22 }}{\comp{for all} \arg[1]{\inset{va,vb,vc}{vA}}}.
23 \end{sdefinition}
24 %

```

Output:

**Definition 3.2.2.** A binary operation  $\circ : A \times A \rightarrow A$  is called **associative**, if  $(a \circ b) \circ c = a \circ (b \circ c)$  for all  $a, b, c \in A$ .

And indeed, if we look at the [OMDOC](#) tab of the [HTML](#) preview, we can see that not only does [MMT](#) attach the `definiens` to the `symbol`, it has also inferred the `type` of “associative” from the `definiens` ([Figure 24](#)).

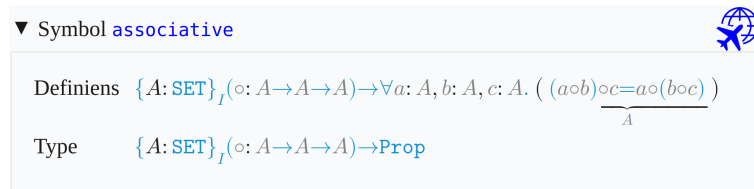


Figure 24: Type Inferred from Definiens

## Using Symbols Without Semantic Macros and Exporting Code in Modules

So now we don’t have a `semantic macro` for “associative”, but it *does* take an argument. How can we ever actually *use* the `symbol` now?

The answer is: with the `\symuse` macro. Like `\symref` and friends, `\symuse` takes a `symbol` name or the name of its `semantic macro` as argument, but behaves otherwise like using a `semantic macro` directly. So for, say, `addition`, `\symuse{addition}` and `\symuse{plus}` behave exactly like `\plus`.

In our case, this means we can do `\symuse{associative}`. “associative” does not have a `notation`, but in practice, we say something like “+ is associative” rather than using some specific mathematical `notation` for the same thing.



Combining this with what we just learned, we can now say that `addition` is `associative` by doing:

```
\symuse{associative}{$\arg{\plus!}$ \comp{is associative}}
```

In fact, we would do the exact same thing every time we want to say that *some* operator is associative, so it makes sense to introduce a `macro` for this. In fact, such a `macro` is easy to define using standard `LATEX` methods. This is where `\STEXexport` becomes very handy:

In a `module`, we can put arbitrary `LATEX` code in an `\STEXexport`, and this code will be executed every time the `module` is imported via `\usemodule` or `\importmodule`. This is especially useful for `macro` definitions, and this way `modules` can almost act like `LATEX` packages!

So we can define a new `macro` `\isassociative` that applies “`associative`” to an arbitrary operation and produces the semantically marked-up text “`#1 is associative`”, and wrap that `macro` definition in an `\STEXexport`, and whenever we use the `Associative` `module`, we also get the `\isassociative` `macro`:

```
\STEXexport{
\def\isassociative#1{
\symuse{associative}{\arg{#1} ~is ~\comp{associative}}
}
}
```

And now, we can do e.g. `\isassociative{\plus!$}` to produce “`+ is associative`”.



For technical reasons, `\STEXexport` processes its content in the `expl3` category code scheme – what this means is that all spaces are ignored entirely, and the characters `_` and `:` are valid characters in `macro` names. In practice, this means you will have to use the `~` character for spaces, and if you want to use a subscript `_`, you should use the `macro` `\c_math_subscript_token` instead.

### Exercise

Analogously to all the above, implement a `module` for *commutativity*; i.e. the property of a binary operation that  $a \circ b = b \circ a$  for all  $a, b$ . Make the `module` export a `macro` `\iscommutative` analogously to `\isassociative`.

---

*Lösung:* Can be found in `[sTeX/MathTutorial]props/Commutative.en.tex`

---

TODO<sup>5</sup>

### 3.2.2 Assertions

Having defined `associativity` and `commutativity`, we can now assert that both properties hold for `addition` and `multiplication`.

For *assertions* (i.e. theorems, lemmata, axioms, claims,...), `sTeX` provides the `sassertion` environment.

In the simplest case, that can look like the following:

---

<sup>5</sup>TODO: intent?

```

\begin{sassertion}
  \isassociative{\Sn{plus}}
\end{sassertion}

```

which yields

Addition is associative

Do we want this to be typeset as a **Theorem**? For that we just add a `[style=theorem]` to the `sassertion` environment, provided we have a customization for that – (see [chapter 9](#)). We can also load the `stexthm` package, which uses the `amsthm` package to provide common typesettings for the types: `theorem`, `observation`, `corollary`, `lemma`, `axiom` and `remark`.

So far, this is not too useful – after all, we could have just as well used e.g. the `amsthm` package and gone straight for the non-`STEX` variant

```

\begin{theorem}
  \isassociative{\Sn{plus}}
\end{theorem}

```

But as with `sdefinition`, we can immediately add a corresponding `symbol` in the `sassertion` environment, and have it be documented directly by the environment:

```

\begin{sassertion}[style=theorem,name=addition is associative]
  \isassociative{\Sn{plus}}
\end{sassertion}

```

And now, if we do `\sn{addition is associative}`, we get `addition is associative` with a corresponding hover pop-up (in the [HTML](#)).

Of course, the usefulness of this grows with more elaborate assertions. For very short assertions such as the above, we might not even want to typeset them in such a space hungry manner.

For that purpose, we provide the `\inlineass` macro (and analogously: `\inlinedef` for `sdefinition`), which takes the same optional arguments as the environment. So we could also do:

```

\inlineass[name=addition is associative]{\isassociative{\Sn{plus}}}

```

So far, `MMT` is blissfully unaware of the semantic contents of our assertions. We can easily remedy that by wrapping the expression representing the assertion in a `\conclusion` macro, analogously to the `definiens` macro in `sdefinitions`:

```

\inlineass[name=addition is associative]{
  \conclusion{\isassociative{\Sn{plus}}}
}

```

We can now see the statement in the [OMDoc](#) tab of the [HTML](#) preview ([Figure 25](#)).

Figure 25: Assertion Statement in [OMDoc](#)

Not exactly pretty – the [OMDoc](#) tab uses `notations` to render content, and we did not provide any for `associative`.

Notice the  $\vdash$  symbol after the name of the assertion? As an aside for those who are curious:

STEX

The **judgments as types** paradigm represents the validity of **proposition** via a designated *type of proofs*: For any **proposition**  $P$ , we introduce a collection  $\vdash P$  of *proofs* of  $P$ .

To say that the **proposition holds** is then equivalent to positing that *some* element  $p : \vdash P$  exists – in which case *proofs* become typed objects in their own right.

Let’s consider a more interesting statement now. How about the **induction axiom**?

```
\begin{sassertion}[style=axiom,name=induction axiom]
Let  $\varphi(n)$  a property on  $\mathbb{N}$ . If
\begin{enumerate}
\item  $\varphi(0)$  and
\item if  $\varphi(m)$  holds for some  $m$ , then
 $\varphi(m+1)$  also holds,
\end{enumerate}
then  $\varphi(n)$  holds for all  $n \in \mathbb{N}$ .
\end{sassertion}
```

**Axiom 3.2.3.** Let  $\varphi(n)$  a property on *natural numbers*. If

1.  $\varphi(0)$  and
2. if  $\varphi(m)$  holds for some  $m$ , then  $\varphi(m+1)$  also holds,

then  $\varphi(n)$  holds for all  $n \in \mathbb{N}$ .

### Exercise

Annotate the above by:

1. **Variables** with appropriate **notations** for  $\varphi$ ,  $m$  and  $n$ , and
2. marking up the second premise (“if  $\varphi(m)$  holds for some...”.) in text mode as the formula  $\forall m. \varphi(m) \Rightarrow \varphi(m+1)$  using the **semantic macros** `\forall` (which we saw earlier already) and `\imply` (**implication**) from `[sTeX/Logic/General]mod/syntax?Implication`. The text fragments that should be highlighted are “if” and “then”.
3. marking up the conclusion (“ $\varphi(n)$  holds for all  $n \in \mathbb{N}$ ”) in text mode as the formula  $\forall n. \varphi(n)$ . The text fragment that should be highlighted is “for all”.

---

*Lösung:* Can be found in `[sTeX/MathTutorial]mod/NatTheorems.en.tex`

---

So how can we teach **MMT** the semantics of this statement? Here’s what we can do:

1. As with the simpler assertions (and hence the name), the *conclusion* of the assertion can be marked up with `\conclusion`.

2. As with `sdefinition`, we can mark `variables` as *bound* (using either `bind` in the `\vardef` or `\varbind`). If a `symbol` that can act as a `universal quantifier` is in scope, `variables` marked as bound are abstracted away using that `symbol`.
3. Similarly to `\conclusion`, `premises` can be marked up as such using the `\premise` macro. If a `symbol` is in scope that can act as an `implication`, that will be used to connect the premise(s) to the conclusion.

Hence, if we mark the variable  $\varphi$  as bound and use `\premise` and `\conclusion` (see `[sTeX/MathTutorial]mod/NatTheorems.en.tex`), we can inspect the `OMDOC` tab in the `HTML` preview again and see that `MMT` has now constructed the `proposition` (Figure 26).

▷ Assertion `induction axiom`  $\vdash \forall \varphi: \mathbb{N} \rightarrow \text{Prop}. \varphi(0) \Rightarrow (\forall m: \mathbb{N}. \varphi(m) \Rightarrow \varphi(m+1)) \Rightarrow (\forall n: \mathbb{N}. \varphi(n))$

Figure 26: The `Induction Axiom` in `OMDoc`

### 3.2.3 Proofs



`sTeX` provides the `sproof` environment for marking up *proofs*. The markup mechanism for `sproof` is still highly experimental and likely subject to change in the near future. As such, we omit a closer explanation of its usage until the syntax and functionality have sufficiently stabilized.

## 3.3 Mathematical Structures

A common concept in mathematics is that of a `mathematical structure` – a *tuple* of interdependent components. For example: A *monoid* is a `structure`  $\langle M, \circ, e \rangle$  such that certain axioms hold; where  $M$  is a set,  $\circ$  is a binary operation, and  $e \in M$ .

From a representational perspective, this is particularly interesting:  $M$ ,  $\circ$  and  $e$  in the above are not `symbols` in the same way that the previous `symbols` we considered were – they don't represent definite objects. Instead, they are *components* of some other object, namely a monoid; where a *particular* monoid could either be a fixed object (such as  $\langle \mathbb{Z}, +, 0 \rangle$ ) or an *indefinite* monoid; i.e. a `variable`. We call the components of a `mathematical structure` **fields**.

In this section, we will discuss how to declare and use `mathematical structures` in `sTeX`, build them up modularly, and connect them among each other to avoid duplication.

We will do so by considering *lattices* both algebraically and order-theoretically, and identify the two perspectives.

### 3.3.1 Declaring and Using Structures

The simplest kinds of `structures` are *magmas* and (*directed*) *graphs*, so we might as well start there:

**Definition 3.3.1.** A **magma** is a structure  $\langle U, \circ \rangle$ , where  $U$  is a collection and  $\circ$  a binary operation  $U \times U \rightarrow U$ .

The obvious start is to create a new module `Magma`. Within this module, we import the `Functions` module so we can later assign a type to the operation. We can then use the `mathstructure` environment, that creates a new symbol “magma”:

```
\begin{smodule}{Magma}
  \importmodule[sTeX/MathBase/Functions]{mod?Function}
  \begin{mathstructure}{magma}
    ...
  \end{mathstructure}
\end{smodule}
```

`mathstructure` behaves very similarly as `smodule` – within the environment, we can declare new symbols, notations and all that.

So within the `mathstructure`, we can add symbols for the two fields  $U$  and  $\circ$ :

```
\symdef{univ}[name=universe,type=\collection]{U}
\symdef{op}[name=operation,args=a,assoc=bin,
type=\funspace{\univ,\univ}\univ
]{\argsep{#1}{\mathbin{\maincomp\circ}}}
```

Once we close the environment (with `\end{mathstructure}`), the symbols are “gone”. However, we now have a new symbol “magma” with semantic macro `\magma`. It’s usage is somewhat more complicated than “normal” semantic macros, but one thing we can do with it now is  $\magma!$ , which will produce  $\langle U, \circ \rangle$ .

Notably however, the `\magma` macro is already available *within* the `mathstructure` environment as well.

This allows us to provide an `sdefinition` using the semantic macros declared in the structure:

### Example 11

Input:

```
File [sTeX/MathTutorial]algebra/Magma.en.tex
7  \begin{mathstructure}{magma}
8  \symdef{univ}[name=universe,type=\collection]{U}
9  \symdef{op}[name=operation,args=a,assoc=bin,
10 type=\funspace{\univ,\univ}\univ
11 {\argsep{#1}{\mathbin{\maincomp\circ}}}]
12
13 \begin{sdefinition}[for={magma,univ,op}]
14 A \definame{magma} is a \sr{mathstruct}{structure}  $\magma!$ ,
15 where  $\univ$  is a \sn{collection} and  $op!$ 
16 a binary operation  $\funspace{\univ,\univ}\univ$ .
17 \end{sdefinition}
18 \end{mathstructure}
```

Output:

**Definition 3.3.2.** A **magma** is a structure  $\langle U, \circ \rangle$ , where  $U$  is a collection and  $\circ$  a binary operation  $U \times U \rightarrow U$ .

## Instantiating Structures

More importantly however, we can now declare a `variable magma`, using the optional `return=` argument. For example, we can now do

```
\vardef{vM}[name=M,return=\magma]{M}
```

and we get the semantic macro `\vM` with which we can do the following:

Syntax	Result
$\mathcal{M}$	$M$
$\langle \mathcal{M} \rangle$	$\langle U_M, \circ_M \rangle$
$\mathcal{U}_M$	$U_M$
$\circ_M$	$\circ_M$
$a \circ_M b \circ_M c$	$a \circ_M b \circ_M c$

In other words: Given a `symbol` or `variable` with `semantic macro \foo` and `return=\struct`, then `\foo{<fn>}` behaves like the `semantic macro \fn` *within* the `mathstructure environment` for `struct` – but instantiated for the specific instance `foo`.

By default, `STEX` attaches the `symbol's` (or `variable's`) `operator notation` as a subscript suffix to the notation component marked with `\maincomp` – e.g., since the “`\circ`” in the `notation` for `op` is marked with `\maincomp`, doing `\vM{op}{a,b}` ultimately outputs `a \circ_{\vM!} b`. Hence, we get `a \circ_M b`.

We can change the way the `\maincomp notation` component is modified, by using the optional argument `comp=` in the `semantic macro` for the `mathematical structure`. For example, to not change it at all, we can do:

```
\vardef{vM}[name=M,return={\magma[comp=##1]}]{M}
```

...or to suffix it with a `'`, we can do

```
\vardef{vMp}[name=Mp,return={\magma[comp=##1']}] {M'}
```

This allows us to do things like:

```
Let \vM! := \vM{}$ and \vMp! := \vMp{}$ \sns{magma}. Then...
```

yielding

Let  $M := \langle U, \circ \rangle$  and  $M' := \langle U', \circ' \rangle$  magmas. Then...

We can also *assign* fields to (arbitrary) expressions, by doing `name=<tex>` in square brackets. For example we can do the following:

```
\vardef{vA}[type=\collection]{A}
\vardef{vM}[name=M,return={\magma[comp=##1][univ=\vA]}]{M}
\vardef{vMp}[name=Mp,return={\magma[comp={##1}']][univ=\vA]}]{M'}
```

```
Let \vM! := \vM{}$ and \vMp! := \vMp{}$ \sns{magma} on \vA$...
```

Let  $M := \langle A, \circ \rangle$  and  $M' := \langle A, \circ' \rangle$  magmas.

Of course, we can also use `return=` with `variable sequences` – for example:

```
\varseq{vMs}[name=M,return={\magma[comp={##1}_{##1]}],op=(M_i)_1^n}
{1,\ellipses,n}{\maincomp{M}_{##1}}
Let \vMs! := \vMs{i}{_1^n}$ a sequence of \sns{magma}...
```

Let  $(M_i)_1^n := \langle U_i, \circ_i \rangle_1^n$  a sequence of **magnas**...

Note that in the above, it seems that using **#1** in the **return** argument is allowed. Indeed, it is - the **return** statement takes the same arguments as the **semantic macro** itself does and is appropriately instantiated. Since the first (and only) argument to the sequence `\vMs` is the index, when doing `\vMs{i}`... the **#1** in the **return**-statement will be replaced by **i**.

Also, note that if we want to produce  $M_i$  - i.e. the **magma** at index  $i$  in the sequence, we can do `\vMs{i}!`.



Think of the **!** as a “stop sign” - if the expression up to the **!** has an associated presentation, the **!** tells **TeX** to “stop eating arguments” and present whatever it has until now.

### 3.3.2 Extending Structures and Axioms

It is extremely common to “build up” **structures** in a hierarchical manner by adding new fields or axioms: A *semigroup* is an associative magma. A *band* is an idempotent semigroup. A *monoid* is a semigroup with a unit. A *partial order* is an antisymmetric preorder.

We alluded to the fact earlier, that the **mathstructure environment** behaves like an **smodule** - that is literally true: Every **mathstructure** `foo` in a **module** `FooMod` is in fact also a **module** `?FooMod/foo-module`. We can therefore easily extend **structures** using `\importmodule{...?FooMod/foo-module}` - but extending **structures** is so common, and using `\importmodule` tiring, that there is a shortcut: the **extstructure environment**. It takes as second argument a comma-separated list of **structure** names. That allows us to easily define **semigroups**:

#### Example 12

Input:

```

File [sTeX/MathTutorial]algebra/Semigroup.en.tex
8 \begin{extstructure}{semigroup}{magma}
9 \begin{sdefinition}
10 A \definame{semigroup} is a \sn{magma} $\semigroup!$,
11 where \inlineass [name=associative axiom]{
12 \conclusion{\isassociative{$\op!$}}.
13 }
14 \end{sdefinition}
15 \end{extstructure}

```

Output:

**Definition 3.3.3.** A **semigroup** is a **magma**  $\langle U, \circ \rangle$ , where  $\circ$  is **associative**.

Note our usage of `\inlineass` to generate a new **symbol** for the **associative axiom**. If we look at the **OMDOC** tab in the **HTML** preview window, we can see the output in **Figure 27**.

So **MMT** has decided that our statement is an *axiom*.

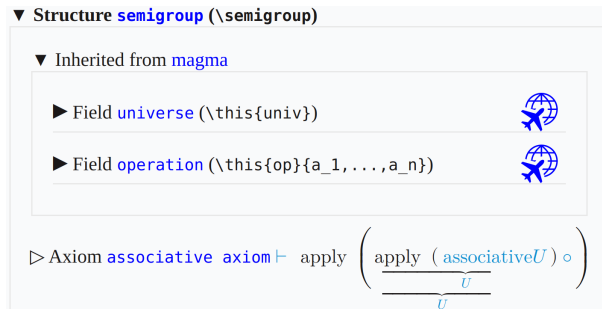


Figure 27: Axioms in OMDoc

### Conservative Extensions

For **structures**, there is a *critical* distinction between *defined* and *undefined symbols*; and analogously between *theorems* and *axioms*.

Remember that **structures** are more like *templates* that are *instantiated* by particular objects. An *undefined* field in a **structure**, in that sense, is like an *obligation*: If something is supposed to be a **semigroup**, it *has to* have a **universe**, an **operation** and the **operation** needs to satisfy the **associative axiom**.

*Defined* fields on the other hand have a *definiens* on the basis of the remaining fields – they don’t need to be explicitly provided for something to instantiate the **structure**; if all the *undefined* fields are provided, the *defined* ones we get “for free”.

The same holds for *theorems*: If a statement is *provable* from the axioms, then we don’t need to explicitly prove it to hold for some particular instance – we have a proof already, provided the axioms hold.

The relation between axioms and theorems is not just analogous to that between undefined and defined **symbols**: It is the very same. Remember the **judgments as types** paradigm?

**STEX** For a **proposition**  $P$ , an assertion in **STEX** induces a **symbol** of **type**  $\vdash P$ . Without a proof, this **symbol** is *undefined* – and hence an *axiom*. A *proof* for  $P$  is a specific term of **type**  $\vdash P$  – i.e. a potential *definiens*. To prove an assertion turns it into a *theorem*, which is to say that the **symbol** can be *defined*.

One consequence of this is: Extending a **structure** only by *defined* fields does not actually (conceptually) introduce a *new structure* – every instance of the old one *should* also be an instance of the new one. The new fields are basically just “syntactic sugar”.

There is a name for extending a **structure** only by defined fields (or theorems): A *conservative extension*.

**STEX** provides the **extstructure\*** environment for that purpose. Unlike **extstructure**, it does *not* take a name (technically, **STEX** generates one internally). Instead, conceptually **extstructure\*** modifies the extended **structure** directly, rather than generating a new **structure**. The caveat however is, that every **symbol** introduced in an **extstructure\*** **must** be defined.

Consider the following conservative extension:



### Example 13

Input:

```
File [sTeX/MathTutorial]algebra/MagmaSquare.en.tex
7 \begin{extstructure*}{magma}
8 \begin{sdefinition}[macro=sq,args=1]
9 \notation{sq}[op=\cdot^2][\comp 2]
10 \vardef{va}[name=a,type=\univ,bind]{a}
11 Let $\inset{va}{\univ}$. We define
12 $\defnotation{sq{va}} := \definiens{op{va,va}}$.
13 \end{sdefinition}
14 \end{extstructure*}
```

Output:

```
Definition 3.3.4. Let  $a \in U$ . We define  $a^2 := a \circ a$ .
```

Via `\definiens`, the new symbol `sq` is now *defined* (note the `macro=` argument, taht generates a *semantic macro* as well). Whenever we import the containing *module*, we now have an additional field `sq` in (any extension of) `magma` – e.g., as `semigroup` extends `magma`, the following is now valid:

```
\usemodule [sTeX/MathTutorial]{algebra?MagmaSquare}
\vardef{vsg}[name=S,return=\semigroup]{S}
$\vsg{sq}{a}$
```

...producing  $a^2$ .

### 3.3.3 Nesting Structures and `\this`

A perhaps not too surprising, but a notable aspect of *structures* is that fields themselves can be instances. This is important for example for implementing *vector spaces*, but can also be used to bundle things that are not normally thought of as *structures*, such as objects with certain defining properties.

Take as an example, the notion of a (*magma*) *homomorphism*:

```
Definition 3.3.5. Let  $M_1 = \langle U_1, \circ_1 \rangle$  and  $M_2 = \langle U_2, \circ_2 \rangle$  magmas. A magma homomorphism is a function  $F : U_1 \rightarrow U_2$  such that  $F(a \circ_1 b) = F(a) \circ_2 F(b)$  for all  $a, b \in U_1$ .
```

So a *homomorphism* is a *function* with certain properties. And *structures* can be used to “bundle” the *function* itself with both the *magma* on whose universes the *function* operates, as well as the *axiom* that *makes* it a *homomorphism*. After all, considered as a mere *function*,  $F : U_1 \rightarrow U_2$  contains no information about the operation with respect to which it is homomorphic.

The first thing to note is that we can provide *mathstructure* with an optional argument for a *name* distinct from the name of its *semantic macro*. We then add two fields that *return magmas*. So far, so unexciting:

```
\begin{mathstructure}{magmahom}[magma homomorphism]
\symdef{dom}[name=domain,return={\magma[comp={##1}_1]}]{M_1}
\symdef{cod}[name=codomain,return={\magma[comp={##1}_2]}]{M_2}
```

For the `function` itself, we know how to give it a meaningful `type`, already:

```
\symdef{f}[type=\funspace{\dom{univ}}{\cod{univ}},args=1]{???
```

...but what should its `notation` be? Ideally we would want it to just be the `notation` of whatever particular instance it is – in informal mathematics, we rarely distinguish notationally between a `homomorphism` and its underlying `function` (to the point where it's not clear, whether *informally* the distinction is even meaningful). Similarly, we rarely distinguish e.g. between a `magma` (or semigroup, monoid, group, ring, vector space,...) and its underlying universe.

This is where `\this` comes into play (pun intended). Within an `mathstructure` or `exstructure`, or in the context of a particular instance of one, `\this` represents “the” instance.

We can set it in the context of `mathstructure` as a further optional argument; e.g.

```
\begin{mathstructure}{magmahom}[magma homomorphism,this=F]
```

and then use `\this` in the `notation` for the `function`. We can further provide the `homomorphism condition` as an axiom using `\inlineass`:

#### Example 14

Input:

```
File [sTeX/MathTutorial]algebra/Homomorphism.en.tex
9 \begin{mathstructure}{magmahom}[magma homomorphism,this=F]
10 \symdef{dom}[name=domain,return={\magma[comp={##1}_1]}]{M_1}
11 \symdef{cod}[name=codomain,return={\magma[comp={##1}_2]}]{M_2}
12 \symdef{f}[op=\this,args=1,
13 type=\funspace{\dom{univ}}{\cod{univ}}
14 ]{\this \dobrackets{#1}}
15
16 \begin{sdefinition}[for={magmahom,dom,cod,f}]
17 \vardef{va}[name=a,type=\dom{univ}]{a}
18 \vardef{vb}[name=b,type=\dom{univ}]{b}
19 Let $\dom!=\dom{}$ and $\cod!=\cod{}$ \sns{magma}.
20 A \definame{magmahom} is a function
21 $\fun{f!}{\dom{univ}}{\cod{univ}}$ such that
22 \inlineass[name=homomorphism condition]{\conclusion{\forall{
23 \arg[2]{\eq{
24 \f{\dom{op}}{\va,\vb}}, \cod{op}{\f{\va},\f{\vb}}
25 }}$ \comp{for all} $\arg[1]{\inset{\va,\vb}{\dom{univ}}}$$.
26 }}}
27 \end{sdefinition}
28 \end{mathstructure}
```

Output:

**Definition 3.3.6.** Let  $M_1 = \langle U_1, \circ_1 \rangle$  and  $M_2 = \langle U_2, \circ_2 \rangle$  magmas. A **magma homomorphism** is a function  $F : U_1 \rightarrow U_2$  such that  $F(a \circ_1 b) = F(a) \circ_2 F(b)$  for all  $a, b \in U_1$ .

Now if we instantiate our `magma homomorphism`:

```
\vardef{vh}[name=H,return={\magmahom[this=H]}]{H}
```

Here is a list of what we can do now:

Syntax	Result
$\backslash\text{vh}!$	$H$
$\backslash\text{vh}\{ \}$	$\langle M_1, M_2, H \rangle$
$\backslash\text{vh}\{f\}!$	$H$
$\backslash\text{vh}\{f\}\{a\}$	$H(a)$
$\backslash\text{vh}\{\text{dom}\}!$	$M_1$
$\backslash\text{vh}\{\text{cod}\}\{ \}$	$\langle U_2, \circ_2 \rangle$
$\backslash\text{vh}\{\text{cod}\}\{\text{univ}\}$	$U_2$
$\backslash\text{vh}\{\text{dom}\}\{\text{op}\}!$	$\circ_1$
$\backslash\text{vh}\{\text{cod}\}\{\text{op}\}\{a, b, c\}$	$a \circ_2 b \circ_2 c$

Note how – as one would expect – we can treat  $\backslash\text{vh}\{\text{dom}\}$  and  $\backslash\text{vh}\{\text{cod}\}$  like any other instance of [magma](#).



Note that some of the outputs in the above table are probably not quite what we want. Determining the precise typesetting of an expression involving *nested paths* of fields is difficult, to say the least (e.g., what exactly should  $\backslash\text{this}$  refer to in a deeply nested sequence of fields?).

Using instances within [structures](#) is still very useful; at the very least when defining [structures](#). When subsequently *using* [structures](#), however, accessing fields of fields (of fields (of ...)) of an instance should be avoided.

Luckily, there is rarely a need for doing so – in practice, those fields we might want to access in such a way, we usually also want to provide specific [notations](#) and talk about independently of the “containing” instance, such that introducing a new [variable](#) (or [symbol](#)), and assigning the corresponding field to that [variable](#), makes considerably more sense. And subsequently using the [variable](#) is easier than concatenating  $\{ \dots \}$ , too.

### 3.4 Complex Inheritance and Theory Morphisms



We are starting to approach seriously experimental territory.

While the theory behind all the following is relatively well understood, and their implementation in [MMT](#) is mature, the same can not be said out the implementation in [S<sub>TE</sub>X](#).

There are still kinks to be ironed out, but feel free to experiment.

We now have all the tools available to progress towards something more interesting. Here is a list of documents with respective [modules](#) and [symbols](#) we will build on in the following:

[sTeX/MathTutorial]props/Idempotent.en.tex

**Definition 3.4.1.** Let  $e \in A$  and  $\circ : A \times A \rightarrow A$ .  $e$  is called **idempotent** with respect to  $\circ$ , if  $e \circ e = e$ .

**Definition 3.4.2.** The operation  $\circ : A \times A \rightarrow A$  is called **idempotent**, if **every** element  $a \in A$  is **idempotent** with respect to  $\circ$ .

[sTeX/MathTutorial]props/Distributive.en.tex

**Definition 3.4.3.** Let  $\odot : B \times A \rightarrow A$  and  $\oplus : A \times A \rightarrow A$ . We say  $\odot$  **distributes over**  $\oplus$ , if  $b \odot (a_1 \oplus a_2) = (b \odot a_1) \oplus (b \odot a_2)$  for all  $a_1, a_2 \in A$  and  $b \in B$ .

[sTeX/MathTutorial]props/Absorption.en.tex

**Definition 3.4.4.** Let  $\odot : A \times B \rightarrow A$  and  $\oplus : A \times B \rightarrow B$ . We say  $\odot$  **absorbs**  $\oplus$ , if  $a_1 \odot (a_1 \oplus b) = a_1$  for all  $a_1 \in A$  and  $b \in B$ .

[sTeX/MathTutorial]algebra/Band.en.tex

**Definition 3.4.5.** A **band** is an **idempotent semigroup**.

[sTeX/MathTutorial]algebra/Semilattice.en.tex

**Definition 3.4.6.** A **semilattice** is a **commutative band**.

[sTeX/MathTutorial]props/Reflexive.en.tex

**Definition 3.4.7.** A binary relation  $R$  on  $A$  is called **reflexive**, if  $R(a, a)$  for all  $a \in A$ .

[sTeX/MathTutorial]props/Symmetric.en.tex

**Definition 3.4.8.** A binary relation  $R$  on  $A$  is called **symmetric**, if  $R(a, b)$  implies  $R(b, a)$  for all  $a, b \in A$ .

[sTeX/MathTutorial]props/Transitive.en.tex

**Definition 3.4.9.** A binary relation  $R$  on  $A$  is called **transitive**, if  $R(a, b)$  and  $R(b, c)$  implies  $R(a, c)$  for all  $a, b, c \in A$ .

[sTeX/MathTutorial]props/Antisymmetric.en.tex

**Definition 3.4.10.** A binary relation  $R$  on  $A$  is called **antisymmetric**, if  $R(a, b)$  and  $R(b, a)$  implies  $a = b$  for all  $a, b \in A$ .

[sTeX/MathTutorial]orders/Graph.en.tex

**Definition 3.4.11.** A **directed graph** is a **structure**  $\langle U, R \rangle$ , where  $U$  is a **collection** and  $R$  a binary relation on  $U$ .

**Definition 3.4.12.** An **(undirected) graph** is a **directed graph**  $\langle U, R \rangle$ , where  $R$  is **symmetric**.

[sTeX/MathTutorial]orders/Preorder.en.tex

**Definition 3.4.13.** A structure  $\langle U, \leq \rangle$  is called a **preorder** (or **quasiorder**, or **preordered set**; in short **proset**), if  $\leq$  is **reflexive** and **transitive**.

[sTeX/MathTutorial]orders/Poset.en.tex

**Definition 3.4.14.** A preorder  $\langle U, \leq \rangle$  is called a **partial order** (or **poset**), if  $\leq$  is **antisymmetric**.

[sTeX/MathTutorial]orders/InfSup.en.tex

**Definition 3.4.15.** Let  $\langle U, \leq \rangle$  a **poset**. An element  $a \in U$  is called an **infimum** or **greatest lower bound** of  $x_1$  and  $x_2$ , if  $a \leq x_1$ ,  $a \leq x_2$ , and for any  $x$  with  $x \leq x_1$  and  $x \leq x_2$ , we have  $x \leq a$ .

**Definition 3.4.16.** Let  $\langle U, \leq \rangle$  a **poset**. An element  $a \in U$  is called a **supremum** or **least upper bound** of  $x_1$  and  $x_2$ , if  $x_1 \leq a$ ,  $x_2 \leq a$ , and for any  $x$  with  $x_1 \leq x$  and  $x_2 \leq x$ , we have  $a \leq x$ .



Note that **infima** and **suprema** are more generally defined on *sets* of elements. Doing so in **sTeX** is significantly more complicated *for now*, and will require some amount of research to make convenient – especially if we want to subsequently define *operators* on pairs of elements, as below. We therefore opt for the simpler version where it is defined as binary from the get go.

[sTeX/MathTutorial]orders/MeetJoinSemilattice.en.tex

**Definition 3.4.17.** A **poset**  $\langle U, \leq \rangle$  is called a **meet semilattice** if for every two elements  $a, b$  the **infimum**  $a \wedge b$  exists.

**Definition 3.4.18.** A **poset**  $\langle U, \leq \rangle$  is called a **join semilattice** if for every two elements  $a, b$  the **supremum**  $a \vee b$  exists.

**Definition 3.4.19.** An **(order) semilattice** is a **meet** and **join semilattice**.

### Exercise

Try to implement all of the above yourself!

## 3.4.1 Glueing Structures Together

We now want to progress towards **lattices**, i.e. the following:

**Definition 3.4.20.** A **lattice** is a structure  $\langle U, \wedge, \vee \rangle$  such that  $\langle U, \wedge \rangle$  and  $\langle U, \vee \rangle$  are **semilattices**, and  $\vee$  **absorbs**  $\wedge$  and vice versa; i.e.  $a \vee (a \wedge b) = a$  and  $a \wedge (a \vee b) = a$ . The operations  $\wedge$  and  $\vee$  are called **meet** and **join**, respectively.

So we make a new `module`, open an `extstructure environment` and... realize two problems:

1. We can't just extend `semilattice`: We need *two* copies of `semilattice` that share a universe, and importing `semilattice` twice is of course redundant.
2. We also want to *rename* the operations of the two `semilattices` to be subsequently called `join` and `meet`.

What we need is a way to *inherit* from `semilattice` while a) *modifying* the `symbols` therein, and b) not be `idempotent` – i.e. two imports from the same `structure` or `module` should not be identified. We can do that with the `\copymod macro`, which takes three arguments:

1. A *name* for the copy,
2. the `structure` or `module` to copy, and
3. a comma-separated list of renamings and redefinitions of the `symbol`.  $\langle symbol \rangle = \langle def \rangle$  redefines  $\langle symbol \rangle$ ,  $\langle symbol \rangle @ \langle newname \rangle$  renames it,  $\langle symbol \rangle = \langle def \rangle @ \langle newname \rangle$  (or  $\langle symbol \rangle @ \langle newname \rangle = \langle def \rangle$ ) does both.

In our case, we want two copies of `semilattice`, which we will call `meets1` and `joins1`. In the first copy, we only want to rename `op` to `meet`. In the second, we want to rename `op` to `join`, and *also* redefine the universe to be the one from `meets1`:

```
\copymod{meets1}{semilattice}{
  op @ meet
}
\copymod{joins1}{semilattice}{
  univ = \univ,
  op @ join
}
```

You might have already noticed some problem with that – which of the two universes does `\univ` refer to now? (They are *defined* as equal, but `LATEX` does not know that!) Or which of the two `commutative axioms` does “`commutative axiom`” refer to now? Everything is ambiguous now!

Not really - if you have wondered why the `\copymod` takes a *name* as argument: The name is prefixed to every `symbol` name. Hence, the `universe` in `joins1` is now called `joins1/universe`, and the one in `meets1` is called `meets1/universe`. Furthermore, `\copymod` by default generates no `semantic macros` for any of the imported `symbols` – except for those renamed with `@`. In fact, what the `@` syntax actually does, is to generate a `semantic macro` by that name. If we want to change the *name* (that is shown when using `\symname` et al), we add that new name in square brackets. Hence, what we really want to do is:

```
\copymod{meets1}{semilattice}{
  univ @ univ,
  op @ [meet]meet
}
\copymod{joins1}{semilattice}{
  univ = \univ,
  op @ [join]join
}
```

This now gives us two copies of `semilattice`, generates semantic macros `\univ` for `meetsl/universe`, `\meet` for `meetsl/op` and `\join` for `joinsl/op`, and renames `meetsl/op` to `meet` and `joinsl/op` to `join`.

That allows us to then add the `absorption` axioms, an `sdefinition` for `lattice` and subsequently `\lattice!` produces  $\langle U, \wedge, \vee \rangle$ , with all axioms inherited (see `[sTeX/MathTutorial]algebra/Lattice.en.tex`).

### 3.4.2 Realizations

A very common situation we find in connection with `mathematical structures` is that “every *this* is a *that*” (or the concrete case “*this* is a *that*”).

With what we did so far, we are in this situation regarding the algebraic definition of `semilattices` and the order-theoretic one (exemplary `meet semilattice`).

In `MMT` parlance, this corresponds to a `total (implicit) theory morphism` from “that” to “this”.

In `sTeX` words, we want to inherit from “that” by assigning all the `symbols` in “that” to concrete terms. In our case:

`[sTeX/MathTutorial]algebra/SemiLatticeOrder.en.tex`

**Definition 3.4.21.** Let  $\langle U, \circ \rangle$  a `semilattice`. We let  $a \leq b$  iff  $a \circ b = a$ .

**Theorem 3.4.22.**  $\langle U, \leq \rangle$  is a `meet semilattice`.

*Proof:* We need to prove the following

**reflexivity**  $a \leq a$ : We need to show  $a \circ a = a$ . Follows from the `idempotent axiom`.

**antisymmetry**  $a \leq b$  and  $b \leq a$  implies  $a = b$ : Assume  $a \circ b = a$  and  $b \circ a = b = a \circ b$  (by the `commutative axiom`). Hence,  $a = b$

**transitivity** If  $a \leq b$  and  $b \leq c$ , then  $a \leq c$ . : Assume  $a \circ b = a$  and  $b \circ c = b$ . Then  $a \circ c = (a \circ b) \circ c = a \circ (b \circ c) = a \circ b = a$ . Hence,  $a \leq c$ .

$a \circ b$  is the `infimum` of  $\{a, b\}$ : By definition (and the `commutative axiom`),  $a \circ b \leq a$  and  $a \circ b \leq b$ . We need to show, that if  $x \leq a$  and  $x \leq b$ , then  $x \leq a \circ b$ . Assume  $x \circ a = x$  and  $x \circ b = x$ . Then  $x \circ (a \circ b) = (x \circ a) \circ b = x \circ b = x$ . Hence  $x \leq a \circ b$

So to be precise, we want to provide *definiencia* for all undefined `symbols` in `meet semilattice` (i.e. the `relation` and `meet`) and *proofs* for all *axioms* (`reflexive axiom`, `antisymmetric axiom`, `transitive axiom`, and `infimum axiom`), and by so obtain the fact that every `semilattice` is a `meet semilattice`.

For that purpose, we have the `\realize macro`. It behaves like `\copymod`, but does not take a name, and additionally requires that all undefined fields get assigned. So we could do the following:

#### Example 15

Input:

File [sTeX/MathTutorial]algebra/SemiLatticeOrder1.en.tex

```

8 \begin{extstructure*}{semilattice}
9 \realize{meetsl}{
10 univ = \univ,
11 meet = \op!,
12 rel @ [order]order = \map{a,b}{\eq{\op{a,b},a}},
13 reflexive axiom = trivial,
14 transitive axiom = trivial,
15 antisymmetric axiom = trivial,
16 infimum axiom = trivial
17 }
18 \end{extstructure*}
19
20 \vardef{mysl}[return=\semilattice]{S}
21 $\mysl{order}{a,b} \quad \mysl{}[univ,op,order]$

```

Output:

$$a \leq_S b \quad \langle U_S, \circ_S, \leq_S \rangle$$

As we can see, we can now access the field `order`, which is renamed from `relation` in `meet semilattice` and also has the desired definiens in `MMT`. But of course this approach is very “declarative”: We do all the assigning in one `macro`, rather than narratively as what they *should* be: definitions and proofs.

If we want to achieve the more narrative version at the beginning of the subsection, we can use the `realization environment` instead. It behaves like the `\realize macro`, but allows us to do the assignments and renamings individually somewhere in the body of the `environment`, interleaved with arbitrary text. Additionally, within the `environment`, all `sTeX` features that introduce *definiencia* (like the `\definiens macro`) induce assignments instead.

To declaratively rename or assign fields, we can then use the `\assign` and `\renamedecl` macros instead. That allows us to do the following instead:

```

\begin{realization}{meetsl}
\assign{univ}{\univ}
\assign{meet}{\op!}
\renamedecl{rel}[order]{order}
...

```

...and then use text to do the remaining assignments. For example, we can use the `sdefinition environment` to assign `rel` to the desired definiens:

```

\usestructure{meetsl}
\begin{sdefinition}[for=order]
\varbind{va,vb}
Let $\semilattice![univ,op]$ a \sn{semilattice}.
We let $\rel{\va,\vb}$
iff $\definiens{\eq{\op{\va,\vb},\va}}$.
\end{sdefinition}

```

And now `sTeX` will use the `\definiens` to assign  $a, b \mapsto a \circ b = a$  to the `relation` of `meet semilattice`.

Analogously, we can use the `sproof` and `subproof` environments to produce “definiencia” (i.e. proofs) for the axioms (see [sTeX/MathTutorial]algebra/SemiLatticeOrder.en.tex)



## Chapter 4

# Extensions for Education

The last two chapters have shown generic markup and semantization facilities in `STEX`. As said before, investments in semantic markup pay off, iff the impact of a document is high, e.g. if there are many more readers than authors or if the semantic services afforded by the semantic markup can help reduce the help readers need to understand the material.

Educational documents constitute one category of high-impact documents which are supported by the `STEX` ecosystem, we will cover them here. In fact, educational documents have been one of the initial document categories `STEX` has been developed for. The idea is that if we can mark up the meaning and didactic role of learning objects, we can base learning support services on that and embed them into the documents.

Another reason educational documents are particularly interesting is that in a sense all academic communication is educational, as all documents try to “teach” the reader new concepts and results.

Concretely, we cover a document class for combining slides and course notes ([section 4.1](#)) and functionality for marking up problems and exercises ([section 4.2](#)) and for marking up homework assignments and exams ([section 4.3](#)).

### 4.1 Slides and Course Notes

`TODO`<sup>6</sup>

#### Contents

### 4.2 Problems and Exercises

#### 4.2.1 Background

Problems/exercises are text fragments that contain a task assigned to the learner – e.g. computing the value of a specified quantity, simplifying an expression, modeling a described situation in a mathematical structure, or judging the veracity of a given statement. Problems/exercises are used in very different contexts, in

---

<sup>6</sup>`TODO: notesslides.sty`

- *homework assignments and formal exams*, where they are graded and points are awarded for course credit,
- *self-study materials* that allow learners explore applications of some concepts and/or practice,
- *automated tutoring systems* to determine learner competencies to trigger computer supported learning services.

In all of these contexts, diagnosis the correctness or suitability of an answer plays a great role, e.g. for grading, the generation of feedback, or the suggestion of remedial actions that may “heal” any diagnosed competency gaps or misconceptions. To support that we might want to specify “answer classes” that classify answers received for automating/triggering feedback and grading.

There are various types of problems/exercises in practical use. They are mainly distinguished by how they support diagnosis of student answers: there are

- open problems, where expected answers are free-form, and classification into answer classes is usually a manual process; see [subsection 4.2.3](#)
- single/multiple choice questions where the answer classes and thus feedback/grading correspond to the selection pattern of items; see [subsection 4.2.5](#).
- fill-in-the-blanks questions where we may want to specify how the answer string is interpreted; see [subsection 4.2.6](#).

Additionally, Problems and exercises often come with text fragments that serve auxiliary functions: hints, notes, and and grading specifications. Furthermore, we can specify how long solving a given problem is estimated to take and how many points will be awarded for a perfect solution – e.g. in an exam. See [subsection 4.2.9](#) for details.

## 4.2.2 The Package, Options, and Configuration

The `problem` package provides functionality for marking up problems and exercises as semantic sources from which the presentations for the various contexts can be generated. E.g. documents without solutions for paper or online exams, and the corresponding exams with master solutions for exam reviews. Similarly with/without hints, or points. Their visibility is specified in the options of the `problem` package, which can be used in any L<sup>A</sup>T<sub>E</sub>X class. The following is a typical preamble for a problem file:

```
\documentclass [lang=de] {article}
\usepackage [solutions,hints,pts,min] {problem}
```

Here we have specified the options `solutions` (solutions should be shown), `hints` (hints should be given), `pts` (display the points awarded for solving the problem?), `min` (display the estimated minutes for problem solving). Leaving out the options would make the corresponding functionality invisible.

The `problem` package generates content and labels according to the language specified in the `lang` key of the main document<sup>1</sup>, these can be localized by in the files `ldf/problem-<lang>.ldf`<sup>7</sup>.

<sup>1</sup>EDNOTE: MK: document the attribute somewhere

<sup>7</sup>The current crop of language definition files is quite limited. Please feel free to contribute the to the localization effort

The `problem` package also supplies the `test` option, which specifies that the content is intended for use in an assessment situation, where certain additional content (e.g. exam legalese) should be shown while other content (e.g. solutions) should not be.

### 4.2.3 (Open) Problems

The main environment provided by the `problem` package is (surprise surprise) the `sproblem` environment. It is used to mark up problems and exercises. The following example shows the main functionality:

#### Example 16

Input:

```

File [sTeX/Documentation]tutorial/ext/simple-problem.en.tex
1 \begin{document}
2 \begin{sproblem}[id=prob.elefants,pts=10,min=2,title=Fitting Elefants]
3   How many Elefants can you fit into a Volkswagen beetle?
4   \begin{hint}
5     Think positively, this is simple!
6   \end{hint}
7   \begin{exnote}
8     Justify your answer
9   \end{exnote}
10  \begin{gnote}
11    if they do not give the justification deduct 5 pts
12  \end{gnote}
13  \begin{solution}
14    Four, two in the front seats, and two in the back.
15  \end{solution}

```

Output:

```

Exercise (Fitting Elefants)
How many Elefants can you fit into a Volkswagen beetle?
-----
Lösung: Four, two in the front seats, and two in the back.
-----

```

The `sproblem` environment takes an optional key/value 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.

The `sproblem` can contain several `solution` environments, which take the well-known `id`, `style`, and `title` attributes, and also the `testspace` and `answerclass`. The former

The additional functionality is specified in the `hint` `exnote` (notes in exercises), `solution`, and `gnote` (grading notes) environments. Here, the first three are shown whereas the grading notes are hidden, since the corresponding option was not given in the `\usepackage[...]{problem}`. All of these environments can occur any number

of times in the `sproblem` environment. The `solution` environment takes an optional argument that is interpreted as the identifier.

#### 4.2.4 Structured Problems

Problems can be structured into subproblems via the `subproblem` environment. It takes the same arguments and allows the same content model as `sproblem`.

##### Example 17

Input:

```

File [sTeX/Documentation]tutorial/ext/structured-problem.en.tex
1 \begin{document}
2 \begin{sproblem}[id=prob.elefants-mod,title=Fitting Elefants Modularly]
3 Consider the problem of fitting elephants into a Volkswagen beetle.
4 \begin{subproblem}[pts=1,min=2]
5 Estimate the number of elephants on the front seats.
6 \begin{solution}
7 Two on each side one.
8 \end{solution}
9 \end{subproblem}
10 \begin{subproblem}[pts=1,min=2]
11 Estimate the number of elephants on the back seats.
12 \begin{solution}
13 Three little elephants.
14 \end{solution}
15 \end{subproblem}

```

Output:

**Exercise (Fitting Elefants Modularly)**

Consider the problem of fitting elephants into a Volkswagen beetle.

- Estimate the number of elephants on the front seats.

---

*Lösung:* Two on each side one.

---
- Estimate the number of elephants on the back seats.

---

*Lösung:* Three little elephants.

---
- What is the total number?

---

*Lösung:* A family of five; we do not want elephants in the frunk.

---

By default, subproblems are numbered subordinate to the “mother problem”. The `problem` package does not make any assumptions about whether subproblems can be used independently from their sister problems, or on order requirements.

## 4.2.5 Single/Multiple Choice Blocks

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

---

`\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 correct answers, `F` (the default value) for false ones,
- `Ttext` the verdict for correct 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 18

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} \mcc[F,feedback=that is for C and C++] {function}
6     \mcc[F,feedback=that is for Standard ML] {fun}
7     \mcc[F,Ftext=Noooooooooooo,feedback=that is for Java] {public static void}
8   \end{mcb}
9 \end{sproblem}
```

Output:

#### Exercise (Functions)

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

- def<sup>a</sup>
- function<sup>b</sup>
- fun<sup>c</sup>
- public static void<sup>d</sup>

---

<sup>a</sup>Korrekt

<sup>b</sup>Falsch

*that is for C and C++*

<sup>c</sup>Falsch

*that is for Standard ML*

<sup>d</sup>Noooooooooooo

*that is for Java*

In “exam mode” where disable solutions (here via `\stopsolutions`) we get the questions without solutions (that is what the students see during the exam/quiz).

### Example 19

Input:

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

Output:

#### Exercise (Functions)

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

- def
- function
- fun
- public static void

What we have seen is the default rendering of the multiple choice block, which is as an itemize like environment with check boxes. There are alternatives to that which we can choose with the `style` key in the optional attribute of the `mcb` environment. Currently the only alternative is `inline` style:

### Example 20

Input:

```
1 \startsolutions
2 \begin{sproblem}[title=Functions,name=functions1]
3   What is the keyword to introduce a function definition in python?
4
5   \begin{mcb}[style=inline]
6     \mcc[T]{def} \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:

**Exercise (Functions)**

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

def<sup>a</sup>  function<sup>b</sup>  fun<sup>c</sup>  public static void<sup>d</sup>

<sup>a</sup>Korrekt

<sup>b</sup>Falsch

*that is for C and C++*

<sup>c</sup>Falsch

*that is for Standard ML*

<sup>d</sup>Noooooooooo

*that is for Java*

This is the solutions mode, i.e. with solutions, otherwise, the footnotes will fully disappear.

Analogously, single choice blocks are marked up by the `scc` environment and the individual choices by the `\scc` macro. In PDF, there is little difference to the multiple choice case. In interactive formats like HTML, single choice blocks are typically realized via radio buttons to enforce single choice.<sup>2</sup>

#### 4.2.6 Filling-In Concrete Solutions

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

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

##### Example 21

Input:

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

Output:

**Exercise (Fitting Elefants)**

How many Elefants can you fit into a Volkswagen beetle?

and the actual solution in solutions mode:

##### Example 22

Input:

<sup>2</sup>EdNOTE: MK: are there any differences between `mcc` and `scc` we need to discuss here?

```

1 \startsolutions
2 \begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
3   How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
4 \end{sproblem}

```

Output:

**Exercise (Fitting Elefants)**  
 How many Elefants can you fit into a Volkswagen beetle? 4

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>8</sup>

#### 4.2.7 Answer Classes

3.

#### 4.2.8 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 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.

#### 4.2.9 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

<sup>8</sup>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.

<sup>3</sup>EdNOTE: MK: describe them



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`      The solution environment takes an **TODO**<sup>9</sup>

`\testemptypage`      solution

## 4.3 Homework Assignments and Exams

### 4.3.1 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.

### 4.3.2 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 for a description of the intended behavior).

`hints`

`gnotes`

`pts`

`min`

---

`multiple`      Furthermore, the `hwexam` package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

`test`      Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the `LATEX` source.

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

---

<sup>9</sup>**TODO: check what is still undescribed `problem.sty` and make examples for it.**

### 4.3.4 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.

### 4.3.5 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.

```
reqpts1 \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)

2023-10-13

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

Write the solutions to the sheet.

The estimated time for solving this exam is 23 minutes, leaving you 37 minutes for revising your exam.

You can reach 23 points if you solve all problems. You will only need 27 points for a perfect score, i.e. -4 points are bonus points.

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.	4.1	1.1	1.2	2.1	2.2	2.3	2.4	2.5	2.6	
total	0	0	0	10	0	0	0	0	0	
reached										

good luck

10

<sup>10</sup>MK: The first three “problems” come from the stex examples above, how do we get rid of this?

## Part II

# User Manual

*The dynamic [HTML](https://stexmt.mathhub.info/sTeX/fullhtml?archive=sTeX/Documentation&filepath=manual.xhtml) version of this part can be found at*  
*<https://stexmt.mathhub.info/sTeX/fullhtml?archive=sTeX/Documentation&filepath=manual.xhtml>*

# Chapter 5

## Basics

### 5.1 Package and Class Options

- `debug=<prefixes>`: (see Developer Manual)
- `lang=<languages>`: If set, `STEX` will load the `babel` package with the provided languages. Supported languages (currently) are:

<code>ar</code>	arabic
<code>bg</code>	bulgarian
<code>de</code>	german (with option <code>ngerman</code> )
<code>en</code>	english
<code>fi</code>	finnish
<code>fr</code>	french
<code>ro</code>	romanian
<code>ru</code>	russian
<code>tr</code>	turkish (with option <code>shorthands=!</code> )

- `mathhub=<path>`: Uses the provided file path as MathHub directory (see [section 5.2](#)).
- `usesms/writesms`: If `writesms` is set, content loaded from external [math archives](#) (i.e. [modules](#)) is persisted in the file `\jobname.sms`.

If `usesms` is set, the content of the `.sms`-file is loaded, obviating the need to reprocess the original files.

The options are not mutually exclusive, but care should be taken if dependencies have changed between builds.

This offers two advantages:

1. If a document has many (transitive) dependencies, `usesms` should significantly speed up the build process, and
2. setting `usesms` allows for distributing the `.sms`-file to make the document *standalone*, allowing for compilation without needing imported/used modules to be present.

The options `debug`, `mathhub`, `usesms` and `writesms` can also be set by the environment variables `STEX_DEBUG`, `MATHHUB`, `STEX_USESMS` and `STEX_WRITESMS`. In fact, the `MATHHUB` environment variable is the recommended way to set the `MathHub` directory. This is the only option where the *package option* overrides the environment variable.

The environment variables for `USE/WRITESMS` are particularly useful, in that they allow for convenient compilation workflows. For example, the `Build PDF/XHTML/OMDoc` button in the `IDE` does the following:

```
STEX_WRITESMS=true pdflatex <job>.tex
[bibtex|biber] <job>
STEX_USESMS=true pdflatex <job>.tex
STEX_USESMS=true pdflatex <job>.tex
```

Guaranteeing (in the first run) that all dependencies are loaded from their respective sources and persisted, and in the two subsequent runs read from the generated `.sms` file, (likely) speeding up the subsequent runs significantly.

## 5.2 Math Archives and the MathHub Directory

`STEX` uses `math archives` to organize document content modularly, without a user having to specify absolute paths, which would differ between users and machines.

All `STEX` archives need to exist in the local `MathHub`-directory. `STEX` knows where this folder is via one of five means:

1. If the `STEX` package is loaded with the option `mathhub=/path/to/mathhub`, then `STEX` will consider `/path/to/mathhub` as the local `MathHub` directory.
2. If the `mathhub` package option is *not* set, but the macro `\mathhub` exists when the `STEX`-package is loaded, then this macro is assumed to point to the local `MathHub` directory; i.e. `\def\mathhub{/path/to/mathhub}\usepackage{stex}` will set the `MathHub` directory as `path/to/mathhub`.
3. Otherwise, `STEX` will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local `MathHub` directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.
4. If that too fails, `STEX` will look for a file `~/stex/mathhub.path`. If this file exists, `STEX` 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, and is used by the `IDE` setup.
5. Finally, if all else fails, `STEX` considers `~/MathHub` to be the `MathHub` directory.

The `STEX IDE` allows you to directly download `math archives` from `gl.mathhub.info` – currently available `archives` there are:

- `sTeX/*` – a group of semi-experimental documents showcasing `STEX3.3` features,
- `smglom/*` – a vast collection of multilingual `modules` of concepts in mathematics and computer science. The `SMGloM` predates `STEX3` and is thus largely “underannotated” with respect to (formal) semantics,
- `MiKoMH/*` – a vast collection of lecture slides and notes in computer science for courses held by Michael Kohlhase. They largely make use of `SMGloM modules`.

## 5.2.1 The Structure of Math Archives

An **archive** group/name is stored in the directory `<MathHub>/group/name`; e.g. assuming your local MathHub-directory is set as `/user/foo/MathHub`, then in order for the `sTeX/Documentation` archive to be found by the `sTeX` system, it needs to be in `/user/foo/MathHub/sTeX/Documentation`.

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 discussed in [section 5.3](#).

## 5.2.2 MANIFEST.MF-Files

The `MANIFEST.MF` in the `META-INF` directory consists of key-value-pairs, informing `sTeX` (and associated software, e.g. `MMT`) of various properties of an **archive**. For example, the `MANIFEST.MF` of the `sTeX/Documentation` archive looks like this:

```
id: sTeX/Documentation
ns: http://mathhub.info/sTeX/Documentation
narration-base: http://mathhub.info/sTeX/Documentation
format: stex
title: The sTeX Documentation
teaser: The full Documentation for the sTeX system
url-base: https://stexmt.mathhub.info/:sTeX
dependencies: sTeX/ComputerScience/Software, sTeX/MathTutorial
ignore: */code/*|*/tikz/*|*/tutorial/solution/*
```

Many of these are in fact ignored by `sTeX`, but some are important:

**id:** The name of the **archive**, including its group (e.g. `sTeX/Document`). This is used by the `MMT` system in favor of the directory, but `TeX`'s limited access to the file system enforces the directory structure.

**source-base** or

**ns:** The namespace from which all **symbol** and **module MMT-URIs** in this **archive** are formed.

**narration-base:** The namespace from which all document **MMT-URI** in this repository are formed. It can safely match the `ns`-field.

**url-base:** A URL that is formed as a basis for *external references*. and hyperlinks. An `MMT` (or comparable system) instance should run there and host (`sTeX`-generated) **HTML**.

**dependencies:** All **archives** that this **archive** depends on. `sTeX` ignores this field, but `MMT` can pick up on them to resolve dependencies, e.g. when downloading **archives** in the **IDE**, which will also download all dependencies.

**ignore:** A regular expression of `.tex` files in the **source** directory that should be ignored; e.g. they will not be compiled when building a whole directory or archive in the **IDE**.

## 5.3 The lib-Directory

A [math archive](#) group/archive may have a `lib` directory primarily intended for preamble code, `packages`, `.bib` files, etc., the files in which can be referenced in various ways.

Additionally, a *group* of archives `group` may have an additional `archive` group/`meta-inf`. If this `meta-inf` archive has a `/lib`-subdirectory, they too will be considered by the following.

---

`\libinput` `\libinput` {some/file} searches for a file `some/file[.tex]` 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 `\inputs` all found files in reverse order; e.g. `\libinput{preamble}` in a `.tex`-file in `sTeX/Documentation` will *first* input `../sTeX/meta-inf/lib/preamble.tex` and then `../sTeX/Documentation/lib/preamble.tex`.

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

`\libinput[some/archive]{some/file}` will do the same, but starting in the `lib` directory of the [math archive](#) `some/archive`.

---

`\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.sty` is found.

---

`\addmhbibresource` `\addmhbibresource` [some/archive]{some/file} searches for a file like `some/file.bib` in `some/archive`'s `lib` directory and calls `\addbibresource` to the result.

---

`\libusetikzlibrary` `\libusetikzlibrary` behaves like `\libusepackage` but looks specifically for `tikz` libraries and calls `\usetikzlibrary` on the results.

throws an error if not *exactly one* candidate for for the library is found.

A good practice is to have individual [S<sub>T</sub>E<sub>X</sub>](#) fragments follow basically this document frame:

```
\documentclass{stex}
\libinput{preamble}
\setsectionlevel{<your preference>}
\begin{document}
  \IfInputref{}{
    ...
    \maketitle
    \ifstexhtml \else \tableofcontents \fi
  }
  ...
  \IfInputref{}{\libinput{postamble}}
\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 a `postamble.tex` can e.g. print the bibliography, index etc.

`\libusepackage` is particularly useful in such a `preamble.tex` when we want to use custom packages that are not part of a [TeX](#) distribution, or on CTAN. In this case we commit the respective packages in one of the `lib` folders and use `\libusepackage` to load them.

## 5.4 Basic Macros

---

`\sTeX`  
`\stex` The `\sTeX` macro produces this  $\text{\sTeX}$  logo. It is provided by the `stex-logo` [package](#), included with the `stex` [package](#).

---

`\ifstexhtml` The [TeX](#) conditional `\ifstexhtml` is *true* if the current compilation generates [HTML](#), and *false* otherwise (i.e. generates [PDF](#)).

---

`\STEXinvisible` `\STEXinvisible{<code>}`  
Processes `<code>`, but does not generate any output. In the [HTML](#), `<code>` is exported with `display:none`.  
Can be used to declare formal content and preserve its semantics in [HTML](#) without generating output.

# Chapter 6

## Document Features

### 6.1 Document Fragments

`sfragment` (*env.*) To make reusability of document fragments more feasible, `TeX` provides the `sfragment` environment. `\begin{sfragment}[id=<id>,short=<short title>]{section title}` calls `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph` or `\subparagraph` with argument `{section title}` depending on the current *section level* and availability, and increases the level accordingly.

The `<id>` can be used for cross-document references (see [section 6.3](#)).

`blindfragment` (*env.*) In the case where we want to increase the section level *without* producing a corresponding section header, the `blindfragment` environment can be used. This allows e.g. typesetting `\sections` before the first `\chapter`.

---

`\skipfragment` The `\skipfragment` macro “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`.

---

`\setsectionlevel` The `\setsectionlevel` macro sets the current section level to that provided as argument. This is particularly useful in the preamble of a document, as to be ignored in e.g. `\inputref` and make sure that sectioning proceeds as desired; e.g. `\setsectionlevel{section}` make sure that the first `sfragment` will be typeset as a `\section` (rather than e.g. a `\part`).

---

`\currentsectionlevel`  
`\Currentsectionlevel` The `\currentsectionlevel` macro produces the literal string corresponding to the current section level – e.g. within a chapter (but outside of a section), `\currentsectionlevel` produces “chapter”.

The `\Currentsectionlevel` macro does the same, but capitalizes the first letter; e.g. in the above situation, `\Currentsectionlevel` produces “Chapter”.

## 6.2 Using and Referencing Document Fragments

---

`\inputref` `\inputref` [`<archive>`]{`<file>`}

Inputs the file `<file>` in `<archive>`'s source directory. If [`<archive>`] is empty, the current archive's source directory is used. If there is no current archive, `<file>` is resolved relative to the current file.

The file's content is processed within a `TeX` group when using `pdflatex`. When converting to `HTML` however, the file is not processed *at all*, and instead, a reference to the file is inserted, that can be replaced by the `HTML` generated by the referenced file by e.g. the `MMT` system.

This is the recommended method to assemble documents from individual `.tex` files.

---

`\mhinput` Like `\inputref`, but actually calls `\input` in both `PDF` and `HTML` mode. Useful for small fragments or those without `modules`, but generally `\inputref` should be preferred.

---

`\ifinputref` `\ifinputref` is a `TeX` conditional for whether the current file is currently processed via `\IfInputref` `\inputref`.

`\IfInputref` `{<true code>}{<false code>}` behaves like

`\ifinputref``<true code>\else``<false code>\fi` when using `pdflatex`; in `HTML` mode however, *both* arguments are processed and marked-up accordingly, so a hosting server (like `MMT`) can dynamically decide which parts to show or omit.

---

`\mhgraphics` `\mhgraphics` If the `graphicx` package is loaded, `\mhgraphics` takes the same arguments as `\includegraphics`, with the additional optional key `archive`. It then resolves the file path in `\mhgraphics``[archive=some/archive]{some/image}` relative to the source-folder of the `some/archive` archive. If no `archive` is provided, the file path `some/image` is resolved relative to the current archive (if existent).

`\cmhgraphics` additional wraps the image in a `center`-environment.

---

`\lstinputmhlising` `\clstinputmhlising` Like `\mhgraphics`, but for `\lstinputlisting` instead of `\includegraphics`. Only defined if the `listings` package is loaded.

## 6.3 Cross-Document References

If we take features like `\inputref` and `\mhinput` (and the `sfragment` environment) 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 `\inputrefed` 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.

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 if the reference target occurs in the *same* document, but if not, we need to know where to find the label,
- (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=<file1>]
<{label}>[archive=<archive2>,file=<file2>,title=<title>]
```

This macro references `<label>` (declared in `<file1>` in `math archive <archive1>`). If the object (section, figure, etc.) with that label occurs (eventually) 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 `<file2>` in `archive <archive2>`, followed by the title.

In `HTML` mode, the reference additionally links to the `HTML` of the `file1`.<sup>11</sup>

This works by storing labels during compilation in a file `<jobname>.sref`, analogous to e.g. the `.toc`. Note that this consequently requires both `file1.tex` and `file2.tex` to have been compiled previously, to generate the `.sref` file.

For example, doing

```
\sref[file=tutorial/full.en]{sec:basics}[file=tutorial.en,title=the \stex Tutorial]
```

in this very document fragment (`[sTeX/Documentation]macros/sref.en.tex`) will yield [Part I \(The Basics\) in the sTeX Tutorial](#) if compiled itself, or if compiled as part of the `sTeX` manual, and will yield the `\autoref` link [chapter 2](#) in the documentation (which includes the tutorial).

---

`\srefsetin`

```
\srefsetin [(archive2)]{<file2>}{<title>}
```

Sets a default value for the optional arguments `<archive2>`, `<file2>` and `<title>` of `\sref`. If the second set of optional arguments in `\sref` are omitted, these default values are used. Particularly useful to set in a preamble.

---

`\sreflabel`

```
\sreflabel {<label>} sets a label analogous to \label{<label>}, but for use in \sref.
```

Note that for every `sTeX macro` or `environment` that takes an optional `id=<id>` argument, the `<id>` (if non-empty) generates an `\sreflabel` automatically.

For example, `\begin{sfragment}[id=foo]{Foo}` is equivalent to `\begin{sfragment}{Foo}\sreflabel{foo}`.

---

`\extref`

```
\extref [archive=<archive1>,file=<file1>]  
{<label>}{archive=<archive2>,file=<file2>,title=<title>}
```

Like `\sref`, but with the third argument mandatory, `\extref` will *always* produce the output as if `<label>` would *not* occur in the current document.

# Chapter 7

## Modules and Symbols

### 7.1 Modules

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

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

`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.

`style` (*<string>\**) for use in customizations, see [chapter 9](#)

`id` (*<string>*) for cross-referencing, see `\sreflabel`.

`ns` (*<URI>*) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed from the containing file and `archive`'s namespace.

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

`sig` (*<language>*) see below.

---

`\STEXexport` `\STEXexport`{*<code>*} executes *<code>* immediately and every time the current `module` is being used.



For technical reasons, `\STEXexport` processes its content in the `expl3` category code scheme – what this means is that all spaces are ignored entirely, and the characters `_` and `:` are valid characters in `macro` names.

In practice, this means you will have to use the `~` character for spaces, and if you want to use a subscript `_`, you should use the `macro` `\c_math_subscript_token` instead.

Also, note that no *global* `macro` definitions should happen in `\STEXexport`; this can lead to unexpected behaviour if the containing `module` has been used previously in the current document.

### 7.1.1 Signature `Modules`, Languages, and Multilinguality

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.

For example, we can have a file `Foo.en.tex`, that declares and documents a `module` `Foo` (using `\begin{smodule}{Foo}`). If we put a file `Foo.de.tex` next to it, we can do `\begin{smodule}[sig=en]{Foo}` to have all the content in the `module` `Foo` (as declared in `Foo.en.tex`) available and translate its document content to german.

The `MMT` backend, when serving `STEX` content as `HTML`, will always attempt to find documentation in the language corresponding to the context; e.g. a user's preference.

## 7.2 Symbol Declarations

---

`\symdecl`

```
\symdecl {<mname>}[<options>]
```

The `\symdecl` `macro` is the simplest way to introduce a new `symbol`. If `<options>` contains `name=<name>`, then `<name>` is the `name` of the `symbol`; otherwise, `<mname>` is used for the `name`. Additionally, a `semantic macro` `\mname` is generated.

The starred variant `\symdecl*` does not generate a `semantic macro`, in which case the `name`-option is superfluous.

```
←M→ \symdecl introduces a new MMT/OMDOC constant in the current module (i.e.
→M→ MMT/OMDOC theory). Correspondingly, they get assigned the MMT-URI
~T~> <module-URI>?<constant-name>.
```

`\symdecl` takes the following optional arguments:

`name` see above,

`args` the arity of the `symbol` and its `semantic macro`; may be a number 0...9 or a string consisting of the characters `i`, `a`, `b` and `B` of length  $\leq 9$ ,

`type` the `symbol`'s `type`,

`def` the `symbol`'s `definiens`,

`return` the `symbol`'s *return code* (see below), most commonly the `semantic macro` of a `mathematical structure`,  
`assoc` how to resolve arguments of `mode` `a` or `B`; may be `pre`, `bin`, `binl`, `binr` or `conj`,  
`reorder` how to reorder the arguments in `OMDoc` (*advanced*),  
`role` `symbols` with certain roles are treated in particular ways in `MMT/OMDoc` (*advanced*),  
`argtypes` **TODO**<sup>12</sup>.

---

`\textsymdecl` `\textsymdecl{⟨mname⟩}[⟨options⟩]{⟨code⟩}`

Like `\symdecl`, but requires that the `symbol` has arity 0 (hence `\textsymdecl` does not take the `args`-option), and generates a `semantic macro` that takes no arguments in either text or math mode, and produces marked-up `⟨code⟩` as output.

Additionally, a `macro` `\⟨mname⟩name` is generated that produces `⟨code⟩` without any semantic markup.

---

`\symdef` `\symdef{⟨mname⟩}[⟨options⟩]{⟨notation⟩}`

Combines the functionalities and optional arguments of `\symdecl` and `\notation` in one.

### 7.2.1 Returns

Assume we have a `symbol` `foo` with `semantic macro` `\foo`, (exemplary) taking two arguments, and `return=⟨code⟩`. If we do `\foo{a}{b}!`, the return code is simply ignored. If we do `\foo{a}{b}` *without* the `!`, here is what happens:

1. `\TeX` will replace `#1` and `#2` in `⟨code⟩` by `a` and `b`, yielding `⟨retcode⟩`.
2. `\TeX` will insert `⟨retcode⟩{\foo{a}{b}!}` in the input token stream.

This means that `⟨code⟩` should contain at most `⟨arity of foo⟩` argument markers, and eat precisely one argument appended to `⟨code⟩`.

When in doubt, we recommend only using `semantic macros` for `mathematical structures` (with only optional arguments) and `\apply` (with only optional arguments) in `return`.

## 7.3 Referencing Symbols

---

`\symref`  
`\sr` `\symref{⟨symbol⟩}{⟨text⟩}`

The `\symref` `macro` (and its short version `\sr`) is the most general variant to mark-up arbitrary `LATEX` code `⟨text⟩` with the `symbol`.

---

<sup>12</sup>**TODO: experimental**





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

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

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

If `symbol` does *not* correspond to a `semantic macro` `\symbol` and does *not* contain a `?`, then  $\LaTeX$  checks all `symbols` currently in scope until it finds one, whose name is `symbol`. If `symbol` is of the form `pre?name`,  $\LaTeX$  first looks through all `modules` currently in scope, whose full `MMT-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 `\symname{Integers?addition}` or `\symname{RealNumbers?addition}` in the case where several `additions` are in scope.

$\LaTeX$  `\symref{<symbol>}{<text>}` in `MMT/OMDOC` generates the term `<OMS name="{<symbol URI>}" />`.

---

`\symname` `\symname` [`pre=<pre>`, `post=<post>`] {<symbol>}  
`\sn`  
`\Symname` If the `symbol` referenced by <symbol> has name `name`, this is a shortcut for  
`\Sn` `\symref{<symbol>}{<pre>name<post>}`.  
`\sns` For example, given a `symbol` `agroup` with name `abelian group`, we can do  
`\Sns` `\symname` [`pre=Non-`, `post=s`] {`agroup`} to produce `Non-abelian groups`.  


---

`\sn` is a shorter variant for `\symname`; `\Symname` and `\Sn` additionally capitalize the first letter. `\sns` and `\Sns` are short for `\sn` [`post=s`] and `\Sn` [`post=s`], respectively.

---

`\srefsym` `\srefsym` {<symbol>} {<text>}  
`\srefsymuri` turns <text> into a link to

- The documentation of <symbol>, if it occurs in the same document, or
- the `symbol`'s documentation online, based on the containing `math archive`'s `url-base`.

`\srefsymuri` does the same, but expects a `symbol`'s full `MMT-URI` as first argument. This is particularly useful for e.g. customizing highlighting (see [chapter 9](#)).

---

`\symuse` `\symuse` {<symbol>} behaves exactly like a `semantic macro` for <symbol>.

## 7.4 Notations and Semantic Macros

---

`\notation` `\notation{<symbol>}[<options>]{<code>}`

introduces a new `notation` for the referenced `symbol`.

The starred variant `\notation*` sets this `notation` as the (new) default `notation`.  
The optional arguments are:

- `prec=<opprec>;<argprec 1>x...x<argprec n>`: An `operator precedence` and one `argument precedence` for each argument of the `semantic macro`. If no `argument precedences` are given, all `argument precedences` are equal to the `operator precedence`. By default, all `precedences` are 0, unless the `symbol` takes no argument, in which case the `operator precedence` is `\neginfprec` (negative infinity).

`prec=nobrackets` is an abbreviation for `prec=\neginfprec;\infprec x...x\infprec`.

- `op=<code>`: An `operator notation`. If none is given, the notation component marked with `\maincomp` is used. If no `\maincomp` occurs in the `notation`, the default `operator notation` is `\symname{<symbol>}`.
- `variant=<id>`: An id for this `notation`. The key `variant=` can be omitted; i.e. `\notation[foo]` is equivalent to `\notation[variant=foo]`.

---

`\comp` `\maincomp` `\comp` is used to mark notation components in a `\notation` to be highlighted. Additionally, each `notation` can use `\maincomp` at most once to mark the *primary* notation component.



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 `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 macros` may ever occur inside a `notation`.

Both criteria are not just required for technical reasons, but conceptually 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` application `\plus{a}{b}` 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 `\plus`.

Similarly, a `semantic macro` can not conceptually be part of the `notation` of `\plus`,

since a **symbol** represents a *distinct (mathematical) concept with its own semantics*, and **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 **TeX** users, like us), and might want to give those another thought - quite likely, the concept you aim to implement does not actually represent a semantically meaningful (mathematical) concept, and you will want to use `\def` and similar native **TeX** macro definitions rather than **semantic macros**.

---

`\setnotation` The first **notation** provided will stay the default **notation** unless explicitly changed:  
`\setnotation{\langle symbol \rangle}{\langle id \rangle}` sets the default **notation** of  $\langle symbol \rangle$  to that with id  $\langle id \rangle$ .

### 7.4.1 Precedences and Bracketing

---

`\infprec` `\neginfprec` and `\neginfprec` represent *infinitely large* and *infinitely small* **precedences**, respectively.



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

#### Example 23

Consider **semantic macros** `\plus` and `\mult` taking two arguments, with **notations**  $a + b$  and  $a \cdot b$  respectively, and **precedences** 100 for `\plus` and 50 for `\mult`.

Consider `\plus{a, \mult{b, \plus{c, d}}}` (i.e.  $a + b \cdot (c + d)$ ). Then:

1. **TeX** starts out with  $p_d = \text{\infprec}$ .
2. **TeX** encounters `\plus` with  $p_{op} = 100$ . Since  $100 \not> \text{\infprec}$ , it inserts no parentheses.
3. Next, **TeX** encounters the two arguments for `\plus`. Both have no specifically provided **argument precedence**, so **TeX** uses  $p_d = p_{op} = 100$  for both and recurses.
4. Next, **TeX** encounters `\mult{b, ...}`, whose **notation** has  $p_{op} = 50$ .
5. We compare to the current downward **precedence**  $p_d$  set by `\plus`, arriving at  $p_{op} = 50 \not> 100 = p_d$ , so **TeX** again inserts no parentheses.

6. Since the notation of `\mult` has no explicitly set argument precedences, `\TeX` again uses the operator precedence for the arguments of `\mult`, hence sets  $p_d = p_{op} = 50$  and recurses.
7. Next, `\TeX` encounters the inner `\plus{c, \dots}` whose notation has  $p_{op} = 100$ . We compare to the current downward precedence  $p_d$  set by `\mult`, arriving at  $p_{op} = 100 > 50 = p_d$  – which finally prompts `\TeX` to insert parentheses, and we proceed as before.

---

`\dobracket` `\dobrackets{⟨code⟩}` wraps parentheses around `{⟨code⟩}`.

---

`\withbrackets` `\withbrackets{⟨left⟩}{⟨right⟩}{⟨code⟩}` uses the opening and closing parentheses `⟨left⟩` and `⟨right⟩` for the next pair of parentheses automatically inserted in `{⟨code⟩}`.

### 7.4.2 Notations for Argument Sequences

The following macros can be used in notations that take mode a or B arguments:

---

`\argsep` `\argsep{⟨parameter token⟩}{⟨separator⟩}`

takes the elements of the argument sequence in position `⟨parameter token⟩` and separates them by `⟨separator⟩`.

Note that the first argument *must* be a parameter token of the form `#k`, and the argument at position `k` of the notation has to have argument mode a or B.

---

`\argmap` `\argmap{⟨parameter token⟩}{⟨code⟩}{⟨separator⟩}`

takes the elements of the argument sequence in position `⟨parameter token⟩`, applies the code `{⟨code⟩}` to each of them (which therefore should use `##1`) and separates them by `⟨separator⟩`.

For example, the notation `{\argmap{#1}{X^{##1}}{++}}` applied to the argument `{a,b,c}` produces  $X^a ++ X^b ++ X^c$ .

---

`\argarraymap` **TODO**<sup>13</sup>

### 7.4.3 Semantic Macros

Assume we have a semantic macro `\smacro` taking (exemplary) two arguments. The precise behaviour of `\smacro` depends on whether we are in text or math mode.

**Math Mode** `\smacro!` produces the default operator notation of its symbol. Without `!`, `\smacro` expects at least two arguments, and `\smacro{a}{b}!` produces the default notation of its symbol.

If the symbol has a return code, then `\smacro{a}{b}` continues with executing the return code. Otherwise, `\smacro{a}{b}` also simply produces the default operator notation.

The starred variants `\smacro*` and `\smacro!*` behave as in *text mode*.

**Text Mode** `\smacro!\{<arg>` marks up `<arg>` similarly to how `\symref{smacro}{<arg>` would.

Without the `!`, `\smacro` still only takes a single argument, but it is expected, that within `<arg>`, the arguments for the `symbol` are explicitly marked up. The `\comp macro` is allowed in `<arg>` to determine the components of `<arg>` to be highlighted.

---

`\arg` The `\arg macro` can be used to explicitly mark the arguments of a `semantic macro` in text mode.

By default, they are numbered consecutively; e.g. `\smacro{... \arg{a}... \arg{b}}` determines `a` and `b` to be the (first and second) arguments.

The starred variant `\arg*` allows for marking up the arguments, but does not produce any output. This can be used to provide arguments that are not mentioned in the text we want to mark up, because they are implicitly obvious or mentioned elsewhere.

If we want to change the order of the arguments, we can provide the precise argument number as an optional argument; e.g. `\smacro{... \arg[2]{a}... \arg[1]{b}}` determines `b` to be the first and `a` to be the second argument.

An argument number may be used repeatedly, if the corresponding `argument mode` is `a` or `B`.

$\hookrightarrow$  Applications of `semantic macros` with arguments are translated to `MMT/OMDoc` as OMA-terms with head `<OMS name="<symbol>" />`, or `<OMBIND name="<symbol>" />`, depending on the absence or presence of `argument mode b` or `B` arguments.  
 $\rightarrow$  Semantic macros with no arguments or invoked with `!` correspond to OMS directly.

## 7.5 Simple Inheritance

There are three `macros` that allow for opening a `module`, making its contents available for use:

---

`\usemodule` `\usemodule{<module>` is allowed anywhere and makes the `module`'s contents available up to the current `TeX` group. This is the right `macro` to use outside of `modules`, or when none of its contents use any of the used `module`'s `symbols` directly (e.g. in `types` or `definientia`).

---

`\requiremodule` `\requiremodule{<module>` is only allowed in `modules` and makes the required `module`'s contents available within the current `module`. The imported `symbols` can be safely used in `types` and `definientia`, but not in the `return` code of `symbols`, and the imported content is not exported further – i.e. using the current `module` does not also open the required `module`.

---

`\importmodule` `\importmodule{<module>` is only allowed in `modules` and makes the required `module`'s contents available within the current `module`. The imported `symbols` can be safely used anywhere, and the imported content exported to any `modules` subsequently importing the current one.

$\hookrightarrow$  In `MMT`, every *document* and every `module` induces an `MMT theory`. `\usemodule`  
 $\rightarrow$  induces and `MMT include` in the document `theory`, `\importmodule` and  
 $\rightsquigarrow$  `\requiremodule` both induce an `include` in the `module's theory`.

It is worth going into some detail how exactly `\usemodule`, `\importmodule` and `\requiremodule` 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 `MMT-URI`.

Ideally, `STEX` would allow for arbitrary `MMT-URIs` for `modules`, with no forced relationships between the *logical* namespace of a `module` and the *physical* location of the file declaring it – like `MMT` in fact allows for.

Unfortunately, `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 `STEX` 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 `math 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 `math 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` directory 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 file 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 file, 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` directory.
- 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` directory, 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.

## 7.6 Variables and Sequences

---

`\vardef`

`\vardef{⟨mname⟩}[⟨options⟩]{⟨notation⟩}`

Takes the same arguments as `\symdef`, but produces a `variable` rather than a `symbol`. `Variables` definitions are always local to the current `TEX` group and are allowed anywhere (i.e. outside of `modules`).

`⟨options⟩` may include the additional keyword `bind`, in which case the `variable` will be appropriately abstracted away in statements (see also `\varbind`).

Unlike `\symdef`, there is no starred variant `\vardef*` – `variables` always generate a `semantic macro`.

The `semantic macro` for a `variable` behaves analogously to that of a `symbol`.

`Variables` induces the same `MMT/OMDOC` terms as `symbols` do, except for the head of the term being `<OMV name="...">/>` instead of `<OMS/>`.

---

`\varnotation`

`\varnotation{⟨variable⟩}[⟨options⟩]{⟨notation⟩}`

Takes the exact same arguments as `\notation`, but attaches an additional `notation` to the `variable` `⟨variable⟩` rather than a `symbol`.

---

`\svar`

`\svar[⟨name⟩]{⟨text⟩}`

Semantically marks up `⟨text⟩` as representing a `variable` `⟨name⟩`. The `variable` does not need to have been defined prior. If no `⟨name⟩` is given `⟨text⟩` will be used as the name.

This is useful in situations like “throwaway expressions” or remarks; e.g.

`⋄\plus{\svar{n},\svar{m}}⋄` means...

---

`\varseq`

`\varseq{⟨mname⟩}[⟨options⟩]{⟨range⟩}{⟨notation⟩}`

Declares a new `variable` sequence. The `⟨options⟩` are the same as for `\vardef`. If not provided, `args=1` by default (a 0-ary sequence would just be a normal `variable`).

A `type` (given as `type=`) is interpreted to be the `type` of every element  $a_i$  of the sequence  $a_1, \dots, a_n$  (not of the sequence itself). If the `type` is itself a sequence  $A_1, \dots, A_n$ , the assumption is that its range is the same as the one of the new sequence, and the type of every  $a_i$  in the sequence is  $A_i$ .

`⟨range⟩` needs to be a comma-separated sequence of either `args` many arguments, or `\ellipses`.

The resulting `semantic macro` is allowed anywhere `STEX` expects an `argument mode` a or B argument.

---

`\ellipses` Represents ellipses in a range; produces `\ellipses` in math mode.

---

`\seqmap` `\seqmap{<code>}{<sequence>}`

Maps the function `<code>` (containing `#1`) over every element of the `<sequence>`.  
Is allowed anywhere `STEX` expects an `argument mode a` or `B` argument.

## 7.7 Structures

Mathematical structure bundle interdependent symbols together.

`mathstructure` (*env.*) `\begin{mathstructure}{<mname>}[<name>,this=<code>]` opens a new mathematical structure with name `<mname>` (if provided) or `<mname>` (otherwise), and semantic macro `\mname`. It subsequently behaves like the `smodule` environment.

---

`\this` The optional `this=<code>` option allows for setting the typesetting of the `\this` macro within a `mathstructure`. In particular, `\this` can be used in notations for symbols declared in the structure. `\this` can be thought of as representing “the” (current) instance of this structure.

`extstructure` (*env.*) `\begin{extstructure}{<mname>}[<name>,this=<code>]{<structs>}` opens a new mathematical structure extending the structures given in `<structs>` (a comma-separated list of names).

`extstructure*` (*env.*) `\begin{extstructure*}{<struct>}` opens a new mathematical structure conservatively extending the (single) structure `<struct>`. *Conservative* meaning: Every symbol newly introduced in this structure needs to have a *definiens*. The new symbols are attached as fields directly to `<struct>`.

---

`\usestructure` The `\usestructure` macro behaves like `\usemodule` for mathematical structures, making the symbols available to use directly.

`mathstructure` make use of the *Theories-as-Types* paradigm (see [MueRabKoh:tat18]):

$\hookrightarrow$  `\begin{mathstructure}{<name>}` creates a nested theory with name `<name>-module`. The constant `<name>` is defined as a *dependent record type with manifest fields*, the fields of which are generated from (and correspond to) the constants in `<name>-module`.

### 7.7.1 Semantic Macros for Structures

Assume we have a mathematical structure with semantic macro `\struct`:

#### Example 24

```
\begin{mathstructure}{struct}
  \symdef{fieldda}{a}
  \symdef{fielddb}[args=2]{#1 \maincomp{b} #2}
  \symdef{fielddc}[args=2,def=\sn{fielddb}]{#1 \maincomp{c} #2}
```



```

\inliness[name=axiom1]{\conclusion{some axiom}}
\end{mathstructure}
\notation{struct}{StRuCt}

```

- If `\struct` has no `notations`, then  $\$ \backslash struct ! \$$  produces  $\langle a, b, c \rangle$ . Otherwise, it produces the notation, i.e.  $StRuCt$ . In both cases,  $\$ \backslash struct \{ \} \{ \} \$$  produces  $\langle a, b, c \rangle$  (for reasons that will become clearer in a moment).
- $\$ \backslash struct \{ \} \{ \} \$$  (or  $\$ \backslash struct ! \$$  in the case where no `notations` are around) can be modified in the following ways:
  - $\$ \backslash struct \{ \} \{ \} [ \langle fieldname \rangle , \dots ] \$$  lets you pick, which precise fields to show, so e.g.  $\$ \backslash struct \{ \} \{ \} [ fielda , fieldb ] \$$  produces  $\langle a, b \rangle$ . By default,  $\$ \backslash struct \{ \} \{ \} \$$  shows exactly the fields that have `semantic macros` (which are also used to access the fields in  $\$ \backslash struct \{ \dots \} \{ fieldname \}$ ).
  - $\$ \backslash struct \{ \} \{ \} [ \langle id \rangle ] \$$  lets you pick the `notation` of the “mathematical structure” symbol to use to typeset the `structure`; e.g.  $\$ \backslash struct \{ \} \{ \} [ parens ]$  yields  $\langle a, b, c \rangle$ , and (combining both)  $\$ \backslash struct \{ \} \{ \} [ fielda , fieldb ] [ angle ]$  yields  $\langle a, b \rangle$ .
- The two arguments in  $\$ \backslash struct \{ first \} \{ second \} \$$  represent 1. the term that is to be treated as an instance of `\struct`, and 2. the precise field to invoke. If the first is empty, then there is no instance. If the second is empty, `TEX` will present all of them (that are not assertions). Hence,  $\$ \backslash struct \{ S \} \{ \} \$$  yields  $\langle a_S, b_S, c_S \rangle$ ,  $\$ \backslash struct \{ S \} \{ fielda \} \$$  produces  $a_S$ ,  $\$ \backslash struct \{ \} \{ fielda \} \$$  produces  $a$ , and  $\$ \backslash struct \{ S \} \{ fieldb \} \{ x \} \{ y \} \$$  produces  $xb_Sy$ .
- For the sake of completion,  $\$ \backslash struct \{ first \} ! \$$  simply produces the given argument; e.g.  $\$ \backslash struct \{ S \} ! \$$  simply produces  $S$ .

More precisely:  $\$ \backslash struct \{ \langle code \rangle \} \$$  acts like a “type coercion” of  $\langle code \rangle$  to be an instance of `\struct`.

Of course, it is (occasionally, but) rarely useful to use the `semantic macro` `\struct` by giving it two arguments *manually*; but this is what `TEX` does when using `\struct` in the return of a `symbol` (or `variable`).

Continuing:

- $\$ \backslash struct [ comp = \langle code \rangle ] \{ \dots \} \{ \dots \} \$$  applies  $\langle code \rangle$  (using #1) to every occurrence of `\maincomp` in the `notations` of the fields, as a replacement for the default modification  $\{ \#1 \} \_ \{ \backslash this \}$ . For example,  $\$ \backslash struct [ comp = \{ \#1 \} \wedge \{ Foo \} ] \{ S \} \{ \} \$$  produces  $\langle a^{Foo}, b^{Foo}, c^{Foo} \rangle$ , and  $\$ \backslash struct [ comp = \{ \#1 \} \wedge \{ \backslash this \} ] \{ S \} \{ fieldb \} \{ x \} \{ y \} \$$  yields  $xb^Sy$ .
- $\$ \backslash struct [ this = \langle code \rangle ] \{ \dots \} \{ \dots \} \$$  modifies the way `\this` is being typeset; i.e. the presentation of the first  $\{ \dots \}$  argument. For example  $\$ \backslash struct [ this = T ] \{ S \} \{ \} \$$  produces  $\langle a, b, c \rangle$  – since `\this` is not being used in the `notations` of the fields. Note that the `this=` and `comp=` variants are (as of yet) mutually incompatible.
- Finally,  $\$ \backslash struct [ \langle fieldname \rangle = \langle code \rangle ] \{ \dots \} \{ \dots \} \$$  assigns the field  $\langle fieldname \rangle$  to the term  $\langle code \rangle$ .  $\langle code \rangle$  is subsequently used when using  $\{ \dots \} \{ \}$ , but not in fields directly. For example,  $\$ \backslash struct [ fielda = A ] \{ S \} \{ \} \$$  produces  $\langle A!, b_S, c_S \rangle$ , but  $\$ \backslash struct [ fielda = A ] \{ S \} \{ fielda \} \$$  produces  $a_S$ .

Note the insertion of ! behind the A – this is to make sure that assignments to [semantic macros](#) that takes arguments don't accidentally eat more than they should.

Also note that multiple assignments can be done in the same pair of [], or chained – i.e. both `\struct[fielda=A,fieldb=B]...` and `\struct[fielda=A][fieldb=B]...` are valid and equivalent. `\struct[fielda=A,fielda=B]...` however is not – every field may be assigned at most once.

## Chapter 8

# Statements

**STEX** provides four environments to semantically annotate various kinds of statements:

**sdefinition** (*env.*) The **sdefinition** environment represents (primarily mathematical) *definitions*; in particular for **symbols**. The contents of the environment will be used as *documentation* for any **symbol** that either occurs as a `\definiendum` (or `\definame`) within the **sdefinition**, or that is listed in the optional `for=` argument of the environment.

If a `\definiens` occurs, this will be used by **MMT** as the formal definiens for the respective **symbol**.

**sassertion** (*env.*) The **sassertion** environment represents *assertions*, i.e. **propositions** such as *theorems*, *lemmata*, *axioms*, etc. If a `\conclusion` occurs within the **sassertion**, its argument will be used as the formal *statement* of the assertion.

**sexample** (*env.*) The **sexample** environment represents examples, the `for=` attribute specifies the concept this is an example for. For counterexamples use `style=counterexample`

**sparagraph** (*env.*) The **sparagraph** environment represents all other kinds of (logical) paragraphs, such as remarks, comments, transitions between topics, recaps, reminders, etc.

All of these take the same arguments:

- `for=<csl>`: a comma-separated list of **symbols**.
- The same optional arguments as `\symdecl`, with `macro=` replacing the name of the **semantic macro**. All of them are only relevant, if either `name=` or `macro=` are provided.

As with `\symdecl`, if no `name` is given, but `macro` is, then the same name is used for both the **semantic macro** and the **symbol** itself.

If `name` is given but `macro` isn't, no **semantic macro** is generated. Subsequently, the newly generated **symbol** is added the `for=`-list.

- `style`: see **chapter 9**.
- `title`: a title to use in various styles (see **chapter 9**).
- `id`: a label to use for `\sref`.

---

`\inlineass` The macros `\inlineass`, `\inlinedef` and `\inlineex` behave like the `sassertion`,  
`\inlinedef` `sdefinition` and `sexample` environments respectively, but take the text to annotate as  
`\inlineex` an argument, rather than as the body of an environment, and do not break paragraphs.  
 The same macros available in the environments are also available in the argument  
 of the `\inline*` macros.

---

`\varbind` `\varbind{<cls>}`  
 retroactively attaches the `bind` option to every variable provided (as a comma-separated  
 list).

## 8.1 More on Definitions

In `sdefinition` (and `sparagraph` with `style=symdoc`), the following additional macros  
 are available:

---

`\definiendum` The `\definiendum` macro behaves largely like `\symref`, but it uses a dedicated highlight-  
`\definame` ing for *definienda* and adds the referenced symbol to the `for=` list of the environment.  
`\Definame` `\definame` is to `\definiendum` as `\symname` is to `\symref`. Analogously, `\Definame`  
`\defnotation` behaves like `\Symname`.  
`\defnotation` can be used in math mode to apply the `\definiendum` highlighting  
 to notations.

---

`\definiens` The `\definiens` macro can be used to semantically annotate the *definiens* in a  
`\definiens` `sdefinition`.

If the `sdefinition` environment has several elements in its `for` list, an optional  
 argument `\definiens[<symbol>]{...}` can be used to tell `STEX` which symbol's definiens  
 this is. By default, the *first* symbol in the `for` list is used.

Here is how `MMT` will treat the fragment marked up with `\definiens`:  
 Firstly, it will attempt to translate its contents into an `MMT/OMDoc` term. This  
 succeeds easily if `\definiens` is some semantic macro (applied to arguments).  
 $\hookrightarrow$  Secondly, it will check for all variables currently in scope, that were defined with  
 $\hookrightarrow$  the optional argument `bind`. It then will check, whether a symbol is in scope, that  
 $\hookrightarrow$  has `role=lambda`. If so, it will use that lambda symbol to bind all these variables  
 (in the order in which they were defined) in the term. If no lambda symbol is  
 found, it will use the bind symbol that ships with `STEX`.  
 The final term will be attached as definiens to the corresponding `MMT` constant,  
 if it was declared in the same module as the `\definiens` occurrence.

## 8.2 More on Assertions

---

`\premise`    The `\conclusion` macro can be used to mark up the actual statement within an `sassertion`. The `\premise` macro can be used to additionally mark up *premises*.  
`\conclusion`

---

If the `sassertion` environment has several elements in its `for` list, an optional argument `\conclusion[⟨symbol⟩]{...}` can be used to tell `TEX` which `symbol`'s statement this is. By default, the *first* `symbol` in the `for` list is used.

Here is how `MMT` will treat the fragments marked up with `\conclusion` and `\premise`:

Firstly, it will attempt to translate the contents of `\conclusion` into an `MMT/OM-DOC` term  $c$ . This succeeds easily if the `\conclusion` is some `semantic macro` (applied to arguments).

Secondly, it will collect all fragments marked up with `\premise` and do the same to them  $(p_1, \dots, p_n)$ .

It will then check, whether a `symbol` is in scope, that has `role=implication`. If so, it will use that `implication symbol` to attach all the premises to the conclusion, resulting in  $t := \text{imply}(p_1, \dots, p_n, c)$ .

Next, it will check for all `variables` currently in scope, that were defined with the optional argument `bind`. It then will check, whether a `symbol` is in scope, that has `role=forall`. If so, it will use that `forall symbol` to bind all these variables (in the order in which they were defined) in the term  $t$ .

Finally, it will check, whether a `symbol` is in scope, that has `role=judgment`. If so, it will set  $t := \text{judgment}(t)$ .

If no `forall symbol` is found, it will first apply the `judgment symbol` (if existent) and then use the `bind symbol` that ships with `TEX` to bind the `variables`.

The final term will be attached as `type` to the corresponding `MMT constant`, if it was declared in the same `module` as the `\definiens` occurrence.

`sproof (env.)`

TODO<sup>14</sup>

---

<sup>14</sup>TODO: proofs

## Chapter 9

# Customizing Typesetting

There are two kinds of typesetting that can be customized in [L<sup>A</sup>T<sub>E</sub>X](#): [symbol](#) references ([notation](#) components, `\symref`, [variables](#), etc.) are highlighted using a small set of [macros](#) that can be simply redefined by authors.

Other [macros](#) and [environments](#) usually have more complicated “typesetting rules” associated with them – often in the form of other already existing [environments](#) that should be used.

Lastly, in [HTML](#) we can provide custom CSS rules in [math archives](#) that determine the styling of certain [environments](#), so that the actual presentation depends on the document in which the fragments are included (e.g. via `\inputref`), rather than the file the fragment is implemented in.

It is generally recommended to implement these customizations in a preamble in the `lib` directory (see [section 5.3](#))

### 9.1 Highlighting Symbol References

---

`\symrefemph`  
`\symrefemph@uri`

`\symrefemph` governs how references via `\symref` (or `\symname`, or their short variants) are highlighted;

Doing `\symref{<symbol>}{<text>}` ultimately expands to `\symrefemph@uri{<text>}{<symbol URI>}`, the default implementation of which is just `\symrefemph{<text>}`. The default implementation of `\symrefemph`, in turn, is just `\emph{<text>}`.

If you only want to change e.g. the color of `\symrefs`, you only need to redefine `\symrefemph`, e.g. using

```
\renewcommand\symrefemph[1]{\textcolor{red}{#1}}
```

the `@uri` variant is useful if you want to link somewhere, or show the URI in a tooltip. The `stex-highlighting` [package](#) does both, using:

```
\usepackage{pdfcomment}
\protected\def\symrefemph@uri#1#2{%
  \pdftooltip{%
    \srefsymuri{#2}{\symrefemph{#1}}%
  }{%
    URI:~\detokenize{#2}%
  }%
}
```

```

}
\def\symrefemph#1{%
\ifcsname textcolor\endcsname
\textcolor{teal}{#1}%
\else#1\fi
}

```

---

`\compemph`    Like `\symrefemph` and `\symrefemph@uri`, but governs the highlighting of components (marked with `\comp` or `\maincomp`) in *notations*.

---



---

`\defemph`    Like `\symrefemph` and `\symrefemph@uri`, but governs the highlighting of definienda marked with `\definiendum` (or `\definame`).

---



---

`\varemph`    Like `\compemph` and `\compemph@uri`, but governs the highlighting of components (marked with `\comp` or `\maincomp`) in the *notations* of *variables*.  
`\varemph@uri`    The second argument to `\varemph@uri` is the *name* of the *variable*.

---

## 9.2 Styling Environments and Macros

A variety of *environments* and *macros* provided by `STeX` are *stylable* using the `macros` `\stexstyle⟨name⟩[⟨style⟩]{...}`. These stylable *environments* and *macros* bind various of their parameters to `macros` `\this⟨parameter⟩`, which can then be used in the styles.

For example, if we have a *definition environment* that we would want to use to style our *sdefinitions*, we can do (in the simplest case)

```
\stexstyledefinition{\begin{definition}}{\end{definition}}
```

This tells `STeX` to insert `\begin{definition}` at the beginning of every *sdefinition environment*, and `\end{definition}` at the end.

If we have a *environments theorem* and *lemma*, we probably want the *sassertion environment* to use those for theorems and lemma. We can achieve that by doing

```
\stexstyleassertion[theorem]{\begin{theorem}}{\end{theorem}}
\stexstyleassertion[lemma]{\begin{lemma}}{\end{lemma}}
```

Now if we do `\begin{sassertion}[style=theorem]`, it will wrap the *environment* with `\begin{theorem}... \end{theorem}`.

Of course, many such statements might have a title, as e.g. in

**Theorem 9.2.1** (Gödel's First Incompleteness Theorem). ...

In *sassertion*, we can provide that title as optional argument using `title=...`. Before calling the styling provided, *sassertion* will store that title in the macro `\thistitle`, which we can use in the styling. For example, we might prefer to pass it on to the *theorem environment*:

```
\stexstyleassertion[theorem]{\ifx\thistitle\@empty
\begin{theorem}\else\begin{theorem}[\thistitle]\fi}
{\end{theorem}}
```

---

```

\stexstylemodule
\stexstylecopymodule
\stexstyleinterpretmodule
\stexstylerealization
\stexstylemathstructure
\stexstyleextstructure
\stexstyledefinition
\stexstyleassertion
\stexstyleexample
\stexstyleparagraph
\stexstyleproof
\stexstylesubproof

```

---

TODO<sup>15</sup>

Additionally, we can style certain [macros](#), if we want them to produce output. For example, we might (for debuggin or documentation purposes) `\symdecl` to give a short summary of the symbol.

We can achieve that by doing, for example:

```

\stexstylesymdecl[debug]{
  Symbol \thisdeclname~(with arity \thisargs) of type $\thistype$.
}

```

in which case

```

\symdecl{foo}[args=2,type=\mathbb{N},style=debug]

```

will yield

Symbol foo (with arity ii) of type N.

---

```

\stexstyleusemodule
\stexstyleimportmodule
\stexstylerequiremodule
\stexstyleassign
\stexstylerenamedecl
\stexstyleassignMorphism
\stexstylecopymod
\stexstyleinterpretmod
\stexstylerealize
\stexstylesymdecl
\stexstyletextsymdecl
\stexstylenotation
\stexstylevarnotation
\stexstylesymdef
\stexstylevardef
\stexstylevarseq
\stexstylespfsketch
\stexstyleMMTinclude

```

---

TODO<sup>16</sup>

### 9.3 Custom CSS for Environments



# Chapter 10

## Additional Packages

### 10.1 NotesSlides Manual

#### 10.1.1 Introduction

The `notesslides` document class is derived from `beamer.cls` [`beamerclass:on`], 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.

#### 10.1.2 Package Options

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

---

`slides`  
`notes` The options `slides` and `notes` switch between slides mode and notes mode (see [subsection 10.1.3](#)).

---

`sectocframes` If the option `sectocframes` is given, then for the `sfragments`, special frames with the `sfragment` title (and number) are generated.

---

`frameimages`  
`fiboxed` If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated frames (see ??). If also the `fiboxed` option is given, the slides are surrounded by a box.

### 10.1.3 Notes and Slides

`frame` (*env.*) Slides are represented with the `frame` environment just like in the `beamer` class, see [Tantau:ugbc] 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 `\notestru` or `\notesfalse` is strongly discouraged however.



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



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

`\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

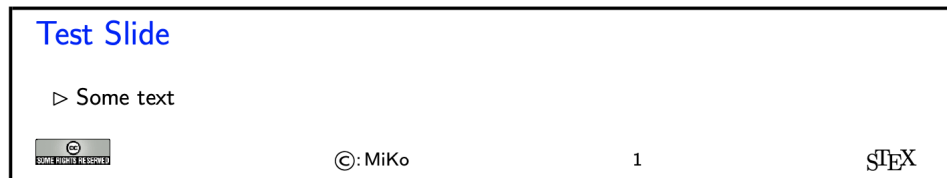
`nexample` (*env.*) level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,

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

`nassertion` (*env.*)

### 10.1.4 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  $\TeX$ -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.

`\setslideologo` The default logo provided by the `notesslides` package is the  $\TeX$  logo it can be customized using `\setslideologo{<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.

### 10.1.5 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  $\TeX$  notes.

---

`\frameimage` In this case we can use `\frameimage[⟨opt⟩]{⟨path⟩}`, where `⟨opt⟩` are the options of  
`\mhframeimage` `\includegraphics` from the `graphicx` package [CarRah:tpp99] 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 doc-  
 ument 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: 

## 10.1.6 Ending Documents Prematurely

---

`\prematuarestop` For prematurely stopping the formatting of a document, `TEX` provides the `\prematuarestop`  
`\afterprematuarestop` macro. It can be used everywhere in a document and ignores all input after that – back-  
 ing out of the `sfragment` environments as needed. After that – and before the implicit  
`\end{document}` it calls the internal `\afterprematuarestop`, which can be customized  
 to do additional cleanup or e.g. print the bibliography.

`\prematuarestop` 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 `\prematuarestop` 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 [lmhtools:github:on].

Text fragments and modules can be made more re-usable by the use of global vari-  
 ables. 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.

## 10.1.7 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.

---

`\setSGvar` `\setSGvar{<vname>}{<text>}` to set the global variable `<vname>` to `<text>` and `\useSGvar{<vname>}` to reference it.

---

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

### 10.1.8 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.

Local Variables: mode: latex TeX-master: ../stex-manual End:

## 10.2 Problem Manual

### 10.2.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: `hints`, `notes`, and `solutions`. 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.

### 10.2.2 Problems and Solutions

---

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code> (should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?), <code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then the corresponding auxiliary parts of the problems are output, otherwise, they remain invisible.
<code>notes</code>	
<code>hints</code>	
<code>gnotes</code>	
<code>pts</code>	
<code>min</code>	
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so that they are more visible in the text. Finally, the <code>test</code> option signifies that we are in a test situation, so this option does not show the solutions (of course), but leaves space for the students to solve them.
<code>test</code>	

---

`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 25

Input:

```

1 \documentclass{article}
2 \usepackage[solutions,hints,pts,min]{problem}
3 \begin{document}
4   \begin{sproblem}[id=elefants,pts=10,min=2,title=Fitting Elefants]
5     How many Elefants 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}
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:

**Exercise (Fitting Elefants)**  
 How many Elefants can you fit into a Volkswagen beetle?

---

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

---

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

`\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.

### 10.2.3 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` [*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 26

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:



### Exercise (Functions)

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

- def<sup>a</sup>
- function<sup>b</sup>
- fun<sup>c</sup>
- public static void<sup>d</sup>

<sup>a</sup>Korrekt

<sup>b</sup>Falsch

*that is for C and C++*

<sup>c</sup>Falsch

*that is for Standard ML*

<sup>d</sup>Noooooooooo

*that is for Java*

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

### Example 27

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:

### Exercise (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 a single argument, which contains a concrete solution (i.e. a number, a string, ...), which generates a fill-in-box in test mode:

### Example 28

Input:

```
1 \stopsolutions
2 \begin{sproblem}[id=elefants.fillin,title=Fitting Elefants]
Output: How many Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
4 \end{sproblem}
```

```
Exercise (Fitting Elefants)
and the actual solution in solutions mode:
Example 29: How many Elefants can you fit into a Volkswagen beetle? 
```

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

```
Output:
Exercise (Fitting Elefants)
How many Elefants 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>17</sup>

## 10.2.4 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>17</sup>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.

## 10.2.5 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.

<code>\testspace</code>	<code>\testspace</code> takes an argument that expands to a dimension, and leaves vertical
<code>\testsmallspace</code>	space accordingly. Specific instances exist: <code>\testsmallspace</code> , <code>\testsmallspace</code> ,
<code>\testsmallspace</code>	<code>\testsmallspace</code> give small (1cm), medium (2cm), and big (3cm) vertical space.
<code>\testsmallspace</code>	<code>\testnewpage</code> makes a new page in <code>test</code> mode, and <code>\testemptypage</code> generates an
<code>\testnewpage</code>	empty page with the cautionary message that this page was intentionally left empty.
<code>\testemptypage</code>	Local Variables: mode: latex TeX-master: <code>../stex-manual</code> End:
	LocalWords: solutions,notes,hints,gnotes,pts,min,boxed,test gnotes elephants,pts gnote
	LocalWords: 2,title exnote hint,exnote,gnote ifsolutions mcb keyvals Ttext Ftext Local-
	Words: Functions,name F,feedback Nooooooooo,feedback 2,title includeproblem Local-
	Words: elephants.fillin,title

## 10.3 HWExam Manual

### 10.3.1 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.

### 10.3.2 Package Options

---

<code>solutions</code>	The <code>hwexam</code> package and class take the options <code>solutions</code> , <code>notes</code> , <code>hints</code> , <code>gnotes</code> , <code>pts</code> , <code>min</code> , and <code>boxed</code> that are just passed on to the <code>problems</code> package (cf. its documentation for a description of the intended behavior).
<code>notes</code>	
<code>hints</code>	
<code>gnotes</code>	
<code>pts</code>	
<code>min</code>	

---

<code>multiple</code>	Furthermore, the <code>hwexam</code> package takes the option <code>multiple</code> 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.).
-----------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<code>test</code>	Finally, there is the option <code>test</code> that modifies the behavior to facilitate formatting tests. Only in <code>test</code> mode, the macros <code>\testspace</code> , <code>\testnewpage</code> , and <code>\testemptypage</code> have an effect: they generate space for the students to solve the given problems. Thus they can be left in the $\text{\LaTeX}$ source.
-------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 10.3.3 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).

### 10.3.4 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.

### 10.3.5 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)

2023-10-13

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

Write the solutions to the sheet.

The estimated time for solving this exam is 23 minutes, leaving you 37 minutes for revising your exam.

You can reach 23 points if you solve all problems. You will only need 27 points for a perfect score, i.e. -4 points are bonus points.

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.	4.1	1.1	1.2	2.1	2.2	2.3	2.4	2.5	2.6	
total	0	0	0	10	0	0	0	0	0	
reached										

good luck

18

Local Variables: mode: latex TeX-master: ../stex-manual End:

LocalWords: hwexam solutions,notes,hints,gnotes,pts,min gnotes testemptypage repts LocalWords: inputassignment reqpts hour,min 60,reqpts

## 10.4 Tikzinput Manual

---

**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}
```

<sup>18</sup>MK: The first three “problems” come from the stex examples above, how do we get rid of this?

```

3 \usetikzpackage{...}
4 \begin{document}
5   \begin{tikzpicture}
6     ...
7   \end{tikzpicture}
8 \end{document}

```

The standalone class is a minimal L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X 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.

---

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

---

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

Local Variables: mode: latex TeX-master: `../stex-manual` End:

**Part III**  
**Documentation**

## Chapter 11

# sTEX Developer Manual



Keeping the implementation properly up-to-date is pretty much incompatible with the kinds of workflows systemically enforced in academia. Any of the following may be out of date.

★ indicates fully expandable functions, which can be used within an `x`-type argument (in plain TEX terms, inside an `\edef`), as well as within an `f`-type argument. ☆ indicates restricted expandable functions, which can be used within an `x`-type argument but cannot be fully expanded within an `f`-type argument. *TF* indicates conditional (`if`) functions whose variants with `T`, `F` and `TF` argument specifiers expect different “true”/“false” branches.

---

`\stex_debug:nn`

`\stex_debug:nn`  $\{\langle prefix \rangle\}$   $\{\langle msg \rangle\}$

Logs the debug message  $\{\langle msg \rangle\}$  under the prefix  $\{\langle prefix \rangle\}$ . A message is shown if its prefix is in a list of prefixes given either via the package option `debug= $\langle prefixes \rangle$`  or the environment variable `STEX_DEBUG= $\langle prefixes \rangle$` , where the latter overrides the former.

---

`\_stex_do_deprecation:n`

TODO `\_stex_do_deprecation:n`



## 11.1 Documents

---

`\l_stex_doheader_sect` integer register keeping track of the current sectioning level:

- 0 part
- 1 chapter
- 2 section
- 3 subsection
- 4 subsubsection
- 5 paragraph
- > 5 subparagraph

`\setsectionlevel` sets `\l_stex_doheader_sect` to the corresponding integer value.

---

`\stex_ref_new_doc_target:n` internal variant of `\sreflabel`. If the argument is empty, the label is determined to be `REF<counter>` and `<counter>` is increased.

---

`\stex_ref_new_symbol:n` registers a new link target for the symbol with the given full uri (as string), using the `url-base`-field of the current archive's manifest file to link the symbol to `<url-base>/symbol?<uri>`.

---

`\stex_ref_new_sym_target:n` sets a new label for the symbol with the given full uri (as string). If called in sms-mode, defers to `\stex_ref_new_symbol:n`, if not already registered. Otherwise, sets a `\label` for the symbol.

---

`\stex_ref_new_sym_target:nn` `\stex_ref_new_sym_target:nn` `{<symbol>}{<target>}`

redirects links for `<symbol>` to the one for the symbol `<target>`. Useful for e.g. symbols elaborated from structural features. Note that this acts as a *default*, in that previous or subsequent calls of `\stex_ref_new_sym_target:n{<symbol>}` are prioritized.

Requires that either `\stex_ref_new_sym_target:n{<target>}` or `\stex_ref_new_sym_target:nn{<target>}{<other-target>}` have been called previously.

## 11.2 Modules

The contents of a module with full URI `<uri>` are represented as four macros:

- `\c_stex_module_<uri>_code`: code to be executed every time the module is *activated*; e.g. the contents of `\STEXexport`, defines semantic macros and macros for notations, activates dependency modules, etc.

- `\c_stex_module_{uri}_morphisms_prop`: property list containing all module dependencies of this module (see `\stex_module_add_morphism:nnn`).
- `\c_stex_module_{uri}symbols_prop`: property list containing all declarations in this module (see `\stex_module_add_symbol:nnnnnnN`).
- `\c_stex_module_{uri}_notations_prop`: property list containing all notations introduced in this module (see `\stex_add_module_notation:nnnnn`).

---

`\l_stex_current_module_str` contains the full URI of the current module.

---

`\l_stex_all_modules_seq` contains the full URIs of all modules currently in scope.

---

`\stex_module_setup:n` `\stex_module_setup:n {<name>}`

Computes the full URI of a new module with name `<name>` in the current namespace, initializes the four macros above and sets `\l_stex_current_module_str` accordingly. Also takes care of correct naming for nested modules, activates the meta theory and loads the signature module if `sig=` was provided (according to `\l_stex_key_sig_str`).

---

`\stex_close_module:` closes the current module; checks whether we are currently in sms mode, and if so, calls `\stex_persist:n` to write the module contents to the sms-file.

---

`\stex_every_module:n` `\stex_every_module:n {<code>}`  
 executes `<code>` every time a new module is opened.

---

`\stex_if_in_module_p:` \* tests whether we are currently in a module.  
`\stex_if_in_module:TF` \*

---

`\stex_if_module_exists_p:n` \*  
`\stex_if_module_exists:nTF` \*

tests whether a module with the given full URI exists, in the sense of *has been parsed at some point in the current document*.

---

`\stex_activate_module:n` activates the module with the given full URI *if and only if* it has not already been activated in the current scope.  
`\stex_activate_module:(o|x)`

---

`\stex_execute_in_module:n` executes the provided code, adds it to the current module activation code, and makes sure the macros defined in it are valid in the current module  $\TeX$  group level.  
`\stex_execute_in_module:x`

This macro is a combination of the following two macros:

---

`\stex_do_up_to_module:n` executes the provided code such that all definitions in it are valid in the current module  
`\stex_do_up_to_module:x` regardless of T<sub>E</sub>X group level (using `\stex_metagroup_do_in:nn`).

---

---

`\stex_module_add_code:n` adds the provided code to the module's `\c_stex_module_{uri}_code`-macro.  
`\stex_module_add_code:x`

---

## 11.3 Symbols

A symbol in S<sub>T</sub>E<sub>X</sub> is represented as a tuple of several components:

---

`\stex_module_add_symbol:nnnnnnnN`

---

- #1 : *⟨id⟩*: An *identifier* (possibly empty) that determines its semantic macro name, or e.g. in `mathstructure` its accessor-identifier (if empty, the name is used for that),
- #2 : *⟨name⟩* a unique *name*, which in combination with the containing module determines its URI as *⟨module-URI⟩?⟨name⟩*,
- #3 : *⟨arity⟩* a numeric string in the range 0..9,
- #4 : *⟨args⟩* a list of argument specifiers of the form *⟨i⟩⟨mode⟩{⟨argname⟩}* (the length of *⟨args⟩* must be  $3 \cdot \langle \text{arity} \rangle$ ),
- #5 : *⟨definiens⟩* (or empty),
- #6 : *⟨type⟩* (or empty),
- #7 : *⟨return code⟩* (or empty), and
- #8 : *⟨invokation\_macro⟩*.

the arguments are stored in the property list `\c_stex_module_{\l_stex_current_module}_symbols_prop` under key *⟨name⟩*.

If the identifier *⟨id⟩* is non-empty, the macro `\id` is defined as `{\stex_invoke_symbol:nnnnnnnN}` with the arguments described there.

**T<sub>E</sub>Xhackers note:** `\invokation_macro` must be `\protected`.

---

`\stex_iterate_symbols:n` `\stex_iterate_symbols:n {⟨code⟩}`

`\stex_iterate_break:`

---

`\stex_iterate_break:n`

---

iterates over all symbols currently in scope and calls *⟨code⟩* for all of them with arguments `{⟨module⟩}{⟨name⟩}{⟨arity⟩}{⟨args⟩}{⟨definiens⟩}{⟨type⟩}{⟨return code⟩}{⟨invokation_macro⟩}`.

Iteration can be stopped prematurely with `\stex_iterate_break:` and can stop and return code with `\stex_iterate_break:n`.

---

`\stex_iterate_symbols:nn` `\stex_iterate_symbols:nn {⟨csl⟩}{⟨code⟩}`

iterates over all symbols in the provided *⟨csl⟩* of full module URIs. *⟨code⟩* receives the same arguments as `\stex_iterate_symbols:n`, but iteration can not be stopped early.

---

```

\stex_get_symbol:n
\l_stex_get_symbol_mod_str
\l_stex_get_symbol_name_str
\l_stex_get_symbol_arity_int
\l_stex_get_symbol_args_tl
\l_stex_get_symbol_def_tl
\l_stex_get_symbol_type_tl
\l_stex_get_symbol_return_tl
\l_stex_get_symbol_invoke_cs

```

---

`\stex_get_symbol:n` attempts to find a symbol with the given name or id that is currently in scope. A name may be prefixed with a module name/path separated by ?.

Throws an error if no such symbol is found; otherwise, sets the listed `\l_stex_get_symbol_...` macros to the components of the found symbol.

---

```

\stex_if_check_terms_p: *
\stex_if_check_terms:TF *

```

---

whether to typeset declaration components (notations, types, definientia etc.) in a throw-away box. Is true iff the backend is `pdflatex` and either the `STEX_CHECKTERMS` environment variable or `checkterms` package option is set.

---

```

\stex_check_term:n

```

---

typesets the argument in a throwaway box in math mode iff `\stex_if_check_terms:` is true.

Is deactivated in sms-mode.

---

```

\stex_symdecl_do:
\l_stex_key_name_str
\l_stex_key_args_str
\l_stex_key_argnames_clist
\l_stex_assoc_args_count
\l_stex_argnames_seq

```

---

`\stex_symdecl_do:` processes the shared (mandatory and optional) arguments of e.g. `\symdecl`, `\symdef`, `\vardef` etc.

Requires the following macros to be set

- `\l_stex_key_name_str`: the name of the symbol,
- `\l_stex_key_args_str`: the `args`-string (e.g. `3` or `ai`)
- `\l_stex_key_argnames_clist`: a list of *names* for the arguments, the length of which should be  $\leq$  the arity of the symbol

and will generate the following macros:

- `\l_stex_get_symbol_arity_int`: the arity of the symbol,
- `\l_stex_key_args_str`: the `args` string as a definite sequence of argument-mode characters, whose length is the arity of the symbol; e.g. `3` is turned into `iii`,
- `\l_stex_assoc_args_count`: the number of sequence arguments (i.e. `a` or `B` mode),
- `\l_stex_argnames_seq`: the full sequence of argument names; those not provided by `\l_stex_key_argnames_clist` are set to be `$j`, where `j` is the index of the argument,
- `\l_stex_get_symbol_args_tl`: a token list of triples `jm{<argname>}`, where `j` is the index and `m` the respective argument mode character (i.e. `i`, `a`, `b` or `B`).

---

`\stex_symdecl_check_terms:`

calls `\stex_check_term:n` for the type and definiens stored in `\l_stex_key_type_tl` and `\l_stex_key_def_tl`

---

`\stex_symdecl_top:n`

`\stex_symdecl_top:n`  $\langle\text{maybe name}\rangle$

checks whether `\l_stex_key_name_str` is empty, and if so, sets it to be  $\langle\text{maybe name}\rangle$ . Then calls `\stex_symdecl_do:` and `\stex_symdecl_check_terms:`, writes the components to the HTML (if applicable) and adds the symbol to the current module with invocation macro `\stex_invoke_symbol:` and id/macroname `\l_stex_macroname_str`.

Variables work very similar to symbols, except that their declarations are local to the current  $\text{\TeX}$ -group rather than the current module, and they are not exported in modules.

---

`\l_stex_variables_prop`

stores all variables currently in scope as a property list with key  $\langle\text{name}\rangle$  and value  $\langle\text{id}\rangle\langle\text{name}\rangle\langle\text{arity}\rangle\langle\text{args}\rangle\langle\text{definiens}\rangle\langle\text{type}\rangle\langle\text{return code}\rangle\langle\text{invokation macro}\rangle$ .

The invocation macro for “normal” variables declared with `\vardef` is `\stex_invoke_symbol:`.

---

`\stex_vardecl_notation_macro:`

generates the notation macro for a variable, based on the values of the `\l_stex_key_` macros und `\l_stex_notation_macrocode_cs`.

---

`\stex_get_var:n`

like `\stex_get_symbol:n`, but attempts to retrieve a variables and throws an error if none is found.

---

`\stex_get_symbol_or_var:n`

like `\stex_get_symbol:n`, but first attempts to find a *variable*, and if none is found, defers to `\stex_get_symbol:n`.

## 11.4 Notations

---

`\stex_module_add_notation:nnnnn`

`\stex_module_add_notation:eoexo`

`\stex_module_add_notation:nnnnn`  
 $\langle\text{symboluri}\rangle\langle\text{id}\rangle\langle\text{arity}\rangle\langle\text{code}\rangle\langle\text{op code}\rangle$

stores the arguments in the property list `\c_stex_module_` $\langle\text{\l\_stex\_current\_module}\rangle$ `_notations_prop` under key  $\langle\text{symboluri}\rangle\langle\text{id}\rangle$  and calls `\stex_set_notation_macro:nnnnn`.

---

```

\stex_set_notation_macro:nnnnn \stex_set_notation_macro:nnnnn
\stex_set_notation_macro:eoexo {\symboluri}{id}{arity}{code}<op code>

```

---

Declares the following macros:

An active notation for a symbol with uri  $\langle symboluri \rangle$  and id  $\langle id \rangle$  is represented as a macro  $\backslash l\_stex\_notation\_ \langle symboluri \rangle\_ \langle id \rangle\_ cs$ , that takes  $\langle arity \rangle$  many argument and expands to  $\langle code \rangle$ , and (if nonempty) an *operator* notation as a macro  $\backslash l\_stex\_notation\_ \langle symboluri \rangle\_ op\_ \langle id \rangle\_ cs$  that expands to  $\langle op code \rangle$ .

The default notation is represented as the *empty* id. If the corresponding macros  $\backslash l\_stex\_notation\_ \langle symboluri \rangle\_ cs$ , and  $\backslash l\_stex\_notation\_ \langle symboluri \rangle\_ op\_ cs$  do not yet exist, they are now defined as  $\langle id \rangle$ .

---

```

\stex_iterate_notations:nn \stex_iterate_notations:nn {\csl}{code}

```

---

iterates over all notations in the provided  $\langle csl \rangle$  of full module URIs and calls  $\langle code \rangle$  on each of them with arguments  $\{\langle symboluri \rangle\}\{\langle id \rangle\}\{\langle arity \rangle\}\langle code \rangle\langle op code \rangle$ .

---

```

\stex_notation_parse_and_then:nw \stex_notation_parse_and_then:nw {\code}{notations}
\l_stex_key_prec_str
\l_stex_key_variant_str
\l_stex_notation_macrocode_cs

```

---

parses a notation (which may consist of multiple braced components, depending on the argument modes) and subsequently calls  $\langle code \rangle$ .

Requires the following macros to be set:

- $\backslash l\_stex\_get\_symbol\_arity\_int$ ,
- $\backslash l\_stex\_get\_symbol\_args\_tl$ ,
- $\backslash l\_stex\_key\_prec\_str$ : The precedence string as provided by a user in the optional `prec`-argument,
- $\backslash l\_stex\_key\_variant\_str$ : the id of the notation variant.

Stores the final notation in the macro  $\backslash l\_stex\_notation\_macrocode\_cs$  taking  $\backslash l\_stex\_get\_symbol\_arity\_int$  many arguments.

Some thing to consider when we generate a notation macro:

- The notation macro generated by the `\notation`-command should contain the variant identifier and the precedences, as these are intrinsic parts of the notation.
- It should *not* contain the symbol itself however, so that notations can be copied between symbols.
- Notations as provided by users will largely adhere to the standard L<sup>A</sup>T<sub>E</sub>X category code scheme, and
- notations need to be “persistable” in `.deps`-files.

We therefore want to augment the simple code provided by a user by various “annotations” that contain the relevant information (such as precedences) and to mark the

argument positions for semantic extractions, but we should adhere to the default L<sup>A</sup>T<sub>E</sub>X category code scheme in doing so.

---

```

\STEXInternalTermMathArgiii
\STEXInternalTermMathAssocArgiiii
\STEXInternalAssocArgMarkerI
\STEXInternalAssocArgMarkerII
\STEXInternalTermMathOMSOrOMViii
\STEXInternalTermMathOMAiii
\STEXInternalTermMathOMBiii
\STEXInternalSymbolAfterInvokationTL

```

---

In **OPENMATH**/OMDOC, there are (for our purposes) three kinds of expressions that an application of a semantic macro – and hence a notation macro – can represent, each of which corresponds to a macro taking care of the semantic annotations:

- OMS/OMV: a simple symbol (arity 0) (`\STEXInternalTermMathOMSOrOMViii`)
- OMA: an application of a symbol to argument (arity > 0, `\STEXInternalTermMathOMAiii`)
- OMB: a binding application of a symbol that binds/declares one or more variables (argument string contains **b** or **B**, `\STEXInternalTermMathOMBiii`).

The arguments are marked with `\STEXInternalTermMathArgiii` (**i** or **b**) or `\STEXInternalTermMathAssocArgiiii` (**a** or **B**). Finally, the notation is closed with `\STEXInternalSymbolAfterInvokationTL`.

How this works is best explained by example.

### Example 30

Assume we have a symbol and notation:

```

1 \symdecl{somesymbol}[args=iai]
2 \notation{somesymbol}[prec=10;20x30x40,variant=foo]
3 {First: #1; Second: #2; Third: #3; End}
4 {(#1 -- ##1 split ##2 -- #3)}

```

Since the symbol corresponds to an OMA, the whole notation is wrapped in `\STEXInternalTermMathOMAiii`, taking as arguments the variant identifier (`foo`), the operator precedence (10) and the body of the notation.

The second argument in the notation, being associative, is wrapped in a `\STEXInternalTermMathAssocArgiiii`, taking as arguments the argument number (2), the precedence (30), the T<sub>E</sub>X parameter token (`#2`) the notation body (`(#1 -- ##1 split ##2 -- #3)`), and finally the argument mode (`a`). Additionally, the markers `##1` and `##2` are replaced by `\STEXInternalAssocArgMarkerI` and `\STEXInternalAssocArgMarkerII`, respectively.

Subsequently, the non-sequence parameter tokens are wrapped in `\STEXInternalTermMathArgiii` with arguments `mj` (where `m` is the mode and `j` the index), the precedence (20 or 40 respectively), and the parameter token.

Finally, a `\STEXInternalSymbolAfterInvokationTL` is inserted.

The final expansion of `\l_stex_notation_macrocode_cs` is thus:

```

1 \STEXInternalTermMathOMAiii{foo}{10}{
2   First: \STEXInternalTermMathArgiii{i1}{20}{#1};
3   Second: \STEXInternalTermMathAssocArgiiii{2}{30}{#2}{
4     (\STEXInternalTermMathArgiii{i1}{20}{##1} --

```

```

5     \STEXInternalAssocArgMarkerI split
6     \STEXInternalAssocArgMarkerII --
7     \STEXInternalTermMathArgiii{i3}{40}{##3})
8   }{a};
9   Third: \STEXInternalTermMathArgiii {i3}{40}{#3}; End
10 }\STEXInternalSymbolAfterInvokationTL

```

---

`\stex_notation_top:nnw` `\stex_notation_top:nnw`  $\langle symboluri \rangle$   $\langle code \rangle$

calls `\stex_notation_parse_and_then:nw` $\langle code \rangle$  and, adds the notation for the symbol with URI  $\langle symboluri \rangle$  to the current module and exports it to the HTML (if applicable).

## 11.5 Structural Features

---

`\stex_structural_feature_module:nn`  
`\stex_structural_feature_module_end:` `\stex_structural_feature_module:nn`  $\langle name \rangle$   $\langle typeid \rangle$   
`\g_stex_last_feature_str`

---

opens a new module-like structural feature of type  $\langle typeid \rangle$  with name  $\langle name \rangle$ , which needs to be closed with `\stex_structural_feature_module_end:`.

Its body behaves like a nested module with name  $\langle modulename \rangle / \langle name \rangle$ , the full URI of which is stored in `\g_stex_last_feature_str` for subsequent elaboration.



---

```

\stex_structural_feature_morphism:nnnnn \stex_structural_feature_morphism:nnnnn
\stex_structural_feature_morphism_end:  {\langle morphismname \rangle}{\langle typeid \rangle}{\langle archive \rangle}{\langle domain \rangle}{\langle annotations \rangle}
\l_stex_current_domain_str
\l_stex_feature_name_str
\l_stex_morphism_symbols_prop
\l_stex_morphism_renames_prop
\l_stex_morphism_morphisms_seq

```

---

opens a new morphism-like structural feature of type  $\langle typeid \rangle$  with name  $\langle morphismname \rangle$  and the module  $[\langle archive \rangle]{\langle domain \rangle}$  as domain, which needs to be closed with `\stex_structural_feature_morphism_end:`.

Deactivates `\symdecl`, `\textsymdecl`, `\symdef`, `\notation` and `smodule`, and activates `\assign`, `\assignMorphism` and `\renamedec1`.

Defines the following macros:

- `\l_stex_feature_name_str` =  $\langle name \rangle$ .
- `\l_stex_current_domain_str` = the full uri of  $\langle domain \rangle$ .
- `\l_stex_morphism_symbols_prop`: This property list is initialized as follows: For every symbol transitively included in  $\langle domain \rangle$  with data  $\langle module \rangle$ ,  $\langle name \rangle$ ,  $\langle id \rangle$ ,  $\langle arity \rangle$ ,  $\langle args \rangle$ ,  $\langle definiens \rangle$ ,  $\langle type \rangle$ , and  $\langle return code \rangle$ , the property list contains an entry with key  $[\langle module \rangle]/[\langle name \rangle]$  and value  $\{\langle id \rangle\}\{\langle arity \rangle\}\{\langle args \rangle\}\{\langle definiens \rangle\}\{\langle type \rangle\}\{\langle return code \rangle\}$ .
- `\l_stex_morphism_renames_prop`: An initially empty property list.
- `\l_stex_morphism_morphisms_seq`: TODO

At `\stex_structural_feature_morphism_end:`, the elaboration is computed from the above data thusly:

For every entry in `\l_stex_morphism_symbols_prop`, a new symbol is created with the values  $\langle arity \rangle$ ,  $\langle args \rangle$ ,  $\langle definiens \rangle$ ,  $\langle type \rangle$  and  $\langle return code \rangle$  from that property list, and either:

- if `\l_stex_morphism_renames_prop` does *not* contain an entry with key  $\langle module \rangle?\langle name \rangle$ , then the elaborated name is  $\langle morphismname \rangle/\langle name \rangle$  and its  $\langle id \rangle$  is empty (no semantic macro is generated), or
- if `\l_stex_morphism_renames_prop` contains an entry with key  $\langle module \rangle?\langle name \rangle$ , then its value needs to be of the form  $\{\langle id \rangle\}\{\langle name \rangle\}$ , which are used for the elaborated symbol.

All notations of the symbols transitively included in the domain are copied over to their elaborations.

## 11.6 Imports and Morphisms

---

```
\stex_module_add_morphism:nnnn
\stex_module_add_morphism:(nonn|ooox)
```

---

adds a new module morphism (i.e. inheritance, possibly with modification) to the current module

- #1 : The name of the morphism (may be empty for e.g. `\importmodule`, but may be named for e.g. `copymodule`),
- #2 : the URI of the module being “imported”,
- #3 : the “type” of the morphism (e.g. `import` or `copymodule`),
- #4 : a list of assignments as pairs  $\{\langle source \rangle\}\{\langle target \rangle\}$  that signify that the symbol with full URI  $\langle source \rangle$  is being mapped or elaborated to the new symbol with name  $\langle target \rangle$  in the current module. May be empty for e.g. `\importmodule`.

The provided arguments are stored in `\c_stex_module_⟨uri⟩_morphisms_prop` with key #1 (if non-empty) or [#2] (if #1 is empty) and value  $\{\#1\}\{\#2\}\{\#3\}\{\#4\}$

Import-like statements in  $\text{\TeX}$  are usually given as pairs  $[\langle archive \rangle]\{\langle path \rangle?\langle module \rangle\}$ , which be relative to the current archive and/or file path. For persistence and sms-mode, these pairs first need to be resolved into an *absolute* specification:

---

```
\stex_import_module_uri:nn
\l_stex_import_archive_str
\l_stex_import_name_str
\l_stex_import_uri_str
\l_stex_import_path_str
```

---

`\stex_import_module_uri:nn`  $\{\langle archive \rangle\}\{\langle pathstring \rangle\}$

(where  $\langle archive \rangle$  may be empty) resolves the given arguments into:

- `\l_stex_import_archive_str` the given archive id (in which case `\stex_require_archive:n` is called), or the id of the current archive (if existent and  $\langle archive \rangle$  empty), or empty,
- `\l_stex_import_uri_str` if an archive id is given, or we currently are in an archive, its corresponding namespace; otherwise `\file:`,
- `\l_stex_import_path_str` the path of the URI relative to the given (or current) archive, or (if not existent) the absolute path of  $\langle pathstring \rangle$  (without a module name) resolved relative to the current file’s parent directory, and
- `\l_stex_import_name_str` the name of the module.

If  $\langle pathstring \rangle$  does not contain the character `?`, the whole pathstring is assumed to be the name of the module and the path is empty (or the current file’s parent directory, depending on the above).

---

```
\stex_import_require_module:nnn
```

---

takes as arguments values `\l_stex_import_archive_str`, `\l_stex_import_path_str` and `\l_stex_import_name_str` as computed by `\stex_import_module_uri:nn` and (optionally loads and) activates the corresponding module (or throws an error if it does not exist).

Most complex morphisms (`copymodule` et al) are implemented as structural features using `\stex_structural_feature_morphism:nnnn`.

---

```
\stex_get_in_morphism:n
\l_stex_get_symbol_macro_str
```

---

finds a symbol with the provided name or id in the domain of the current morphism. Sets the same macros as `\stex_get_symbol:n`, and additionally `\l_stex_get_symbol_macro_str` to the  $\langle id \rangle$  of the symbol. Throws an error if no such symbol is found.

## 11.7 Expressions and Semantic Macros

---

```
\_stex_invoke_symbol:nnnnnnN   \l_stex_current_symbol_str   \_stex_invoke_symbol:nnnnnnN
\l_stex_current_arity_str      {<module>}{<name>}{<arity>}{<args>}{<definiens>}
\l_stex_current_args_tl       {<type>}
\l_stex_current_return_tl     {<return code>}
\l_stex_current_type_tl       {\invokation_macro}
```

---

is how a semantic macro is/should be defined. `\_stex_invoke_symbol:nnnnnnN` first checks whether semantic macros are currently allowed, and throws an error if not. Otherwise, it sets the `\comp`-controlled highlighting to `\compemph@uri`, initializes `\STEXInternalSymbolAfterInvokationTL`, defines the following macros for all of its arguments, and subsequently calls the `\invokation_macro`:

- `\l_stex_current_symbol_str ={\langle module \rangle? \langle name \rangle}`
- `\l_stex_current_arity_str ={\langle arity \rangle}`
- `\l_stex_current_args_tl ={\langle args \rangle}`
- `\l_stex_current_type_tl ={\langle type \rangle}`
- `\l_stex_current_return_tl ={\langle return code \rangle}`

The simplest example for an `\invokation_macro` is `\stex_invoke_symbol:.`

---

```
\_stex_invoke_variable:nnnnnnN
```

---

analogous to `\_stex_invoke_symbol:nnnnnnN`, but for variables; sets the `\comp`-controlled highlighting to `\varemp@uri`.

---

```
\stex_invoke_symbol: branches based on the mode and following characters:
```

---

- If math, check next character:
  - ! operator; defer to operator notation
  - else defer to notation and check the value of `\l_stex_current_return_tl = \langle return \rangle`.
    - \* If  $\langle return \rangle$  is empty, call the notation macro,
    - \* otherwise, call  $\langle return \rangle \{ \backslash \text{stex\_invoke\_symbol} ! \}$ .
- If text:

---

```
\stex_invoke_sequence_range:
\stex_invoke_sequence_in:
```

---

TODO

## 11.8 Optional (Key-Value) Argument Handling

L<sup>A</sup>T<sub>E</sub>X3 is surprisingly weak when it comes to handling optional (key-val) arguments in such a manner that *only* the freshly set macros are defined, and to modularly build up sets of argument keys. The following macros attempt to fix that:

---

```
\stex_keys_define:nnnn \stex_keys_define:nnnn {<id>}{<setup code>}
\stex_keys_set:nn      {<keyval setup code>}{<parents>}
```

---

Defines a set of keys and their allowed values with identifier `stex/<id>`, that inherits from the sets with identifiers in `<parents>`.

`\stex_keys_set:nn {<id>}{<CSL>}` first executes `<setup code>` (e.g. to empty the macros holding the values) and then sets the keys in set `<id>` with the values provided in `<CSL>`.

---

```
\_stex_do_id:
```

---

should be called whenever a macro or environment has a label `id`, i.e. calls `\stex_keys_set:nn{id}{...}`, after the title has been typeset. Sets a `\label` by calling `\stex_ref_new_doc_target:n{id}`.

### Example 31

If we define a set of keys with:

```
1 \stex_keys_define:nnnn{archive file}{
2   \str_clear:N \l_stex_key_archive_str
3   \str_clear:N \l_stex_key_file_str
4 }{
5   archive .str_set_x:N = \l_stex_key_archive_str ,
6   file .str_set_x:N = \l_stex_key_file_str
7 }{id}
```

then calling `\stex_keys_set:nn{archive file}{id=foo,file=bar}` sets `\l_stex_key_file_str={bar}`, assures that `\l_stex_key_archive_str` is empty, and executes the code associated with `id`, i.e. it sets `\l_stex_key_id_str={foo}`.

## 11.9 Styable Commands and Environments

---

`\stex_new_stylable_cmd:nnnn` `\stex_new_stylable_cmd:nnnn`  $\langle name \rangle$   $\langle arity \rangle$   $\langle code \rangle$   
 $\langle default \rangle$

Creates a new macro  $\langle name \rangle$  with expansion  $\langle code \rangle$  taking  $\langle arity \rangle$  many arguments, that is customizable in presentation by a user by calling `\stexstyle` $\langle name \rangle$ . On calling `\stex_style_apply`: executes the presentation code provided by a user.  $\langle code \rangle$  should:

- Call `\stex_keys_set:nn` $\{style\}\{\dots\}$  (or a keyset inheriting from `style`),
- set macros with prefix `\this...` that a user might want to use for presentation (e.g. `\thistitle`),
- call `\stex_style_apply`: at some point.

---

`\stex_new_stylable_env:nnnnnn` `\stex_new_stylable_env:nnnnnn`  $\langle name \rangle$   $\langle arity \rangle$   $\langle begincode \rangle$   
 $\langle endcode \rangle$   $\langle default begin \rangle$   $\langle default end \rangle$   $\langle prefix \rangle$

Like `\stex_new_stylable_cmd:nnnn`, but defines a new environment  $\langle prefix \rangle \langle name \rangle$  stylable via `\stexstyle` $\langle name \rangle$ . Should call `\stex_style_apply`: twice; once in the  $\langle begincode \rangle$  and once in  $\langle endcode \rangle$ .

---

`\stex_style_apply`: Sets `\thisstyle` to be the head of the CSL `\l_stex_key_style_clist` and checks whether a style for the current stylable macro/environment has been defined; if not, executes the code for the default style.

### Example 32

`\importmodule` is defined something like the following:

```
1 \stex_new_stylable_cmd:nnnn{importmodule}{0}{ m}{
2   ...
3   \def\thismoduleuri{...}
4   \def\thismodulename{...}
5   \stex_style_apply:
6   ...
7 }{}
```

A user can then customize the output generated by `\importmodule` (by default none) via `\stexstyleimportmodule{...}\thismodulename...`.

### Example 33

`smodule` does something like

```
1 \stex_new_stylable_env:nnnnnn{module}{0}{ m}{
2   ...
3   \def\thismoduleuri{...}
4   \def\thismodulename{...}
5   \stex_style_apply:
6   ...
7 }{
8   ...
```

```

9 \stex_style_apply:
10 ...
11 }{}{}{s}

```

which defines the environment name to be `smodule` and generates `\stexstylemodule`.

## 11.10 Math Archives

Math archives are represented as L<sup>A</sup>T<sub>E</sub>X3 property lists, the keys/values of which correspond to the entries in the archive’s manifest file. The most important fields are

- `id`,
- `narr` the document namespace,
- `ns` the content namespace, and
- `docurl` the document URL base.

---

```

\stex_resolve_path_pair:Nnn \stex_resolve_path_pair:Nnn {\target}{\archive-id}{\pathstring}
\stex_resolve_path_pair:Nxx

```

computes the absolute file path of  $\langle pathstring \rangle$  relative to the `source`-folder of  $\langle archive-id \rangle$  (if non-empty), or the current archive (if existent) or the parent working directory (otherwise), and stores the result in `\target`.

---

```

\l_stex_current_archive_prop

```

`\l_stex_current_archive_prop` always points to the current math archive or is `\undefined`, if the current file is not part of a math archive.

---

```

\c_stex_main_archive_prop \c_stex_main_archive_prop represents the math archive in which the main file resides

```

(if existent).

---

```

\stex_require_archive:n \stex_require_archive:n {\id}
\stex_require_archive:o

```

looks for a math archive  $\langle id \rangle$  in the MathHub directory, parses its manifest file, creates the corresponding property list `\c_stex_mathhub_⟨id⟩_manifest_prop`, and throws a fatal error if the archive is not found.

If the archive has been found and parsed before, does nothing, so it is cheap and safe to call repeatedly for the same id.

---

```

\stex_set_current_archive:n \stex_set_current_archive:n {\id}

```

Calls `\stex_require_archive:n{⟨id⟩}` and sets `\l_stex_current_archive_prop`.

---

```

\stex_in_archive:nn \stex_in_archive:nn {\opt-id}{\code}

```

Executes  $\langle code \rangle$  in the context of math archive  $\langle opt-id \rangle$  (using `\stex_require_archive:n`), i.e. iff  $\langle opt-id \rangle$  is non-empty, changes the current archive to the one with id  $\langle opt-id \rangle$ , call  $\langle code \rangle$  with  $\langle opt-id \rangle$  as argument (in `#1`) and changes it back afterwards.

If  $\langle opt-id \rangle$  is empty,  $\langle code \rangle$  is called with the id of the *current* math archive as `#1`, or with `#1` empty if there is no current math archive.

## 11.11 SMS-Mode

$\S\TeX$  has to extract formal content (i.e. modules and their symbols) from  $\LaTeX$ -files, that may otherwise contain arbitrary code, including macros that may not be defined unless the file is fully processed by  $\TeX$ . Those modules and symbols also may depend on other modules that have not yet been loaded. The naive way to achieve this, which would be to just suppress output (e.g. by storing it in a box register) and then  $\input$  the required file, does not work thanks to  $\TeX$ 's limited *file stack*, which would overflow quickly for modules that have a deeply nested list of dependencies.

To solve those problems,  $\S\TeX$  reads dependencies in what we call *sms mode*, which can be summarized thusly:

- In a first pass, we parse the file token by token, ignoring everything other than a select list of macros and environments that introduce dependencies (such as  $\importmodule$  and  $\begin{smodule}[sig=...]$ ). Instead of loading those, we remember them for later.
- After the file has been fully parsed thusly, the dependencies found are loaded, again in sms-mode.
- In a second pass, we parse the file *again* in the same way, but this time execute all macros that are explicitly allowed in sms mode, such as  $\importmodule$ ,  $\symdecl$ ,  $\notation$ ,  $\symdef$ , etc.
- all this parsing happens additionally in a  $\setbox\throwaway\ vbox{...}$ -block to suppress any accidental output.

This means that  $\TeX$ 's input stack never grows by more than +1, but still behaves *as if* the dependencies were loaded recursively, at the detriment of being somewhat slow.

---

$\stex\_if\_smsmode\_p$ : \* tests for whether we are currently in sms-mode.  
 $\stex\_if\_smsmode$ : *TF* \*

---

$\stex\_file\_in\_smsmode$ :nn  $\stex\_file\_in\_smsmode$ :nn  $\langle filestring \rangle \langle setup-code \rangle$   
 $\stex\_file\_in\_smsmode$ :on

sets up sms-mode and internal grouping, calls  $\langle setup-code \rangle$  and subsequently processes the file  $\langle filestring \rangle$  in sms-mode as described above.

### 11.11.1 Second Pass

---

$\stex\_sms\_allow$ :N  $\stex\_sms\_allow$ :N  $\langle \macro \rangle$

registers the  $\macro$ -command to be allowed in sms mode.

This only works, if  $\macro$  takes no arguments and/or does not touch the subsequent tokens.

For macros taking arguments, we can use

---

`\stex_sms_allow_escape:N` `\stex_sms_allow_escape:N`  $\{\langle\text{macro}\rangle\}$

registers the `\macro`-command to be allowed in sms mode.

If `\macro` is subsequently encountered in sms-mode, parsing is halted and `\macro` can process arguments as desired. It then needs to continue parsing manually though, by calling `\stex_smsmode_do:` as (usually) its last token.

---

`\stex_sms_allow_env:n` `\stex_sms_allow_env:n`  $\{\langle\text{envname}\rangle\}$

registers the environment  $\langle\text{envname}\rangle$  to be allowed in sms mode.

As with `\stex_sms_allow_escape:N`, the `\begin{\langle\text{envname}\rangle}` is escaped, hence the begin-code of the environment needs to call `\stex_smsmode_do:.` Since `\end{\langle\text{envname}\rangle}` never takes arguments, it does not need to be escaped.

---

`\stex_smsmode_do:` continues with sms-mode-style parsing. Does nothing if not in sms-mode, and is therefore safe to be called “just in case”.

### 11.11.2 First Pass

---

`\stex_sms_allow_import:Nn`  
`\stex_sms_allow_import_env:nn`

behave like `\stex_sms_allow_escape:N` and `\stex_sms_allow_env:n` respectively, but the macro or environment provided is now allowed in the *first* pass of sms-mode.

This macro should process arguments, add content to `\g_stex_sms_import_code`, and finally call `\stex_smsmode_do:.`

The code provided in the *second* argument is called before the first pass of sms-mode, as to set up functionality for these macros. For example, `\importmodule` provides code that redefines `\importmodule` to store the identified dependency in `\g_stex_sms_import_code` instead of activating it directly.

---

`\g_stex_sms_import_code` is built up in the first pass of sms mode and called subsequently; before the second pass. Code in this token list should load and activate dependencies found in the first pass.

## 11.12 Strings, File Paths, URIs

---

`\stex_str_if_starts_with_p:nn`  $\star$  `\stex_str_if_starts_with:nn`  $\{\langle\text{first}\rangle\}\{\langle\text{second}\rangle\}$   
`\stex_str_if_starts_with:nnTF`  $\star$

Checks whether the string  $\langle\text{first}\rangle$  starts with the string  $\langle\text{second}\rangle$  (i.e.  $\langle\text{second}\rangle$  is a prefix of  $\langle\text{first}\rangle$ ).

---

`\stex_str_if_ends_with_p:nn`  $\star$  `\stex_str_if_ends_with:nn`  $\{\langle\text{first}\rangle\}\{\langle\text{second}\rangle\}$   
`\stex_str_if_ends_with:nnTF`  $\star$

Checks whether the string  $\langle\text{first}\rangle$  ends with the string  $\langle\text{second}\rangle$  (i.e.  $\langle\text{second}\rangle$  is a suffix of  $\langle\text{first}\rangle$ ).



### 11.12.1 File Paths

*File paths* are represented as L<sup>A</sup>T<sub>E</sub>X<sub>3</sub> sequences. The following methods make sure to

- canonicalize paths, i.e. resolve `..` and `.` segments,
- set all category codes to 12 (other), and
- transform windows file paths containing `\` uniformly to `/`.

---

```
\stex_file_resolve:Nn
\stex_file_resolve:(No|Nx)
```

---

```
\stex_file_resolve:Nn {\macro}{\string}
```

resolves and canonicalizes the file path string `\string` and stores the result in `\macro`.

---

```
\stex_file_set:Nn
\stex_file_set:(No|Nx)
```

---

```
\stex_file_set:Nn {\macro}{\string}
```

represents an already canonicalized file path string as a L<sup>A</sup>T<sub>E</sub>X<sub>3</sub> sequence and stores it in `\macro`.

---

```
\stex_if_file_absolute_p:N *
\stex_if_file_absolute:NTF *
```

---

`\stex_if_file_absolute:N` tests whether the given file path (represented as a canonicalized L<sup>A</sup>T<sub>E</sub>X<sub>3</sub> sequence) is an absolute file path.

---

```
\stex_file_use:N *
```

---

`\stex_file_use:N` expands to a string representation of the given file path.

---

```
\stex_if_file_starts_with:NNTF
```

---

```
\stex_if_file_starts_with:NN {\first}{\second}
```

tests whether the file path `\first` is a child of `\second`. (*Not expandable*)

---

```
\stex_file_split_off_ext:NN
```

---

```
\stex_file_split_off_ext:NN {\target}{\source}
```

splits off the file extension of `\source` and stores the resulting file path in `\target`

---

```
\stex_file_split_off_lang:NN
```

---

```
\stex_file_split_off_lang:NN {\target}{\source}
```

checks whether the file path `\source` ends with a language abbreviation (e.g. `.en`), if so removes it, and stores the result in `\target`.

The following are primarily used in file hooks, but might occasionally be useful to call manually:

---

```
\stex_filestack_push:n
```

---

pushes the given file to the file stack, recomputing `\g_stex_current_file`, the current language, document URI and namespace.

---

```
\stex_filestack_pop:
```

---

pops the current top entry of the file stack. If the file stack is empty, resets to `\c_stex_main_file`.

## File Path Constants and Variables

---

`\c_stex_pwd_file` store the parent working directory and the absolute path of the main file being processed  
`\c_stex_main_file` (with guessed file extension `.tex`).

---

`\c_stex_home_file` stores the user's home directory.

---

`\c_stex_mathhub_file` stores the user's MathHub directory; its string representation is stored in `\mathhub`.

---

`\g_stex_current_file` always points to the *current* file.

### 11.12.2 URIs

MMT URIs are represented as token lists of the form

`{\__stex_path_auth:n{<authority>}}{\__stex_path_path:n{<path>}}{\__stex_path_module:n{<modulename>}}{\__stex_path_name:n{<declname>}}`,

all of which may be empty. Largely, URIs are used as strings only, but the above representation is used in `\stex_uri_resolve:Nn` to canonicalize URIs when they are computed the first time.

---

`\stex_map_uri:Nnnnn` `\stex_map_uri:Nnnnn` `{\uri}{<authority code>}`  
`{<path code>}{<modulename code>}{<declname code>}`  
 executes the provided `<code>`s with the components of the `\uri` as arguments.

---

`\stex_uri_resolve:Nn` behaves analogously to `\stex_file_resolve:Nn`.  
`\stex_uri_resolve:(No|Nx)`

---

`\stex_uri_set:Nn` behaves analogously to `\stex_file_set:Nn`.  
`\stex_uri_set:(No|Nx)`

---

`\stex_uri_use:N` `*` behaves analogously to `\stex_file_use:N`.

A common usage of URIs is computing the namespace of content elements (modules or documents) from the namespace of a math archive and some relative file path within that archive.

---

`\stex_uri_from_repo_file:NNNn` `\stex_uri_from_repo_file:NNNn` `{\target}{\repo_prop}{\filepath}`  
`{<ns_field>}`

computes the namespace URI from the property list `\repo_prop` of some math archive, the file path `\filepath` and the archive field `{<ns_field>}` (`narr` or `ns`), and stores the result in `\target`.

---

`\stex_uri_from_repo_file_nolang:NNNn`

behaves like `\stex_uri_from_repo_file:NNNn`, but makes sure to split off language abbreviations from the file name (e.g. `.en`).

---

`\stex_uri_from_current_file:Nn`

`\stex_uri_from_current_file_nolang:Nn`

Special cases for `\stex_uri_from_repo_file[_nolang]:NNNn`, for `\repo_prop=\l_stex_current_archive_prop` and `\filepath=\g_stex_current_file`.

---

`\stex_set_document_uri:` sets the current value of `\l_stex_current_doc_uri` based on the current file and archive.

---

`\stex_set_current_namespace:`

sets the current value of `\l_stex_current_ns_uri` based on the current file and archive.

---

`\stex_uri_add_module:NNn`

`\stex_uri_add_module:NNn` `{\target}{\uri}`  
`\stex_uri_add_module:NNo` `{\name}`

Checks that URI `\uri` has no module name, adds the provided `\name` and stores the result in `\target`.

### URI Constants and Variables

---

`\l_stex_current_doc_uri` always points to the current document URI.

---

`\l_stex_current_ns_uri` always points to the current content namespace.

## 11.13 Language Handling

---

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

Property lists converting babel languages to/from their abbreviations; e.g.

- `\prop_item:Nn \c_stex_languages_prop {de}` yields `ngerman`, and
- `\c_stex_language_abbrevs_prop {ngerman}` yields `de`.

---

`\l_stex_current_language_str`

always stores the current language.

---

`\stex_set_language:n`  
`\stex_set_language:(x|o)`

---

`\stex_set_language:n`  $\langle abbrev \rangle$

Sets `\l_stex_current_language_str`, and, if the `babel` package is loaded, calls `\selectlanguage` on the language corresponding to  $\langle abbrev \rangle$ .

Note that the package option `lang=` automatically loads the `babel` package.

If  $\langle abbrev \rangle = \text{tr}$ , additionally call `\selectlanguage` with the option `shorthands=!`.

Throws `error/unknownlanguage` if no language with abbreviation  $\langle abbrev \rangle$  is known.

---

`\stex_language_from_file:`

---

infers the current language from file ending (e.g. `.en.tex`) and sets it appropriately.

Is called in a file hook, i.e. always switches language when inputting a file `.<lang>.<ext>`.

## 11.14 Inserting Annotations

`\stex` can be used to produce either `HTML` or `PDF`. In `HTML`-mode, multiple macros exist to insert annotations. The same macros are also valid in `PDF` mode but implemented as null operations.

---

`\stex_suppress_html:n`

---

`\stex_suppress_html:n`  $\langle code \rangle$

turns annotations off temporarily in  $\langle code \rangle$  (e.g. as to not generate additional annotations for elaborated declarations, or in `sms`-mode).

For that to work, code that inserts annotations should use

---

`\stex_if_do_html_p:` \* tests whether to generate `HTML` annotations.  
`\stex_if_do_html:TF` \*

---

---

`\stex@backend`

---

should be set by a backend engine, such that a file `stex-backend-\stex@backend.cfg` exists.

### 11.14.1 Backend macros

Such a backend config file should provide the following:

---

`\stex_if_html_backend_p:` \* can be used to determine whether the backend produces `HTML` (e.g. `RuSTeX` or `LATEXML`)  
`\stex_if_html_backend:TF` \* or not (e.g. `pdflatex`).

---

`\ifstexhtml` is set accordingly.

---

`\stex_annotate:nnn`

---

`\stex_annotate:nnn`  $\langle attr \rangle \langle value \rangle \langle code \rangle$

In `HTML` mode, annotates the output of  $\langle code \rangle$  with the `XML`-attribute  $\langle attr \rangle = \langle value \rangle$ .  
In `PDF` mode, just calls  $\langle code \rangle$ .

---

`\stex_annotate_invisible:nnn`  
`\stex_annotate_invisible:n`    `\stex_annotate_invisible:n`  $\langle code \rangle$

---

Should annotate  $\langle code \rangle$  with `shtml:visible="false" style="display:none;"`. In PDF mode, does nothing.

`\stex_annotate_invisible:nnn` combines `\stex_annotate_invisible:n` and `\stex_annotate:nnn`.

`stex_annotate_env` (*env.*)    `\begin{stex_annotate_env}{\attr}{\value} \langle code \rangle \end{stex_annotate_env}`  
should behave like `\stex_annotate:nnn{\attr}{\value}{\langle code \rangle}`

---

`\stex_mathml_intent:nn`    MathML Intent (TODO)  
`\stex_mathml_arg:nn`

---

## 11.15 Persisting Content from Math Archives in sms-Files

---

`\stex_persist:n`  
`\stex_persist:e`    `\stex_persist:n`  $\langle code \rangle$

---

writes  $\langle code \rangle$  to the `\jobname.sms`-file, if `writesms` is active.

**T<sub>E</sub>Xhackers note:**  $\langle code \rangle$  is being read with `expl3` category codes (except for spaces having `catcode 10`), but not pretokenized; i.e.  $\langle code \rangle$  can safely change the current `catcode` scheme.

---

`\c_stex_persist_mode_bool`  
`\c_stex_persist_write_mode_bool`

---

whether `usesms` or `writesms` are active.

## 11.16 Utility Methods

---

`\stex_macro_body:N` \* expands to the *expansion* of the provided macro, including parameter tokens, with the original category codes intact; e.g. if `\def\foo#1{First #1}`, then `\stex_macro_body:N \foo` expands to `First #1`.

---

`\stex_macro_definition:N` \* expands to the token list *defining* the provided macro, including parameters, command attributes (i.e. `\long`, `\protected`), with the original category codes intact; e.g. if `\protected\def\foo#1{First #1}`, then `\stex_macro_definition:N \foo` expands to `\protected\def\foo#1{First #1}`.

**T<sub>E</sub>Xhackers note:** Does not work with “higher” parameter tokens, i.e. `##1`, `###1` etc.

---

`\stex_deactivate_macro:Nn`    `\stex_deactivate_macro:Nn`  $\{\langle\macro\rangle\}$   $\{\langle msg\rangle\}$   
`\stex_reactivate_macro:N`     `\stex_reactivate_macro:N`

Makes `\macro` throw an error message `error/deactivated-macro` $\{\langle msg\rangle\}$ , notifying an author that the macro is only allowed in certain environments.  
`\stex_reactivate_macro:N` restores the functionality of the macro.

---

`\stex_kpsewhich:Nn`    `\stex_kpsewhich:Nn`  $\{\langle\macro\rangle\}$   $\{\langle args\rangle\}$

Calls “`kpsewhich`  $\langle args\rangle$ ” and stores the result in `\macro`,

**TeXhackers note:** Does not require `shell-escape`

---

`\stex_get_env:Nn`    `\stex_get_env:Nn`  $\{\langle\macro\rangle\}$   $\{\langle envvar\rangle\}$

Stores the value of the environment variable  $\langle envvar\rangle$  in `\macro`.

---

`\stex_fatal_error:n`    Mimic the `\msg_error:-`macros, but make sure that TeX stops processing.  
`\stex_fatal_error:nmn`  
`\stex_fatal_error:nxx`    **TeXhackers note:** Calls `\input{non-existent file}`.

---

`\stex_ignore_spaces_and_pars:`

As the name suggests, ignores all subsequent spaces and `\pars` until the first non-expandable macro is encountered.

Useful for e.g. ending `\symdecl` and related macros with, so that formatting sources with empty lines does not cause paragraph breaks.

### 11.16.1 Group-like Behaviours

---

`\stex_pseudogroup_with:nm`    `\stex_pseudogroup_with:nm`  $\{\langle macros\rangle\}$  $\{\langle code\rangle\}$

Calls  $\langle code\rangle$  and subsequently restores the values of the  $\langle macros\rangle$  given.

**TeXhackers note:** Does *not* work recursively!

---

`\stex_pseudogroup:nm`    `\stex_pseudogroup:nm`  $\{\langle code1\rangle\}$  $\{\langle code2\rangle\}$

Expands  $\langle code2\rangle$ , and inserts the result after  $\langle code1\rangle$ . Works recursively and allows for restoring the values of macros in combination with `\stex_pseudogroup_restore:N`, but *only for macros that take no arguments*:

---

`\stex_pseudogroup_restore:N *`    `\stex_pseudogroup_restore:N`  $\{\langle\macro\rangle\}$

### Example 34

```
1 \stex_pseudogroup:nn{
2   something changing \foo
3   something changing \num
4 }{
5   \stex_pseudogroup_restore:N\foo
6   \int_set:Nn \num {\int_use:N \num}
7 }
```

restores the values of macro `\foo` and register `\num` after calling the first block.

Commands like `\symdecl` and `\importmodule` that generate (semantic) macros should be local *to the current module*, e.g. `smodule`. For that purpose, we open a new “metagroup” with some identifier (e.g. `\l_stex_current_module_str`) and then execute the relevant code *in the metagroup with that identifier*:

---

```
\stex_metagroup_new:n \stex_metagroup_new:n {<id>}
\stex_metagroup_new:o
```

Opens a new metagroup at the current  $\TeX$  group level with identifier `<id>`.

---

```
\stex_metagroup_do_in:nn \stex_metagroup_do_in:nn {<id>}{<code>}
\stex_metagroup_do_in:nx
```

Executes `<code>` and adds its content to `\aftergroup` up until the  $\TeX$  group level of the metagroup with identifier `<id>`.

## Chapter 12

# Additional Packages

### **12.1** NotesSlides Documentation

TODO

### **12.2** Problem Documentation

TODO

### **12.3** HWExam Documentation

TODO

### **12.4** Tikzinput Documentation

TODO



**Part IV**  
**Implementation**

# Chapter 13

## The sTeX Implementation

### 13.1 Setting up

Setup code for the document class

```
1 <*cls>
2 %%%%%%%%%%% stex.dtx %%%%%%%%%%%
3
4 \RequirePackage{expl3, l3keys2e}
5 \ProvidesExplClass{stex}{2023/10/13}{3.4.0}{sTeX document class}
6 \IfFileExists{stex-expl-compat.sty}{
7   \usepackage{stex-expl-compat}
8 }{}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14
15 \LoadClass{article}
16 </cls>
17
18 Setup code for the package
19
20 <*package>
21 \RequirePackage{expl3, l3keys2e, ltxcmds}
22 \ProvidesExplPackage{stex}{2023/10/13}{3.4.0}{sTeX package}
23 \IfFileExists{stex-expl-compat.sty}{
24   \usepackage{stex-expl-compat}
25 }{}
26 \RequirePackage{stex-logo} % externalized for backwards-compatibility reasons
27 \RequirePackage{standalone}
28
29 \message{^^J*~This-is~sTeX~version~3.4.0~*^^J}
30
31 Package options:
32
33 \keys_define:nn { stex / package } {
34   debug      .str_set_x:N = \c_stex_debug_clist ,
35   lang       .clist_set:N = \c_stex_languages_clist ,
36   mathhub    .tl_set_x:N  = \mathhub ,
```

```

31 usesms .bool_set:N = \c_stex_persist_mode_bool ,
32 writesms .bool_set:N = \c_stex_persist_write_mode_bool ,
33 checkterms .bool_set:N = \c_stex_check_terms_bool ,
34 image .bool_set:N = \c_tikzinput_image_bool,
35 nofrontmatter .bool_set:N = \c_stex_no_frontmatter_bool,
36 unknown .code:n = {}
37 }
38 \exp_args:NNo \clist_set:Nn \c_stex_debug_clist \c_stex_debug_clist
39 \ProcessKeysOptions { stex / package }

Error messages:
40 \input{stex-en.ldf}

```

## 13.2 Utilities

```
41 \cs_set_eq:NN \stex_undefined: \undefined
```

### 13.2.1 Calling kpsewhich and Environment Variables

```
42 <@@=stex_envs>
```

`\stex_kpsewhich:Nn`

```

43 \cs_new_protected:Nn \stex_kpsewhich:Nn {\group_begin:
44   \catcode'\ =12
45   \sys_get_shell:nnN { kpsewhich ~ #2 } { } \l_tmpa_tl
46   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
47   \group_end:
48   \exp_args:NNo\str_set:Nn #1 {\l_tmpa_tl}
49   \tl_trim_spaces:N #1
50 }

```

*(End of definition for \stex\_kpsewhich:Nn. This function is documented on page 142.)*

`\stex_get_env:Nn`

```

51 \sys_if_platform_windows:TF{
52   \cs_new_protected:Nn \stex_get_env:Nn {\group_begin:
53     \escapechar=-1\catcode'\ =12
54     \exp_args:NNe \stex_kpsewhich:Nn #1 {-expand-var~\c_percent_str#2\c_percent_str}
55     \exp_args:NNx\use:nn\group_end:{
56       \str_set:Nn \exp_not:N #1 { #1 }
57     }
58   }
59 }{
60   \cs_new_protected:Nn \stex_get_env:Nn {
61     \stex_kpsewhich:Nn #1 {-var-value~#2}
62   }
63 }

```

*(End of definition for \stex\_get\_env:Nn. This function is documented on page 142.)*

## 13.2.2 Logging

```

64 <@=stex_debug>
\stex_debug:nn
65 \cs_new_protected:Nn \stex_debug:nn {
66   \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist { \tl_to_str:n{all} }{
67     \__stex_debug_:nn{#1}{#2}
68   }{
69     \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist { \tl_to_str:n{#1} }{
70       \__stex_debug_:nn{#1}{#2}
71     }
72   }
73 }
74
75 \cs_new_protected:Nn \__stex_debug_:nn {
76   \msg_set:nnn{stex}{debug / #1}{
77     \\Debug~#1:~#2\\
78   }
79   \msg_none:nn{stex}{debug / #1}
80 }

```

(End of definition for `\stex_debug:nn`. This function is documented on page 120.)

```

\stex_fatal_error:n
\stex_fatal_error:nnn
\stex_fatal_error:nxx
To avoid dead locks etc., we throw errors and make tex stop entirely:
81 \cs_new_protected:Nn \stex_fatal_error:n {
82   \msg_error:nn{stex}{#1}\input{Fatal-Error!!}
83 }
84 \cs_new_protected:Nn \stex_fatal_error:nnn {
85   \msg_error:nnn{stex}{#1}{#2}{#3}\input{Fatal-Error!!}
86 }
87 \cs_generate_variant:Nn \stex_fatal_error:nnn {nxx}

```

(End of definition for `\stex_fatal_error:n` and `\stex_fatal_error:nnn`. These functions are documented on page 142.)

We check an environment variable for debugging and set things up:

```

88 \stex_get_env:Nn\__stex_debug_env_str{STEX_DEBUG}
89 \str_if_empty:NTF\__stex_debug_env_str {
90   \clist_set_eq:NN \l__stex_debug_cl \c_stex_debug_clist
91 }{
92   \clist_set:No \l__stex_debug_cl {\__stex_debug_env_str}
93 }
94 \clist_clear:N \c_stex_debug_clist
95 \clist_map_inline:Nn \l__stex_debug_cl {
96   \exp_args:NNo \clist_put_right:Nn \c_stex_debug_clist
97   { \tl_to_str:n{#1} }
98 }
99
100 \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist {\tl_to_str:n{all}} {
101   \msg_redirect_module:nnn{ stex }{ none }{ warning }
102   \stex_debug:nn{all}{Logging-everything!}
103 }{
104   \clist_map_inline:Nn \c_stex_debug_clist {
105     \msg_redirect_name:nnn{ stex }{ debug / #1 }{ warning }
106     \stex_debug:nn{#1}{Logging~#1}

```

```

107 }
108 }

```

### 13.2.3 Languages

```

109 <@@=stex_lang>

```

```

\c_stex_languages_prop
\c_stex_language_abbrevs_prop

```

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

```

110 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
111   en = english ,
112   de = ngerman ,
113   ar = arabic ,
114   bg = bulgarian ,
115   ru = russian ,
116   fi = finnish ,
117   ro = romanian ,
118   tr = turkish ,
119   fr = french
120 }}
121
122 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
123   english   = en ,
124   ngerman   = de ,
125   arabic    = ar ,
126   bulgarian = bg ,
127   russian   = ru ,
128   finnish   = fi ,
129   romanian  = ro ,
130   turkish   = tr ,
131   french    = fr
132 }}
133 % todo: chinese simplified (zhs)
134 %       chinese traditional (zht)

```

(End of definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 139.)

```

\l_stex_current_language_str

```

```

135 \str_new:N \l_stex_current_language_str

```

(End of definition for `\l_stex_current_language_str`. This variable is documented on page 139.)

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

```

\stex_set_language:n
\stex_set_language:x
\stex_set_language:o

```

```

136 \cs_new_protected:Nn \stex_set_language:n {
137   \str_set:Nn \l_stex_current_language_str { #1 }
138   \prop_if_in:NnTF \c_stex_languages_prop {#1} {
139     \tl_set_rescan:Nnx \l__stex_lang_lang_str {}{\prop_item:Nn \c_stex_languages_prop {#1}}
140     \cs_if_eq:NNTF\@onlypreamble\@notprerr{
141       \ltx@ifpackageloaded{babel}{
142         \exp_args:No\selectlanguage\l__stex_lang_lang_str
143       }{}
144     }{
145       \ltx@ifpackageloaded{babel}{}{
146         \str_if_eq:nnTF {#1} {tr} {

```

```

147         \RequirePackage[turkish,shorthands=!]{babel}
148     }{
149         \RequirePackage[\l__stex_lang_lang_str]{babel}
150     }
151 }
152 }
153 }{
154     \msg_error:nnx{stex}{error/unknownlanguage}{#1}
155 }
156 }
157 \cs_generate_variant:Nn \stex_set_language:n {x,o}

```

(End of definition for `\stex_set_language:n`. This function is documented on page 140.)

`\stex_language_from_file:`

```

158 \cs_new_protected:Nn \stex_language_from_file: {
159     \seq_get_right:NN \g_stex_current_file \l_tmpa_str
160     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
161     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % = ".tex/.dtx/.sty"
162     \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
163         \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
164             \exp_args:No \str_if_eq:nnF \l_tmpa_str {ltx} {
165                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
166             }
167         }
168     }
169     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
170     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
171         \seq_pop_right:NN \l_tmpa_seq \l__stex_lang_str
172         \str_if_eq:NNF \l__stex_lang_str \l_stex_current_language_str {
173             \exp_args:NNo \prop_if_in:NnT \c_stex_languages_prop \l__stex_lang_str {
174                 \stex_set_language:o \l__stex_lang_str
175             }
176         }
177         \stex_debug:nn{lang} {Language~\l_stex_current_language_str~
178             inferred~from~file~name}
179     }
180 }

```

(End of definition for `\stex_language_from_file:`. This function is documented on page 140.)

Loading babel:

```

181 \clist_if_empty:NF \c_stex_languages_clist {
182     \bool_set_false:N \l__stex_lang_turkish_bool
183     \seq_clear:N \l_tmpa_seq
184     \clist_map_inline:Nn \c_stex_languages_clist {
185         \str_set:Nx \l_tmpa_str {#1}
186         \str_if_eq:nnT {#1}{tr}{
187             \bool_set_true:N \l__stex_lang_turkish_bool
188         }
189         \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
190             \tl_set_rescan:Nno \l_tmpa_str {} \l_tmpa_str
191             \seq_put_right:No \l_tmpa_seq \l_tmpa_str
192         } {
193             \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}

```

```

194   }
195   }
196   \stex_debug:nn{lang} {Languages:~\seq_use:Nn \l_tmpa_seq {,~} }
197   \bool_if:NTF \l__stex_lang_turkish_bool {
198     \exp_args:NNe \use:nn \RequirePackage
199     {[main=\seq_use:Nn \l_tmpa_seq, ,shorthands=!]}{babel}
200   }{
201     \exp_args:NNe \use:nn \RequirePackage
202     {[main=\seq_use:Nn \l_tmpa_seq, ]}{babel}
203   }
204 }

```

### 13.2.4 Group-like Behaviours

```

205 <@@=stex_groups>

```

```

\stex_pseudogroup:nn
\stex_pseudogroup_restore:N

```

```

206 \cs_new_protected:Npn \stex_pseudogroup:nn {
207   \exp_args:Nne \use:nn
208 }
209 \cs_new:Nn \stex_pseudogroup_restore:N {
210   \tl_if_exist:NTF #1 {
211     \tl_set:Nn \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
212   }{
213     \cs_undefine:N \exp_not:N #1
214   }
215 }

```

(End of definition for `\stex_pseudogroup:nn` and `\stex_pseudogroup_restore:N`. These functions are documented on page 142.)

```

\stex_pseudogroup_with:nn

```

```

216 \cs_new_protected:Nn \stex_pseudogroup_with:nn {
217   \tl_map_inline:nn{#1}{
218     \cs_set_eq:cN{__stex_groups_\tl_to_str:n{##1}}##1
219   }
220   #2
221   \tl_map_inline:nn{#1}{
222     \cs_set_eq:Nc##1{__stex_groups_\tl_to_str:n{##1}}
223     \cs_undefine:c{__stex_groups_\tl_to_str:n{##1}}
224   }
225 }

```

(End of definition for `\stex_pseudogroup_with:nn`. This function is documented on page 142.)

```

\stex_metagroup_new:n List of all currently existing metagroup identifiers

```

```

\stex_metagroup_new:o

```

```

226 \seq_new:N \l__stex_groups_ids_seq
start a new metagroup at the current group level with id #1
227 \cs_new_protected:Nn \stex_metagroup_new:n {
228   \str_set:cx{l__stex_groups_#1_int} {\int_use:N\currentgrouplevel}
229   \seq_put_right:Nn \l__stex_groups_ids_seq {#1}
230 }
231 \cs_generate_variant:Nn \stex_metagroup_new:n {o}

```

(End of definition for `\stex_metagroup_new:n`. This function is documented on page 143.)

```

\stex_metagroup_do_in:nn
\stex_metagroup_do_in:nx
232 \prg_new_conditional:Nnn \__stex_groups_exists:n {TF} {
233   \str_if_exist:cTF{l__stex_groups_#1_int}
234   \prg_return_true: \prg_return_false:
235 }
236
237 \cs_new_protected:Nn \stex_metagroup_do_in:nn {
238   \__stex_groups_exists:nTF{#1}{
239     \__stex_groups_do_in:nn{#1}{#2}
240   }{
241     \msg_error:nnn{stex}{error/metagroup/missing}{#1}
242   }
243 }
244 \cs_generate_variant:Nn \stex_metagroup_do_in:nn {nx}
245
246 \cs_new_protected:Nn \__stex_groups_do_in:nn {
247   \exp_args:Nnx\stex_debug:nn{metagroup}{adding-to~\detokenize{#1}:^^J\tl_to_str:n{#2}}
248   \tl_set:Nn\__stex_groups_tmp{#2}
249   \exp_args:Nx \int_compare:nNnF {\use:c{l__stex_groups_#1_int}}
250     = \currentgrouplevel {
251     \tl_if_exist:cTF{g__stex_groups_#1\_the\currentgrouplevel_content}{
252       \exp_args:Nno \tl_gput_right:cn{g__stex_groups_#1\_the\currentgrouplevel_content}
253     }{
254       \exp_args:Nno \tl_gset:cn{g__stex_groups_#1\_the\currentgrouplevel_content}
255     }\__stex_groups_tmp
256     \bool_if_exist:cF {l__stex_groups\_the\currentgrouplevel_bool} {
257       \group_insert_after:N \__stex_groups_do:
258       \bool_set_true:c {l__stex_groups\_the\currentgrouplevel_bool}
259     }
260   }
261   \__stex_groups_tmp
262 }
263
264 \cs_new_protected:Nn \__stex_groups_do: {
265   \seq_map_inline:Nn \l__stex_groups_ids_seq {
266     \tl_if_exist:cT{g__stex_groups_##1\_int_eval:n{\currentgrouplevel+1}_content}{
267       \exp_args:NNno \exp_args:Nno \__stex_groups_do_in:nn{##1}{
268         \csname g__stex_groups_##1\_int_eval:n{\currentgrouplevel+1}_content\endcsname
269       }
270       \cs_undefine:c{g__stex_groups_##1\_int_eval:n{\currentgrouplevel+1}_content}
271     }
272     \bool_if_exist:cF {l__stex_groups\_int_eval:n\currentgrouplevel_bool} {
273       \group_insert_after:N \__stex_groups_do:
274       \bool_set_true:c {l__stex_groups\_int_eval:n\currentgrouplevel_bool}
275     }
276   }
277 }

```

(End of definition for `\stex_metagroup_do_in:nn`. This function is documented on page 143.)

### 13.2.5 HTML Annotations

```
278 <@@=stex_annotate>
```



```

\stex_if_do_html:TF Whether to (locally) produce HTML output
279 \bool_new:N \_stex_html_do_output_bool
280 \bool_set_true:N \_stex_html_do_output_bool
281
282 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
283   \bool_if:nTF \_stex_html_do_output_bool
284     \prg_return_true: \prg_return_false:
285 }

```

(End of definition for `\stex_if_do_html:TF`. This function is documented on page 140.)

```

\stex_suppress_html:n Temporarily disable HTML output
286 \cs_new_protected:Nn \stex_suppress_html:n {
287   \stex_pseudogroup:nn{
288     \bool_set_false:N \_stex_html_do_output_bool
289     #1
290   }{
291     \stex_if_do_html:T {
292       \bool_set_true:N \_stex_html_do_output_bool
293     }
294   }
295 }

```

(End of definition for `\stex_suppress_html:n`. This function is documented on page 140.)

We determine the backend:

```

\stex_if_html_backend_p:
\stex_if_html_backend:TF
  \ifstexhtml
  \stex@backend
296 \ifcsname if@rustex\endcsname\else
297   \expandafter\newif\csname if@rustex\endcsname
298   \@rustexfalse
299 \fi
300
301 \stex_get_env:Nn\__stex_annotate_env_str{STEX_FORCE_PDF}
302 \exp_args:No \str_if_eq:nnTF \__stex_annotate_env_str {true} {
303   \def\stex@backend{pdflatex}
304 }{
305   \tl_if_exist:NF\stex@backend{
306     \if@rustex
307       \def\stex@backend{rustex}
308     \else
309       \cs_if_exist:NTF\HCode{
310         \def\stex@backend{tex4ht}
311       }{
312         \def\stex@backend{pdflatex}
313       }
314     \fi
315   }
316 }
317
318 \input{stex-backend-\stex@backend.cfg}
319
320 \newif\ifstexhtml
321 \stex_if_html_backend:TF {
322   \stexhtmltrue

```

```

323 \bool_set_true:N \_stex_html_do_output_bool
324 }{
325 \stexhtmlfalse
326 \bool_set_false:N \_stex_html_do_output_bool
327 }

```

(End of definition for `\stex_if_html_backend:TF`, `\ifstexhtml`, and `\stex@backend`. These functions are documented on page 140.)

`\_stex_annotate_force_break:n`

```

328 \stex_if_html_backend:TF {
329 \cs_new_protected:Nn \_stex_annotate_force_break:n {
330 \stex_annotate_invisible:n{~}
331 #1
332 \stex_annotate_invisible:n{~}
333 }
334 }{
335 \cs_new_protected:Nn \_stex_annotate_force_break:n { #1 }
336 }

```

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

`\mmlintnt`

`\mmlarg`

```

337 \stex_if_html_backend:TF {
338 \cs_new_protected:Npn \mmlintnt #1 #2 {
339 \stex_annotate:nn{mml:intent={#1}}{#2}
340 }
341 \cs_new_protected:Npn \mmlarg #1 #2 {
342 \stex_annotate:nn{mml:arg={#1}}{#2}
343 }
344 }{
345 \cs_new_protected:Npn \mmlintnt #1 #2 { #2 }
346 \cs_new_protected:Npn \mmlarg #1 #2 { #2 }
347 }

```

(End of definition for `\mmlintnt` and `\mmlarg`. These functions are documented on page ??.)

## 13.2.6 Auxiliary Methods

348 `<@=stex_aux>`

`\stex_deactivate_macro:Nn`

`\stex_reactivate_macro:N`

```

349 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
350 \tl_set_eq:cN{\tl_to_str:n{#1}---orig}#1
351 \cs_set_protected:Npn#1{
352 \exp_args:Nno\msg_error:nmmn{stex}{error/deactivated-macro}{#1}{#2}
353 }
354 }
355 \cs_new_protected:Nn \stex_reactivate_macro:N {
356 \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1}---orig\endcsname
357 }

```

(End of definition for `\stex_deactivate_macro:Nn` and `\stex_reactivate_macro:N`. These functions are documented on page 142.)

`\stex_ignore_spaces_and_pars:`

```
358 \protected\def\stex_ignore_spaces_and_pars:{
359   \begingroup\catcode13=10\relax
360   \@ifnextchar\par{
361     \endgroup\expandafter\stex_ignore_spaces_and_pars:\@gobble
362   }{
363     \endgroup
364   }
365 }
```

(End of definition for `\stex_ignore_spaces_and_pars:`. This function is documented on page 142.)

`\stex_keys_define:nmn`

```
366 \cs_new_nopar:Nn \stex_keys_define:nmn {
367   \tl_gset:cn {__stex_aux_keys_#1_pre_tl}{#2}
368   \tl_gset:cn {__stex_aux_keys_#1_def_tl}{#3}
369   \tl_if_empty:nF{#4}{
370     \clist_map_inline:nn{#4}{
371       \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_pre_tl}
372       \tl_gput_left:co{__stex_aux_keys_#1_pre_tl} \l__stex_aux_tl
373       \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_def_tl}
374       \tl_gput_left:cn{__stex_aux_keys_#1_def_tl} ,
375       \tl_gput_left:co{__stex_aux_keys_#1_def_tl} \l__stex_aux_tl
376     }
377   }
378   \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_#1_def_tl}
379   \exp_args:Nno \keys_define:nn {stex / #1} {\l__stex_aux_tl}
380 }
```

(End of definition for `\stex_keys_define:nmn`. This function is documented on page 132.)

`\stex_keys_set:nn`

```
381 \cs_new_nopar:Nn \stex_keys_set:nn {
382   \use:c{__stex_aux_keys_#1_pre_tl}
383   \keys_set:nn {stex / #1} { #2 }
384 }
```

(End of definition for `\stex_keys_set:nn`. This function is documented on page 132.)

Some ubiquitous key sets:

```
385 \stex_keys_define:nmn{archive file}{
386   \str_clear:N \l_stex_key_archive_str
387   \str_clear:N \l_stex_key_file_str
388 }{
389   archive .str_set_x:N = \l_stex_key_archive_str ,
390   file .str_set_x:N = \l_stex_key_file_str
391 }{}
392
393 \stex_keys_define:nmn{id}{
394   \str_clear:N \l_stex_key_id_str
395 }{
396   id .str_set_x:N = \l_stex_key_id_str
397 }{}
398
399 \stex_keys_define:nmn{title}{
```

```

400 \tl_clear:N \l_stex_key_title_tl
401 }{
402   title .tl_set:N = \l_stex_key_title_tl
403 }{}
404
405 \stex_keys_define:nmmn{style}{
406   \clist_clear:N \l_stex_key_style_clist
407 }{
408   style .clist_set:N = \l_stex_key_style_clist
409 }{}
410
411 \stex_keys_define:nmmn{deprecate}{
412   \str_clear:N \l_stex_key_deprecate_str
413 }{
414   deprecate .str_set_x:N = \l_stex_key_deprecate_str
415 }{}
416
417 \stex_keys_define:nmmn{uses}{}{
418   uses .code:n = {
419     \clist_map_inline:nn{##1}{
420       \stex_str_if_starts_with:nnTF{##1}[{
421         \__stex_aux_split_at_bracket:w ##1 \_stex_end:
422       }{
423         \usemodule{##1}
424       }
425     }
426   }
427 }{}
428
429 \cs_new_protected:Npn \__stex_aux_split_at_bracket:w [ #1 ] #2 \_stex_end: {
430   \usemodule[#1]{#2}
431 }

```

**\\_stex\_do\_deprecation:n**

```

432 \cs_new:Nn \_stex_do_deprecation:n {
433   \str_if_empty:NF \l_stex_key_deprecate_str {
434     \msg_warning:nnxx{stex}{warning/deprecated}{#1}{\l_stex_key_deprecate_str}
435   }
436 }

```

*(End of definition for \\_stex\_do\_deprecation:n. This function is documented on page 120.)*

**\\_stex\_do\_id:**

```

437 \cs_new_protected:Nn \_stex_do_id: {
438   \stex_if_smsmode:F {
439     \str_if_empty:NF \l_stex_key_id_str {
440       \exp_args:No \stex_ref_new_doc_target:n \l_stex_key_id_str
441     }
442   }
443 }

```

*(End of definition for \\_stex\_do\_id:. This function is documented on page 132.)*

```

\stex_new_stylable_env:nnnnnnn
\stex_new_stylable_cmd:nnnn
\stex_style_apply:
444 \cs_new_protected:Nn \stex_new_stylable_cmd:nnnn {
445   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[2] []{
446     \__stex_aux_patch:nnn{#1}{##1}{##2}
447   }
448   \exp_after:wN \NewDocumentCommand\cs:w #1\cs_end:{#2}{
449     \cs_set:Npn \stex_style_apply: {
450       \__stex_aux_apply_patch:n{#1}
451     }
452     #3
453   }
454   \tl_set:cn {__stex_aux_style_#1:} { #4 }
455 }
456
457 \cs_new_protected:Nn \__stex_aux_apply_patch:n {
458   \clist_if_empty:NTF \l_stex_key_style_clist {
459     \tl_clear:N \thisstyle
460     \use:c{__stex_aux_style_#1:}
461   }{
462     \clist_get:NN \l_stex_key_style_clist \thisstyle
463     \tl_if_exist:cTF{__stex_aux_style_#1\_thisstyle :}{
464       \use:c{__stex_aux_style_#1\_thisstyle :}
465     }{
466       \use:c{__stex_aux_style_#1:}
467     }
468   }
469 }
470
471 \cs_new_protected:Nn \__stex_aux_patch:nnn {
472   \str_if_empty:nTF {#2}{
473     \tl_set:cn{__stex_aux_style_#1:}{#3}
474   }{
475     \tl_set:cn{__stex_aux_style_#1_#2:}{#3}
476   }
477 }
478
479 \cs_new_protected:Nn \stex_new_stylable_env:nnnnnnn {
480   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[3] []{
481     \__stex_aux_patch:nnnn{#1}{##1}{##2}{##3}
482   }
483   \NewDocumentEnvironment{#7#1}{#2}{
484     \cs_set:Npn \stex_style_apply: {
485       \__stex_aux_apply_patch_begin:n{#1}
486     }
487     #3
488   }{
489     \cs_set:Npn \stex_style_apply: {
490       \__stex_aux_apply_patch_end:n{#1}
491     }
492     #4
493   }
494   \tl_set:cn {__stex_aux_style_#1_start:} { #5 }
495   \tl_set:cn {__stex_aux_style_#1_end:} { #6 }
496 }

```

```

497
498 \cs_new_protected:Nn \__stex_aux_patch:nmmm {
499   \str_if_empty:nTF {#2}{
500     \tl_set:cn{__stex_aux_style_#1_start:}{#3}
501     \tl_set:cn{__stex_aux_style_#1_end:}{#4}
502   }{
503     \tl_set:cn{__stex_aux_style_#1_#2_start:}{#3}
504     \tl_set:cn{__stex_aux_style_#1_#2_end:}{#4}
505   }
506 }
507
508 \cs_new_protected:Nn \__stex_aux_apply_patch_begin:n {
509   \clist_if_empty:NTF \l_stex_key_style_clist {
510     \tl_clear:N \thisstyle
511     \use:c{__stex_aux_style_#1_start:}
512   }{
513     \clist_get:NN \l_stex_key_style_clist \thisstyle
514     \stex_debug:nn{styling}{dominant~style::~\thisstyle}
515     \tl_if_exist:cTF{__stex_aux_style_#1\thisstyle_start:}{
516       \use:c{__stex_aux_style_#1\thisstyle_start:}
517     }{
518       \use:c{__stex_aux_style_#1_start:}
519     }
520   }
521 }
522
523 \cs_new_protected:Nn \__stex_aux_apply_patch_end:n {
524   \tl_if_empty:NTF \thisstyle {
525     \use:c{__stex_aux_style_#1_end:}
526   }{
527     \tl_if_exist:cTF{__stex_aux_style_#1\thisstyle_end:}{
528       \use:c{__stex_aux_style_#1\thisstyle_end:}
529     }{
530       \use:c{__stex_aux_style_#1_end:}
531     }
532   }
533 }

```

(End of definition for `\stex_new_stylable_env:nmmmmmm`, `\stex_new_stylable_cmd:nmmmm`, and `\stex_style_apply:.` These functions are documented on page 133.)

`\stex_str_if_ends_with_p:nn`

`\stex_str_if_ends_with:nnTF`

```

534 \prg_new_conditional:Nnn \stex_str_if_ends_with:nn {p,T,F,TF} {
535   \exp_args:Ne \str_if_eq:nnTF {
536     \str_range:nnn{#1}{- \str_count:n{#2}}{-1}
537   }{#2}\prg_return_true: \prg_return_false:
538 }

```

(End of definition for `\stex_str_if_ends_with:nnTF`. This function is documented on page 136.)

`\stex_str_if_starts_with_p:nn`

`\stex_str_if_starts_with:nnTF`

```

539 \prg_new_conditional:Nnn \stex_str_if_starts_with:nn {p,T,F,TF} {
540   \exp_args:Ne \str_if_eq:nnTF {
541     \str_range:nnn{#1}{1}{\str_count:n{#2}}
542   }{#2}\prg_return_true: \prg_return_false:

```

```
543 }
```

(End of definition for `\stex_str_if_starts_with:nnTF`. This function is documented on page 136.)

`\stex_macro_body:N`

```
544 \cs_new:Npn \__stex_aux_start:#1\__stex_aux_end: {\exp_not:n{#1}}
545 \cs_new_protected:Nn \__stex_aux_end: {}
546 \cs_new:Nn \stex_macro_body:N {
547   \exp_args:Nne\use:nn{\exp_after:wN \__stex_aux_start: #1}{
548     \__stex_aux_args:e {\cs_parameter_spec:N #1}\__stex_aux_end:
549   }
550 }
551
552 \cs_new:Nn \__stex_aux_args:n {
553   \tl_if_empty:nF{#1}{
554     {##\exp_args:Ne \tl_head:n {\tl_tail:n {#1}}}
555     \__stex_aux_args:e {\exp_args:Ne\tl_tail:n{\tl_tail:n{#1}}}
556   }
557 }
558 \cs_generate_variant:Nn \__stex_aux_args:n {e}
```

(End of definition for `\stex_macro_body:N`. This function is documented on page 141.)

`\stex_macro_definition:N`

```
559 \cs_new:Nn \stex_macro_definition:N {
560   \__stex_aux_prefix:e {\cs_prefix_spec:N #1}
561   \def\exp_not:N #1
562   \__stex_aux_params:e {\cs_parameter_spec:N #1}
563   {
564     \stex_macro_body:N #1
565   }
566 }
567
568 \cs_new:Nn \__stex_aux_prefix:n {
569   \tl_if_empty:nF{#1}{
570     \str_if_eq:eeTF {
571       \tl_range:nnn{#1}{1}{10}~
572     }{\tl_to_str:n{\protected}}{\
573       \protected
574       \__stex_aux_prefix_long:e {
575         \str_range:nnn{#1}{11}{-1}
576       }
577     }{
578       \__stex_aux_prefix_long:n {#1}
579     }
580   }
581 }
582 \cs_generate_variant:Nn \__stex_aux_prefix:n {e}
583
584 \cs_new:Nn \__stex_aux_prefix_long:n {
585   \tl_if_empty:nF{#1}{
586     \str_if_eq:eeT {
587       \tl_range:nnn{#1}{1}{10}~
588     }{\tl_to_str:n{\long}}{\long}
589   }
590 }
```

```

590 }
591 \cs_generate_variant:Nn \__stex_aux_prefix_long:n {e}
592
593 \cs_new:Nn \__stex_aux_params:n {
594   \tl_if_empty:nF{#1}{
595     \exp_args:NNe \str_if_eq:VnTF \c_hash_str {\tl_head:n{#1}}{
596       #####
597     }{
598       \tl_head:n{#1}
599     }
600     \__stex_aux_params:e {\tl_tail:n{#1}}
601   }
602 }
603 \cs_generate_variant:Nn \__stex_aux_params:n {e}

```

(End of definition for `\stex_macro_definition:N`. This function is documented on page 141.)

### 13.2.7 Persistence

```

604 <@@=stex_persist>

```

We check the environment variables:

```

605 \stex_get_env:Nn\__stex_persist_env_str{STEX_USESMS}
606 \str_if_empty:NF\__stex_persist_env_str{
607   \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false}{
608     \bool_set_true:N \c_stex_persist_mode_bool
609   }
610 }
611 \stex_get_env:Nn\__stex_persist_env_str{STEX_WRITESMS}
612 \str_if_empty:NF\__stex_persist_env_str{
613   \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false}{
614     \bool_set_true:N \c_stex_persist_write_mode_bool
615   }
616 }

```

`\stex_persist:n`

`\stex_persist:e`

```

617 \iow_new:N \c__stex_persist_sms_iow
618
619 \bool_if:NTF \c_stex_persist_write_mode_bool {
620   \stex_if_html_backend:TF{
621     \cs_new:Npn \stex_persist:n #1 {}
622     \cs_new:Npn \stex_persist:e #1 {}
623   }{
624     \cs_new_protected:Nn \stex_persist:n {
625       \iow_now:Nn \c__stex_persist_sms_iow {#1}
626     }
627     \cs_generate_variant:Nn \stex_persist:n {e}
628   }
629 }{
630   \cs_new:Npn \stex_persist:n #1 {}
631   \cs_new:Npn \stex_persist:e #1 {}
632 }

```

(End of definition for `\stex_persist:n`. This function is documented on page 141.)

Is called at the end of the `.sty`-file:



```

633
634 \cs_new_protected:Nn \__stex_persist_load_file:n {
635   \file_if_exist:nT{#1}{
636     \group_begin:
637     \cs_set:Npn \stex_persist:n ##1 {}
638     \cs_set:Npn \stex_persist:e ##1 {}
639     \stex_debug:nn{persist}{restoring~from~sms~file}
640     \catcode'\ =10\relax
641     \cs:w @ @ input \cs_end:#1\relax
642   \group_end:
643 }
644 }
645
646 \cs_new_protected:Nn \__stex_persist_write_only: {
647   \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
648   \AtEndDocument{ \iow_close:N \c__stex_persist_sms_iow }
649 }
650
651 \cs_new_protected:Nn \__stex_persist_read_and_write: {
652   \file_if_exist:nTF{\jobname.sms}{
653     \ior_open:Nn \g_tmpa_ior {\jobname.sms}
654     \iow_open:Nn \g_tmpa_iow {\jobname.sms2}
655     \ior_str_map_inline:Nn \g_tmpa_ior {
656       \iow_now:Nn \g_tmpa_iow {##1}
657     }
658     \iow_close:N \g_tmpa_ior
659     \ior_close:N \g_tmpa_ior
660     \__stex_persist_write_only:
661     \ior_open:Nn \g_tmpa_ior {\jobname.sms2}
662     \ior_str_map_inline:Nn \g_tmpa_ior {
663       \iow_now:Nn \c__stex_persist_sms_iow {##1}
664     }
665     \ior_close:N \g_tmpa_ior
666     \__stex_persist_load_file:n{\jobname.sms2}
667   } \__stex_persist_write_only:
668 }
669
670 \cs_new_protected:Nn \stex_persist_read_now: {
671   \bool_if:NTF \c_stex_persist_mode_bool {
672     \bool_if:NTF \c_stex_persist_write_mode_bool
673     \__stex_persist_read_and_write:
674     {
675       \__stex_persist_load_file:n{\jobname.sms}
676     }
677   } {
678     \bool_if:NT \c_stex_persist_write_mode_bool \__stex_persist_write_only:
679   }
680 }

```

## 13.2.8 Files, Paths and URIs

```
681 <@@=stex_path>
```

```

\stex_file_set:Nn
\stex_file_set:No
\stex_file_set:Nx

```

```

682 \cs_new_protected:Nn \stex_file_set:Nn {
683   \str_if_empty:nTF {#2} { \seq_clear:N #1 }{
684     \exp_args:NNno \seq_set_split:Nnn #1 / { \tl_to_str:n{#2} }
685   }
686 }
687 \cs_generate_variant:Nn \stex_file_set:Nn {No, Nx}

```

(End of definition for `\stex_file_set:Nn`. This function is documented on page 137.)

`\stex_file_resolve:Nn`

`\stex_file_resolve:No`

`\stex_file_resolve:Nx`

```

688 \sys_if_platform_windows:TF{
689   \cs_new_protected:Npn \__stex_path_win_take:w #1#2#3 \__stex_path_: {
690     \str_set:Nn \l__stex_path_win_drive {#1#2}
691     \str_set:Nn \l__stex_path_str{#3}
692   }
693   \cs_new_protected:Nn \stex_file_resolve:Nn {
694     \str_set:Nn \l__stex_path_str {#2}
695     \str_clear:N \l__stex_path_win_drive
696     \exp_args:NNo \str_replace_all:Nnn \l__stex_path_str \c_backslash_str /
697     \exp_args:Nx \str_if_eq:nnT {\str_item:Nn \l__stex_path_str 2} : {
698       \exp_after:wN \__stex_path_win_take:w \l__stex_path_str \__stex_path_:
699     }
700     \stex_file_set:No #1 \l__stex_path_str
701     \__stex_path_canonicalize:N #1
702     \str_if_empty:NF \l__stex_path_win_drive {
703       \seq_pop_left:NN #1 \l__stex_path_str
704       \seq_put_left:No #1 \l__stex_path_win_drive
705     }
706     %\stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
707   }
708 }{
709   \cs_new_protected:Nn \stex_file_resolve:Nn {
710     \str_set:Nn \l__stex_path_str {#2}
711     \stex_file_set:No #1 \l__stex_path_str
712     \__stex_path_canonicalize:N #1
713     % \stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
714   }
715 }
716 \cs_generate_variant:Nn \stex_file_resolve:Nn {No, Nx}
717
718 \cs_new_protected:Nn \__stex_path_canonicalize:N {
719   \seq_if_empty:NF #1 {
720     \seq_pop:NN #1 \l__stex_path_str
721     \seq_clear:N \l__stex_path_seq
722     \str_if_empty:NTF \l__stex_path_str {
723       \seq_map_function:NN #1 \__stex_path_dodots:n
724       \seq_put_left:Nn \l__stex_path_seq {}
725     }{
726       \seq_push:No #1 \l__stex_path_str
727       \seq_map_function:NN #1 \__stex_path_dodots:n
728     }
729     \seq_set_eq:NN #1 \l__stex_path_seq
730   }
731 }

```

```

732
733 \cs_new_protected:Nn \__stex_path_dodots:n {
734   \str_if_empty:nF{#1}{
735     \str_if_eq:nnF {#1} {.} {
736       \str_if_eq:nnTF {#1} {..} {
737         \seq_if_empty:NF \l__stex_path_seq {
738           \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
739         }
740       }{
741         \seq_put_right:Nn \l__stex_path_seq {#1}
742       }
743     }
744   }
745 }

```

(End of definition for `\stex_file_resolve:Nn`. This function is documented on page 137.)

`\stex_if_file_absolute_p:N`  
`\stex_if_file_absolute:NTF`

```

746
747 \sys_if_platform_windows:TF {
748   \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
749     \seq_if_empty:NTF #1 \prg_return_false: {
750       \tl_set:Nx \l__stex_path_maybewin_str {\seq_item:Nn #1 1}
751       \exp_args:No \tl_if_empty:nTF \l__stex_path_maybewin_str \prg_return_true: {
752         \exp_args:Nx \str_if_eq:nnTF {\str_item:Nn \l__stex_path_maybewin_str 2} :
753         \prg_return_true: \prg_return_false:
754       }
755     }
756   }
757 }{
758   \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
759     \seq_if_empty:NTF #1 \prg_return_false: {
760       \exp_args:Nx \tl_if_empty:nTF {\seq_item:Nn #1 1}
761       \prg_return_true: \prg_return_false:
762     }
763   }
764 }

```

(End of definition for `\stex_if_file_absolute:NTF`. This function is documented on page 137.)

`\stex_file_use:N`

```

765 \cs_new:Nn \stex_file_use:N {
766   \seq_use:Nn #1 /
767 }

```

(End of definition for `\stex_file_use:N`. This function is documented on page 137.)

`\stex_if_file_starts_with:NNTF`

```

768 \prg_new_protected_conditional:Nnn \stex_if_file_starts_with:NN {T,F,TF} {
769   \seq_set_eq:NN \l__stex_path_a_seq #1
770   \seq_set_eq:NN \l__stex_path_b_seq #2
771   \tl_clear:N \l__stex_path_return_tl
772   \bool_while_do:nn{
773     \bool_not_p:n{
774       \bool_lazy_any_p:n{

```

```

775     {\seq_if_empty_p:N \l__stex_path_a_seq}
776     {\seq_if_empty_p:N \l__stex_path_b_seq}
777     {\bool_not_p:n{\tl_if_empty_p:N \l__stex_path_return_tl}}
778   }
779 }
780 }{
781   \seq_pop_left:NN \l__stex_path_a_seq \l__stex_path_a_tl
782   \seq_pop_left:NN \l__stex_path_b_seq \l__stex_path_b_tl
783   \str_if_eq:NNF \l__stex_path_a_tl \l__stex_path_b_tl {
784     \tl_set:Nn \l__stex_path_return_tl {\prg_return_false:}
785   }
786 }
787 \tl_if_empty:NTF \l__stex_path_return_tl {
788   \seq_if_empty:NTF \l__stex_path_b_seq \prg_return_true: \prg_return_false:
789 } \l__stex_path_return_tl
790 }

```

(End of definition for `\stex_if_file_starts_with:NNTF`. This function is documented on page 137.)

`\stex_file_split_off_ext:NN`  
`\stex_file_split_off_lang:NN`

```

791 \cs_new_protected:Nn \stex_file_split_off_ext:NN {
792   \seq_set_eq:NN #1 #2
793   \seq_pop_right:NN #1 \l__stex_path_str
794   \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
795   \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
796   \seq_put_right:Nx #1 {\seq_use:Nn \l__stex_path_seq .}
797 }
798 \cs_new_protected:Nn \stex_file_split_off_lang:NN {
799   \seq_set_eq:NN #1 #2
800   \seq_pop_right:NN #1 \l__stex_path_str
801   \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
802   \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
803
804   \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
805   \exp_args:NNNo \prop_if_in:NnF \c_stex_languages_prop \l__stex_path_str {
806     \seq_put_right:No \l__stex_path_seq \l__stex_path_str
807   }
808
809   \seq_put_right:Nx #1 {\seq_use:Nn \l__stex_path_seq .}
810 }

```

(End of definition for `\stex_file_split_off_ext:NN` and `\stex_file_split_off_lang:NN`. These functions are documented on page 137.)

URIs:

`\stex_map_uri:Nnnnn`

```

811 \cs_set_protected:Nn \__stex_path_auth:n {
812   \msg_error:nnx{stex}{error/misused-uri}{\tl_to_str:n{#1}}
813 }
814 \cs_set_eq:NN \__stex_path_path:n \__stex_path_auth:n
815 \cs_set_eq:NN \__stex_path_module:n \__stex_path_auth:n
816 \cs_set_eq:NN \__stex_path_name:n \__stex_path_auth:n
817
818 \cs_set_protected:Nn \stex_map_uri:Nnnnn{
819   \stex_pseudogroup_with:nn{\__stex_path_auth:n\__stex_path_path:n\__stex_path_module:n\__st

```

```

820 \cs_set:Npn \__stex_path_auth:n ##1 {#2}
821 \cs_set:Npn \__stex_path_path:n ##1 {#3}
822 \cs_set:Npn \__stex_path_module:n ##1 {#4}
823 \cs_set:Npn \__stex_path_name:n ##1 {#5}
824 #1
825 }
826 }

```

(End of definition for `\stex_map_uri:Nnnnn`. This function is documented on page 138.)

`\stex_uri_set:Nn`

`\stex_uri_set:No`

`\stex_uri_set:Nx`

```

827 \str_set:Nx\__stex_path_colonslash{\c_colon_str/}
828 \cs_new_protected:Nn \__stex_path_uri_set:NnN {
829   \str_if_empty:nTF {#2} {
830     \msg_error:nxxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{empty}
831   }{
832     \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn #1 \__stex_path_colonslash { \tl_to_str
833     \seq_pop_left:NN #1 \l__stex_path_auth_str
834     \seq_if_empty:NTF #1 {
835       \msg_error:nxxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{missing~authority}
836     }{
837       \exp_args:NNnx \seq_set_split:Nnn #1 ? {\exp_args:NNo \seq_use:Nn #1 \__stex_path_colo
838       \seq_pop_left:NN #1 \l__stex_path_path
839       #3 \l__stex_path_path \l__stex_path_path
840       \seq_if_empty:NTF #1 {
841         \exp_args:NNo \__stex_path_uri_set:Nnxnn #1 \l__stex_path_auth_str
842         {\stex_file_use:N \l__stex_path_path} {} {}
843       }{
844         \seq_pop_left:NN #1 \l__stex_path_mod
845         \seq_if_empty:NTF #1 {
846           \exp_args:NNo \__stex_path_uri_set:Nnxon #1 \l__stex_path_auth_str
847           {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod {}
848         }{
849           \seq_pop_left:NN #1 \l__stex_path_name
850           \seq_if_empty:NTF #1 {
851             \exp_args:NNo \__stex_path_uri_set:Nnxon #2 \l__stex_path_auth_str
852             {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod \l__stex_path_name
853           }{
854             \msg_error:nxxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{too~many~?s}
855           }
856         }
857       }
858     }
859   }
860   \stex_debug:nn{uris}{Set~\tl_to_str:n{#1}~to~\stex_uri_use:N #1}
861 }
862
863 \cs_new_protected:Nn \__stex_path_uri_set:Nnnnn{
864   \tl_set:Nn #1 {
865     \__stex_path_auth:n{ #2 }
866     \__stex_path_path:n{ #3 }
867     \__stex_path_module:n{ #4 }
868     \__stex_path_name:n{ #5 }
869   }

```

```

870 }
871 \cs_generate_variant:Nn \__stex_path_uri_set:Nnnnn {Nnxnn,Nnxon,Nnxoo}
872
873 \cs_new_protected:Nn \stex_uri_set:Nn {
874   \__stex_path_uri_set:NnN #1 {#2} \stex_file_set:No
875 }
876 \cs_generate_variant:Nn \stex_uri_set:Nn {No, Nx}

```

(End of definition for \stex\_uri\_set:Nn. This function is documented on page 138.)

```

\stex_uri_resolve:Nn
\stex_uri_resolve:No
\stex_uri_resolve:Nx

```

```

877 \cs_new_protected:Nn \stex_uri_resolve:Nn {
878   \__stex_path_uri_set:NnN #1 {#2} \stex_file_resolve:No
879 }
880 \cs_generate_variant:Nn \stex_uri_resolve:Nn {No, Nx}

```

(End of definition for \stex\_uri\_resolve:Nn. This function is documented on page 138.)

```

\stex_uri_use:N

```

```

881 \cs_new:Npn \__stex_path_uri_use:w \__stex_path_auth:n #1 \__stex_path_path:n #2 \__stex_path
882   #1\c_colon_str/ #2 \tl_if_empty:nF { #3 }{ ? #3
883   \tl_if_empty:nF { #4 }{ ? #4 } }
884 }
885 \cs_new:Nn \stex_uri_use:N {
886   \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N #1 } \__stex_path_auth:n {
887     \exp_after:wN \__stex_path_uri_use:w #1
888   }{
889     \msg_error:nnnn{stex}{error/invalid-uri}{#1}{Not~a~URI}
890   }
891 }

```

(End of definition for \stex\_uri\_use:N. This function is documented on page 138.)

```

\stex_uri_from_repo_file:NNNn
\stex_uri_from_repo_file_nolang:NNNn

```

```

892 \cs_new_protected:Npn \stex_uri_from_repo_file_nolang:NNNn {
893   \__stex_path_from_repo_file:NNNNn \stex_file_split_off_lang:NN
894 }
895 \cs_new_protected:Npn \stex_uri_from_repo_file:NNNn {
896   \__stex_path_from_repo_file:NNNNn \stex_file_split_off_ext:NN
897 }
898
899 \cs_new_protected:Nn \__stex_path_from_repo_file:NNNNn {
900   #1 \l__stex_path_file #4
901   \prop_if_exist:NTF #3 {
902     \str_clear:N \l__stex_path_uri
903     \prop_get:NnNF #3 {#5} \l__stex_path_uri {
904       \prop_get:NnNF #3 {ns} \l__stex_path_uri {
905         \__stex_path_uri_set:Nnxnn #2 {file}
906         {\stex_file_use:N \l__stex_path_file} {} {}
907       }
908     }
909     \str_if_empty:NF \l__stex_path_uri {\__stex_path_relativize:N #2}
910   }{
911     \exp_args:NNx \__stex_path_uri_set:Nnxnn #2 {\tl_to_str:n{file}}
912     {\stex_file_use:N \l__stex_path_file} {} {}

```

```

913 }
914 }
915
916 \cs_new_protected:Nn \__stex_path_relativize:N {
917   \seq_set_eq:NN \l__stex_path_seq \l__stex_path_file
918   \seq_map_inline:Nn \c_stex_mathhub_file { % mathhub path
919     \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
920   }
921   \stex_file_set:Nx \l__stex_path_path {\prop_item:Nn \l_stex_current_archive_prop {id} }
922   \seq_map_inline:Nn \l__stex_path_path { % id
923     \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
924   }
925   \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl % source
926
927   \stex_uri_set:Nx #1 { \l__stex_path_uri / \stex_file_use:N \l__stex_path_seq }
928 }

```

(End of definition for `\stex_uri_from_repo_file:NNNn` and `\stex_uri_from_repo_file_nolang:NNNn`.  
These functions are documented on page 138.)

```

\stex_uri_from_current_file:Nn
\stex_uri_from_current_file_nolang:Nn
929 \cs_new_protected:Nn \stex_uri_from_current_file:Nn {
930   \stex_debug:nn{docuri}{Here:~\stex_file_use:N \g_stex_current_file}
931   \stex_uri_from_repo_file:NNNn #1 \l_stex_current_archive_prop
932     \g_stex_current_file {#2}
933   \stex_debug:nn{docuri}{resolved:~\stex_uri_use:N #1}
934 }
935 \cs_new_protected:Nn \stex_uri_from_current_file_nolang:Nn {
936   \stex_uri_from_repo_file_nolang:NNNn #1 \l_stex_current_archive_prop
937     \g_stex_current_file {#2}
938 }

```

(End of definition for `\stex_uri_from_current_file:Nn` and `\stex_uri_from_current_file_nolang:Nn`.  
These functions are documented on page 139.)

```

\stex_uri_add_module:NNn
\stex_uri_add_module:NNo
939 \cs_new_protected:Nn \stex_uri_add_module:NNn {
940   \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N #2 } \__stex_path_auth:n {
941     \stex_pseudogroup_with:nn
942       {\__stex_path_auth:n\__stex_path_path:n\__stex_path_module:n\__stex_path_name:n}
943     {
944       \cs_set:Npn \__stex_path_module:n ##1 {
945         \tl_if_empty:nTF{##1}{
946           \exp_not:N \__stex_path_module:n {#3}
947         }{
948           \msg_error:nnn{stex}{error/invalid-dpath}{#2}
949         }
950       }
951       \cs_set:Npn \__stex_path_name:n ##1 {
952         \tl_if_empty:nTF{##1}{
953           \exp_not:N \__stex_path_name:n {}
954         }{
955           \msg_error:nnn{stex}{error/invalid-dpath}{#2}
956         }
957       }
958     }
959 }

```

```

958     \tl_set:Nx #1 {#2}
959   }
960 }{
961   \msg_error:nnnn{stex}{error/invalid-uri}{#2}{Not~a~URI}
962 }
963 }
964 \cs_generate_variant:Nn \stex_uri_add_module:NNn {NNn}

```

(End of definition for `\stex_uri_add_module:NNn`. This function is documented on page 139.)

`\l_stex_current_doc_uri`

```

965 \tl_new:N \l_stex_current_doc_uri

```

(End of definition for `\l_stex_current_doc_uri`. This variable is documented on page 139.)

`\stex_set_document_uri:`

```

966 \cs_new_protected:Nn \stex_set_document_uri: {
967   \stex_uri_from_current_file:Nn \l_stex_current_doc_uri {narr}
968   %\stex_debug:nn{sref}{Document~URI:~\stex_uri_use:N \l_stex_current_doc_uri}
969 }

```

(End of definition for `\stex_set_document_uri:`. This function is documented on page 139.)

`\stex_set_current_namespace:`

```

970 \cs_new_protected:Nn \stex_set_current_namespace: {
971   \stex_uri_from_current_file_nolang:Nn \l_stex_current_ns_uri {source-base}
972   %\stex_debug:nn{modules}{Namespace~URI:~\stex_uri_use:N \l_stex_current_ns_uri}
973 }

```

(End of definition for `\stex_set_current_namespace:`. This function is documented on page 139.)

We determine the PWD

`\c_stex_pwd_file`

`\c_stex_main_file`

```

974 \sys_if_platform_windows:TF{
975   \stex_get_env:Nn\l__stex_path_str{CD}
976 }{
977   \stex_get_env:Nn\l__stex_path_str{PWD}
978 }
979 \stex_file_resolve:No \c_stex_pwd_file \l__stex_path_str
980 \seq_set_eq:NN \c_stex_main_file \c_stex_pwd_file
981 \seq_put_right:Nx \c_stex_main_file {\jobname\tl_to_str:n{.tex}}
982
983 \stex_debug:nn {files} {PWD:~\stex_file_use:N \c_stex_pwd_file}

```

(End of definition for `\c_stex_pwd_file` and `\c_stex_main_file`. These variables are documented on page 138.)



### 13.2.9 File Hooks

keeps track of file changes:

```
984 \seq_gclear_new:N\g__stex_path_stack
985 \seq_gclear_new:N\g__stex_current_file
```

`\stex_filestack_push:n`

```
986 \cs_new_protected:Nn \stex_filestack_push:n {
987   \stex_str_if_ends_with:nnTF {#1}{.tex}{
988     \stex_file_resolve:No \g__stex_current_file {#1}
989   }{
990     \stex_file_resolve:No \g__stex_current_file {#1.tex}
991   }
992   \stex_if_file_absolute:NF \g__stex_current_file {
993     \stex_file_resolve:Nx \g__stex_current_file {
994       \stex_file_use:N \c_stex_pwd_file / \stex_file_use:N \g__stex_current_file
995     }
996   }
997   \seq_gset_eq:NN \g__stex_current_file \g__stex_current_file
998   \exp_args:NNx \seq_gpush:Nn \g__stex_path_stack {\stex_file_use:N \g__stex_current_file}
999   \stex_every_file:
1000 }
1001
1002 \cs_new_protected:Nn \stex_every_file: {
1003   \stex_set_document_uri:
1004   \stex_language_from_file:
1005   \stex_set_current_namespace:
1006 }
1007 %\AtBeginDocument{\stex_every_file:}
```

(End of definition for `\stex_filestack_push:n`. This function is documented on page 137.)

`\stex_filestack_pop:`

```
1008 \cs_new_protected:Nn \stex_filestack_pop: {
1009   \seq_if_empty:NF \g__stex_path_stack {
1010     \seq_gpop:NN \g__stex_path_stack \l__stex_path_str
1011   }
1012   \seq_if_empty:NTF \g__stex_path_stack {
1013     \seq_gset_eq:NN \g__stex_current_file \c_stex_main_file
1014   }{
1015     \seq_get:NN \g__stex_path_stack \l__stex_path_str
1016     \exp_args:NNo \stex_file_set:Nn \g__stex_current_file \l__stex_path_str
1017     \seq_gset_eq:NN \g__stex_current_file \g__stex_current_file
1018   }
1019   \stex_every_file:
1020 }
```

(End of definition for `\stex_filestack_pop:`. This function is documented on page 137.)

Hooks for the current file:

```
1021 \cs_new_protected:Nn \stex_input_with_hooks:n {
1022   \tl_set:Nn \l__stex_path_do_hooks_pre_tl {
1023     \tl_gset:Nn \l__stex_path_do_hooks_pre_tl {}
1024     \stex_debug:nn{HERE}{Hook~for~#1^^J\meaning\CurrentFilePath^^J\CurrentFile}
1025     \tl_if_empty:NTF\CurrentFilePath{
```

```

1026     \exp_args:No \stex_filestack_push:n {\CurrentFile}
1027   }{
1028     \exp_args:Ne \stex_filestack_push:n { \CurrentFilePath / \CurrentFile }
1029   }
1030 }
1031 \input{#1}
1032 \stex_debug:nn{HERE}{Hook~end~for~#1}
1033 \stex_filestack_pop:
1034 }
1035 \tl_new:N \l__stex_path_do_hooks_pre_tl {}
1036
1037 \AddToHook{file/before}{
1038   \l__stex_path_do_hooks_pre_tl
1039 }
1040 %\AddToHook{file/after}{ \stex_filestack_pop: }

```

### 13.3 Math Archives

```

1041 <@@=stex_mathhub>

```

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

```

1042
1043 \sys_if_platform_windows:TF{
1044   \stex_get_env:Nn \l__stex_mathhub_str {homedrive\c_percent_str\c_percent_str homedir}
1045 }{
1046   \stex_get_env:Nn \l__stex_mathhub_str {HOME}
1047 }
1048 \stex_file_resolve:No \c_stex_home_file \l__stex_mathhub_str
1049
1050 \str_if_empty:NTF\mathhub{
1051   \stex_get_env:Nn \l__stex_mathhub_str {MATHHUB}
1052   \str_if_empty:NTF \l__stex_mathhub_str {
1053     \ior_open:NnTF \g_tmpa_ior{\stex_file_use:N \c_stex_home_file/.stex/mathhub.path}{
1054       \group_begin:
1055         \escapechar=-1\catcode'\=12
1056         \ior_str_get:NN \g_tmpa_ior \l__stex_mathhub_str
1057         \str_gset_eq:NN \l__stex_mathhub_str \l__stex_mathhub_str
1058       \group_end:
1059       \ior_close:N \g_tmpa_ior
1060       \stex_debug:nn{mathhub}{MathHub-directory-determined-from-home-directory}
1061     }{
1062       \str_clear:N \l__stex_mathhub_str
1063     }
1064   }{
1065     \stex_debug:nn{mathhub}{MathHub-directory-determined-from-environment-variable}
1066   }
1067 }{
1068   \str_set_eq:NN \l__stex_mathhub_str \mathhub
1069 }
1070
1071 \str_if_empty:NTF \l__stex_mathhub_str {
1072   \msg_warning:nn{stex}{warning/nomathhub}

```

```

1073 \exp_args:NNe \stex_file_set:Nn \c_stex_mathhub_file {\stex_file_use:N \c_stex_home_file \
1074 \seq_clear:N \c_stex_mathhub_file
1075 }{
1076 \stex_file_resolve:No \c_stex_mathhub_file \l__stex_mathhub_str
1077 \stex_if_file_absolute:NF \c_stex_mathhub_file {
1078 \exp_args:NNe \stex_file_resolve:Nn \c_stex_mathhub_file {
1079 \stex_file_use:N \c_stex_main_file / .. / \l__stex_mathhub_str
1080 }
1081 }
1082 }
1083
1084 \exp_args:NNe \str_set:Nn \mathhub {\stex_file_use:N \c_stex_mathhub_file}
1085 \stex_debug:nn{mathhub}{MATHHUB:~\mathhub}

```

(End of definition for `\mathhub`, `\c_stex_home_file`, and `\c_stex_mathhub_file`. These variables are documented on page ??.)

`\l_stex_current_archive`

`\stex_set_current_archive:n`

```

1086 \cs_new_protected:Nn \stex_set_current_archive:n {
1087 \stex_require_archive:n { #1 }
1088 \stex_debug:nn{mathhub}{switching~to~archive~#1}
1089 \prop_set_eq:Nc \l_stex_current_archive_prop {
1090 c_stex_mathhub_#1_manifest_prop
1091 }
1092 }

```

(End of definition for `\l_stex_current_archive` and `\stex_set_current_archive:n`. These variables are documented on page ??.)

`\stex_in_archive:nn`

```

1093 \cs_new_protected:Nn \stex_in_archive:nn {
1094 \cs_if_exist:NF \l__stex_mathhub_cs {
1095 \cs_set:Npn \l__stex_mathhub_cs ##1 {}
1096 }
1097 \stex_pseudogroup:nn{
1098 \cs_set:Npn \l__stex_mathhub_cs ##1 {#2}
1099 \tl_if_empty:nTF{#1}{
1100 \prop_if_exist:NTF \l_stex_current_archive_prop {
1101 \exp_args:Ne \l__stex_mathhub_cs {\prop_item:Nn \l_stex_current_archive_prop { id }
1102 }{
1103 \l__stex_mathhub_cs {}
1104 }
1105 }{
1106 \stex_set_current_archive:n{#1}
1107 \l__stex_mathhub_cs {#1}
1108 }
1109 }{
1110 \stex_pseudogroup_restore:N \l_stex_current_archive_prop
1111 \cs_set:Npn \exp_not:N \l__stex_mathhub_cs ##1 {
1112 \exp_args:No \exp_not:n {\l__stex_mathhub_cs {##1}}
1113 }
1114 }
1115 }

```

(End of definition for `\stex_in_archive:nn`. This function is documented on page 134.)

`\stex_require_archive:n`

`\stex_require_archive:o`

```
1116 \cs_new_protected:Nn \stex_require_archive:n {
1117   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
1118     \seq_if_empty:NTF \c_stex_mathhub_file {
1119       \stex_fatal_error:n{warning/nomathhub}
1120     }{
1121       \stex_debug:nn{mathhub}{Opening~archive:~#1}
1122       \__stex_mathhub_do_manifest:n { #1 }
1123     }
1124   }
1125 }
1126 \cs_generate_variant:Nn \stex_require_archive:n {o}
```

(End of definition for `\stex_require_archive:n`. This function is documented on page 134.)

Code for finding and parsing manifest files:

```
1127 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
1128   \exp_args:Ne \__stex_mathhub_find_manifest:n {\stex_file_use:N \c_stex_mathhub_file / #1}
1129   \str_if_empty:NT \l__stex_mathhub_manifest_str {
1130     \stex_fatal_error:nxx{error/noarchive}
1131     {#1}{\stex_file_use:N \c_stex_mathhub_file}
1132   }
1133   \__stex_mathhub_parse_manifest:n {#1}
1134 }
1135
1136 \cs_new_protected:Nn \__stex_mathhub_find_manifest:n {
1137   \str_clear:N \l__stex_mathhub_manifest_str
1138   \seq_set_split:Nnn \l__stex_mathhub_seq / {#1}
1139   \bool_set_true:N \l__stex_mathhub_bool
1140   \bool_while_do:Nn \l__stex_mathhub_bool {
1141     \tl_if_eq:NNTF \l__stex_mathhub_seq \c_stex_mathhub_file {
1142       \bool_set_false:N \l__stex_mathhub_bool
1143     }{
1144       \__stex_mathhub_check_manifest:
1145       \bool_if:NT \l__stex_mathhub_bool {
1146         \seq_pop_right:NN \l__stex_mathhub_seq \l__stex_mathhub_tl
1147       }
1148     }
1149   }
1150 }
1151 \cs_generate_variant:Nn \__stex_mathhub_find_manifest:n {x}
1152
1153 \cs_new_protected:Nn \__stex_mathhub_check_manifest: {
1154   \__stex_mathhub_check_manifest:n {MANIFEST.MF}
1155   \bool_if:NT \l__stex_mathhub_bool {
1156     \__stex_mathhub_check_manifest:n {META-INF/MANIFEST.MF}
1157     \bool_if:NT \l__stex_mathhub_bool {
1158       \__stex_mathhub_check_manifest:n {meta-inf/MANIFEST.MF}
1159     }
1160   }
1161 }
1162
1163 \cs_new_protected:Nn \__stex_mathhub_check_manifest:n {
1164   \stex_debug:nn{mathhub}{Checking~\stex_file_use:N \l__stex_mathhub_seq / #1}
1165   \file_if_exist:nT {\stex_file_use:N \l__stex_mathhub_seq / #1} {
```

```

1166     \bool_set_false:N \l__stex_mathhub_bool
1167     \str_set:Nx \l__stex_mathhub_manifest_str {\stex_file_use:N \l__stex_mathhub_seq / #1}
1168   }
1169 }
1170
1171 \ior_new:N \c__stex_mathhub_manifest_ior
1172 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
1173   \ior_open:Nn \c__stex_mathhub_manifest_ior \l__stex_mathhub_manifest_str
1174   \prop_clear:N \l__stex_mathhub_prop
1175   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
1176     \exp_args:NNNo \exp_args:NNNx
1177     \seq_set_split:Nnn \l__stex_mathhub_seq \c_colon_str {\tl_to_str:n{##1}}
1178     \seq_pop_left:NNT \l__stex_mathhub_seq \l__stex_mathhub_key {
1179       \exp_args:NNo \str_set:Nn \l__stex_mathhub_key \l__stex_mathhub_key
1180       \str_set:Nx \l__stex_mathhub_val {\seq_use:Nn \l__stex_mathhub_seq :}
1181       \str_case:Nn \l__stex_mathhub_key {
1182         {id}                {\prop_put:Nno \l__stex_mathhub_prop { id }      \l__stex_mathhub_val}
1183         {narration-base}    {\prop_put:Nno \l__stex_mathhub_prop { narr }    \l__stex_mathhub_val}
1184         {url-base}          {\prop_put:Nno \l__stex_mathhub_prop { docurl }  \l__stex_mathhub_val}
1185         {source-base}       {\prop_put:Nno \l__stex_mathhub_prop { ns }     \l__stex_mathhub_val}
1186         {ns}                 {\prop_put:Nno \l__stex_mathhub_prop { ns }     \l__stex_mathhub_val}
1187       }
1188     }
1189   }
1190   \ior_close:N \c__stex_mathhub_manifest_ior
1191   \prop_gset_eq:cN { c_stex_mathhub_#1_manifest_prop } \l__stex_mathhub_prop
1192   \stex_debug:nn{mathhub}{Result:~\prop_to_keyval:N \l__stex_mathhub_prop}
1193   \stex_persist:e {
1194     \prop_gset_from_keyval:cn {c_stex_mathhub_#1_manifest_prop}{
1195       \prop_to_keyval:N \l__stex_mathhub_prop
1196     }
1197   }
1198 }

```

Current MathHub archive

`\c_stex_main_archive_prop`  
`\l_stex_current_archive_prop`

```

1199 \cs_new_protected:Nn \_stex_main_archive: {
1200   \stex_if_file_starts_with:NNTF \c_stex_pwd_file \c_stex_mathhub_file {
1201     \__stex_mathhub_find_manifest:x { \stex_file_use:N \c_stex_pwd_file }
1202     \str_if_empty:NNTF \l__stex_mathhub_manifest_str {
1203       \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~archive}
1204     }{
1205       \__stex_mathhub_parse_manifest:n { main }
1206       \prop_set_eq:NN \c_stex_main_archive_prop \c_stex_mathhub_main_manifest_prop
1207       \cs_undefine:N \c_stex_mathhub_main_manifest_prop
1208       \prop_get:NnN \c_stex_main_archive_prop {id}
1209       \l__stex_mathhub_str
1210       \prop_set_eq:cN { c_stex_mathhub\_l__stex_mathhub_str_manifest_prop }
1211       \c_stex_main_archive_prop
1212       \exp_args:No \stex_set_current_archive:n { \l__stex_mathhub_str }
1213       \stex_debug:nn{mathhub}{Current~archive:~
1214         \prop_item:Nn \l_stex_current_archive_prop {id}
1215       }
1216       \bool_if:NT \c_stex_persist_write_mode_bool {

```

```

1217     \tl_put_right:Nx \_stex_persist_read_now: {
1218     \stex_persist:n {
1219         \prop_gset_from_keyval:cn {c_stex_mathhub\_l__stex_mathhub_str _manifest_prop}{
1220             \prop_to_keyval:N \c_stex_main_archive_prop
1221         }
1222         \prop_gset_eq:Nc \exp_not:N \l_stex_current_archive_prop {
1223             c_stex_mathhub\_l__stex_mathhub_str _manifest_prop
1224         }
1225         \prop_gset_eq:Nc \exp_not:N \c_stex_main_archive_prop {
1226             c_stex_mathhub\_l__stex_mathhub_str _manifest_prop
1227         }
1228     }
1229 }
1230 }
1231 }
1232 }{
1233     \stex_debug:nn{mathhub}{Not~currently~in~the~MathHub~directory}
1234 }
1235 }
1236
1237 %\bool_if:NF \c_stex_persist_mode_bool {
1238     \_stex_main_archive:
1239 }%

```

(End of definition for `\c_stex_main_archive_prop` and `\l_stex_current_archive_prop`. These variables are documented on page [134](#).)

## 13.4 Documents

### 13.4.1 Title

Stores the title, if it exists:

```

1240 <@@=stex_doc>
1241 \tl_new:N \g__stex_doc_title_tl

```

`\stexdoctitle` Initial definition, will be changed at begin document:

```

1242 \cs_new_protected:Npn \stexdoctitle #1 {
1243     \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1244     \global\def\stexdoctitle##1{}
1245 }

```

At begin document, we switch to:

```

1246 \cs_new_protected:Nn \__stex_doc_set_title:n {
1247     \stex_if_smsmode:F{
1248         \global\def\stexdoctitle##1{}
1249         \stex_debug:nn{title}{Setting~title~to:\tl_to_str:n{#1}}
1250         \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1251         \__stex_doc_title_html:
1252     }
1253 }

```

Hooks, changes and HTML:

```

1254 \cs_new_protected:Nn \__stex_doc_title_html: {
1255     \stex_if_do_html:T{\stex_if_html_backend:T{

```

```

1256     \stex_annotate_invisible:nn{shtml:doctitle={}}{ \hbox{\g__stex_doc_title_tl} }
1257   }}
1258 }
1259
1260 \AtBeginDocument {
1261   \tl_if_empty:NTF \g__stex_doc_title_tl {
1262     \cs_set_eq:NN \stexdoctitle \__stex_doc_set_title:n
1263   }{
1264     \stex_debug:nn{title}{Setting~title~to:\exp_args:No\tl_to_str:n\g__stex_doc_title_tl}
1265     \global\def\stexdoctitle#1{}
1266     \__stex_doc_title_html:
1267   }
1268
1269   \cs_set_eq:NN \__stex_doc_maketitle: \maketitle
1270   \global\protected\def\maketitle{
1271     \tl_if_empty:NF \@title {
1272       \exp_args:No \stexdoctitle \@title
1273     }
1274     \__stex_doc_maketitle:
1275   }
1276 }

```

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

## 13.4.2 Sectioning

```

1277 \int_new:N \l_stex_docheader_sect
1278
1279 \tl_set:Nn \stex_current_section_level {document}
1280
1281 \cs_set_protected:Npn \currentsectionlevel {
1282   \stex_if_do_html:TF{
1283     \stex_annotate:nn{shtml:currentsectionlevel={},shtml:capitalize=false}{ }
1284   }{
1285     \stex_current_section_level
1286   }
1287   \tl_if_exist:NT\xspace\xspace
1288 }
1289
1290 \cs_set_protected:Npn \Currentsectionlevel {
1291   \stex_if_do_html:TF{
1292     \stex_annotate:nn{shtml:currentsectionlevel={},shtml:capitalize=true}{ }
1293   }{
1294     \exp_args:No \_stex_capitalize:n \stex_current_section_level
1295   }
1296   \tl_if_exist:NT\xspace\xspace
1297 }
1298
1299 \stex_if_html_backend:TF {
1300   \cs_new_protected:Nn \_sfragment_do_level:nn {
1301     \stexdoctitle{#2}
1302     \par
1303     \begin{stex_annotate_env}{shtml:section={\int_use:N \l_stex_docheader_sect}}
1304       \noindent\stex_annotate:nn{shtml:sectiontitle={}}{ }

```

```

1305     \stex_annotate_force_break:n{#2}
1306   }\par
1307 }
1308 \cs_new_protected:Nn \sfragment_end: {
1309   \end{stex_annotate_env}
1310 }
1311 }{
1312 \cs_new_protected:Nn \sfragment_do_level:nn {
1313   \stexdoctitle{#2}
1314   \tl_if_empty:NTF \l_stex_key_short_tl {
1315     \use:c{#1}
1316   }{
1317     \exp_args:Nnx \use:nn{\use:c{#1}}{[\exp_args:No \exp_not:n \l_stex_key_short_tl]}
1318   }{#2}
1319   \int_incr:N \l_stex_doheader_sect
1320   \tl_set:Nn \stex_current_section_level{#1}
1321 }
1322 \cs_new_protected:Nn \sfragment_end: {}
1323 }
1324
1325
1326 \cs_new_protected:Npn \__stex_doc_do_section:n {
1327   \int_case:nnF \l_stex_doheader_sect {
1328     {0}{\cs_if_exist:NTF \thepart {\sfragment_do_level:nn{part}}{
1329       \int_incr:N \l_stex_doheader_sect
1330       \__stex_doc_do_section:n
1331     }}
1332     {1}{\cs_if_exist:NTF \thechapter {\sfragment_do_level:nn{chapter}}{
1333       \int_incr:N \l_stex_doheader_sect
1334       \__stex_doc_do_section:n
1335     }}
1336     {2}{\sfragment_do_level:nn{section}}
1337     {3}{\sfragment_do_level:nn{subsection}}
1338     {4}{\sfragment_do_level:nn{subsubsection}}
1339     {5}{\sfragment_do_level:nn{paragraph}}
1340   }{\sfragment_do_level:nn{subparagraph}}
1341 }
1342
1343 \stex_keys_define:nmmn{ sfragment }{
1344   \tl_clear:N \l_stex_key_short_tl
1345 }{
1346   short .tl_set:N = \l_stex_key_short_tl
1347 }{id}
1348
1349 \NewDocumentEnvironment{sfragment}{ 0{ } m}{
1350   \stex_keys_set:nn{sfragment}{#1}
1351   \__stex_doc_do_section:n{#2}
1352   \stex_do_id:
1353 }{
1354   \sfragment_end:
1355 }
1356
1357 %\int_incr:N \l_stex_doheader_sect
1358 \NewDocumentEnvironment{blindfragment}{}{

```



```

1359 \__stex_doc_skip_section:
1360 }{
1361 \stex_if_html_backend:T{
1362 \stex_annotate_invisible:n{~}
1363 \end{stex_annotate_env}
1364 }
1365 }
1366
1367
1368 \cs_new_protected:Nn \__stex_doc_skip_section_i: {
1369 \int_case:nn \l_stex_docheader_sect {
1370 {0}{\cs_if_exist:NF \thepart {
1371 \int_incr:N \l_stex_docheader_sect \__stex_doc_skip_section_i:
1372 }}
1373 {1}{\cs_if_exist:NF \thechapter {
1374 \int_incr:N \l_stex_docheader_sect \__stex_doc_skip_section_i:
1375 }}
1376 }
1377 \int_incr:N \l_stex_docheader_sect
1378 }
1379
1380 \stex_if_html_backend:TF {
1381 \cs_new_protected:Nn \__stex_doc_skip_section: {
1382 \__stex_doc_skip_section_i:
1383 \begin{stex_annotate_env}{shtml:skipsection={\int_use:N \l_stex_docheader_sect}}
1384 \stex_annotate_invisible:n{~}
1385 }
1386 }{
1387 \cs_set_eq:NN \__stex_doc_skip_section: \__stex_doc_skip_section_i:
1388 }
1389
1390
1391 \cs_new_protected:Nn \__stex_doc_skip_fragment:n {
1392 \stepcounter{#1}
1393 }
1394
1395 \cs_new_protected:Npn \skipfragment {
1396 \int_case:nnF \l_stex_docheader_sect {
1397 {0}{\cs_if_exist:NTF \thepart {\__stex_doc_skip_fragment:n{part}}{
1398 \int_incr:N \l_stex_docheader_sect
1399 \skipfragment
1400 }}
1401 {1}{\cs_if_exist:NTF \thechapter {\__stex_doc_skip_fragment:n{chapter}}{
1402 \int_incr:N \l_stex_docheader_sect
1403 \skipfragment
1404 }}
1405 {2}{\__stex_doc_skip_fragment:n{section}}
1406 {3}{\__stex_doc_skip_fragment:n{subsection}}
1407 {4}{\__stex_doc_skip_fragment:n{subsubsection}}
1408 {5}{\__stex_doc_skip_fragment:n{paragraph}}
1409 }{\__stex_doc_skip_fragment:n{subparagraph}}
1410 }

```

`\setsectionlevel`

```

1411 \cs_new_protected:Npn \setsectionlevel #1 {
1412   \str_case:nnF{#1}{
1413     {part}{\int_set:Nn \l_stex_docheader_sect 0}
1414     {chapter}{\int_set:Nn \l_stex_docheader_sect 1}
1415     {section}{\int_set:Nn \l_stex_docheader_sect 2}
1416     {subsection}{\int_set:Nn \l_stex_docheader_sect 3}
1417     {subsubsection}{\int_set:Nn \l_stex_docheader_sect 4}
1418     {paragraph}{\int_set:Nn \l_stex_docheader_sect 5}
1419   }{
1420     \int_set:Nn \l_stex_docheader_sect 6
1421   }
1422   \cs_if_eq:NNTF\@onlypreamble\@notprerr{
1423     \stex_annotate_invisible:nn{shtml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
1424   }{}
1425 }
1426
1427 \stex_if_html_backend:T{
1428   \cs_new_protected:Nn \__stex_doc_check_topsect: {
1429     \int_case:nnF \l_stex_docheader_sect {
1430       {0}{\cs_if_exist:NTF \thepart {
1431         \stex_annotate_invisible:nn{shtml:sectionlevel=0}}{}
1432       }{
1433         \int_incr:N \l_stex_docheader_sect
1434         \__stex_doc_check_topsect:
1435       }}
1436       {1}{\cs_if_exist:NTF \thechapter {
1437         \stex_annotate_invisible:nn{shtml:sectionlevel=1}}{}
1438       }{
1439         \int_incr:N \l_stex_docheader_sect
1440         \__stex_doc_check_topsect:
1441       }}
1442     }{
1443       \stex_annotate_invisible:nn{shtml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
1444     }
1445   }
1446   \AtBeginDocument{\__stex_doc_check_topsect:}
1447 }
1448
1449 \AtBeginDocument{
1450   \bool_if:NF \c_stex_no_frontmatter_bool {
1451     \cs_if_exist:NTF\frontmatter{
1452       \let\__stex_doc_orig_frontmatter\frontmatter
1453       \let\frontmatter\relax
1454     }{
1455       \tl_set:Nn\__stex_doc_orig_frontmatter{
1456         \clearpage
1457         %\@mainmatterfalse
1458         \pagenumbering{roman}
1459       }
1460     }
1461     \cs_if_exist:NTF\backmatter{
1462       \let\__stex_doc_orig_backmatter\backmatter
1463       \let\backmatter\relax
1464     }{

```

```

1465     \tl_set:Nn\__stex_doc_orig_backmatter{
1466       \clearpage
1467       %\@mainmatterfalse
1468       \pagenumbering{roman}
1469     }
1470   }
1471   \newenvironment{frontmatter}{
1472     \__stex_doc_orig_frontmatter
1473   }{
1474     \cs_if_exist:NTF\mainmatter{
1475       \mainmatter
1476     }{
1477       \clearpage
1478       %\@mainmattertrue
1479       \pagenumbering{arabic}
1480     }
1481   }
1482   \newenvironment{backmatter}{
1483     \__stex_doc_orig_backmatter
1484   }{
1485     \cs_if_exist:NTF\mainmatter{
1486       \mainmatter
1487     }{
1488       \clearpage
1489       %\@mainmattertrue
1490       \pagenumbering{arabic}
1491     }
1492   }
1493 }
1494 }

```

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

### 13.4.3 References

```
1495 <@@=stex_refs>
```

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

```

1496 \iow_new:N \c__stex_refs_iow
1497 \AtBeginDocument{\iow_open:Nn \c__stex_refs_iow {\jobname.sref}}
1498 \AtEndDocument{\iow_close:N \c__stex_refs_iow}

```

The following macros are written to the `.aux`-file, and hence use L<sup>A</sup>T<sub>E</sub>X<sub>2</sub>ε character code scheme:

```

1499 \cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}
1500
1501 \cs_new_protected:Npn \STEXInternalSetSrefSymURL #1 #2 {
1502   \str_gset:cn{g_stex_sref_sym_t1_to_str:n{#1}_target}{#2}
1503 }
1504

```

```

\stex_ref_new_doc_target:n
  \sreflabel

```

```

1505 \seq_new:N \g__stex_refs_files_seq
1506 \int_new:N \l__stex_refs_unnamed_counter_int

```

```

1507
1508 \cs_new_protected:Nn \_stex_ref_new_id:n {
1509   \str_if_empty:nTF {#1}{
1510     \int_gincr:N \l__stex_refs_unnamed_counter_int
1511     \str_set:Nx \l__stex_refs_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1512   }{
1513     \str_set:Nn \l__stex_refs_str {#1}
1514   }
1515   \str_set:Nx \l_stex_ref_url_str {\stex_uri_use:N \l_stex_current_doc_uri ? \l__stex_refs_s
1516 }
1517
1518 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1519   \_stex_ref_new_id:n{#1}
1520   %\stex_uri_add_module:NNo \l__stex_refs_uri \l_stex_current_doc_uri \l__stex_refs_str
1521   %\stex_debug:nn{sref}{New~document~target::~\stex_uri_use:N \l__stex_refs_uri}
1522   \__stex_refs_add_doc_ref:xo {\stex_uri_use:N \l_stex_current_doc_uri} \l__stex_refs_str
1523   \stex_if_smsmode:F {
1524     \iow_now:Nx \c__stex_refs_iow {
1525       \STEXInternalSrefRestoreTarget
1526       {\stex_uri_use:N \l_stex_current_doc_uri}
1527       {\l__stex_refs_str}
1528       {\@currentcounter}
1529       {\@currentlabel}
1530       {
1531         \tl_if_exist:NT\@currentlabelname{
1532           \exp_args:No\exp_not:n\@currentlabelname
1533         }
1534       }
1535     }
1536     \exp_args:Nx \label {sref@\l_stex_ref_url_str}
1537     \stex_if_do_html:T {
1538       \pdfdest name "sref@\l_stex_ref_url_str" xyz\relax
1539     }
1540   }
1541 }
1542 \NewDocumentCommand \sreflabel {m} {\stex_ref_new_doc_target:n {#1}}
1543
1544 \cs_new_protected:Nn \__stex_refs_add_doc_ref:nn {
1545   \seq_if_in:NnTF \g__stex_refs_files_seq {#1} {
1546     \seq_if_in:cnF {g__stex_refs_#1_seq}{#2}{
1547       \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1548     }
1549   }{
1550     \seq_gput_right:Nn \g__stex_refs_files_seq {#1}
1551     \seq_new:c{g__stex_refs_#1_seq}
1552     \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1553   }
1554 }
1555 \cs_generate_variant:Nn \__stex_refs_add_doc_ref:nn {xo,xx}

```

(End of definition for `\stex_ref_new_doc_target:n` and `\sreflabel`. These functions are documented on page 121.)

`\sref` Optional arguments:  
`\extref`

```

1556 \stex_keys_define:nnnn{sref / 1}{-}{
1557   % TODO get rid of this
1558   fallback .code:n = {},
1559   pre      .code:n = {},
1560   post     .code:n = {}
1561 }{archive file}
1562 \stex_keys_define:nnnn{sref / 2}{-}{-}{archive file, title}
1563
1564 \str_new:N \l__stex_refs_default_archive_str
1565 \str_new:N \l__stex_refs_default_file_str
1566 \tl_new:N \l__stex_refs_default_title_tl
1567
1568 \cs_set_protected:Nn \__stex_refs_set_keys_b:n {
1569   \tl_if_empty:nTF{#1}{
1570     \str_set_eq:NN \l_stex_key_archive_str \l__stex_refs_default_archive_str
1571     \str_set_eq:NN \l_stex_key_file_str \l__stex_refs_default_file_str
1572     \tl_set_eq:NN \l_stex_key_title_tl \l__stex_refs_default_title_tl
1573   }{
1574     \stex_keys_set:nn{ sref / 2 }{ #1 }
1575   }
1576 }
1577
1578 \newcommand\srefsetin[3] []{
1579   \str_set:Nx \l__stex_refs_default_archive_str {#1}
1580   \str_set:Nx \l__stex_refs_default_file_str {#2}
1581   \tl_set:Nn \l__stex_refs_default_title_tl {#3}
1582 }
1583
1584 Auxiliary methods:
1585
1586 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1587   \str_clear:N \l__stex_refs_uri_str
1588   \stex_debug:nn{sref}{
1589     File:~\l_stex_key_file_str^^J
1590     Repo:\l_stex_key_archive_str
1591   }
1592   \str_if_empty:NTF \l_stex_key_file_str {
1593     \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1594     \seq_if_exist:cT{g__stex_refs_\stex_uri_use:N \l_stex_current_doc_uri _seq}{
1595       \exp_args:Nnx \__stex_refs_find_uri_in_file:nnn{#1}
1596       {\stex_uri_use:N \l_stex_current_doc_uri}\seq_map_break:
1597     }
1598     \str_if_empty:NT \l__stex_refs_uri_str {
1599       \seq_map_inline:Nn \g__stex_refs_files_seq {
1600         \__stex_refs_find_uri_in_file:nnn{#1}{##1}{\seq_map_break:n{\seq_map_break:}}
1601       }
1602     }
1603   }{
1604     \str_if_empty:NTF \l_stex_key_archive_str {
1605       \prop_if_exist:NTF \l_stex_current_archive_prop {
1606         \__stex_refs_find_uri_in_prop_file:N \l_stex_current_archive_prop
1607       }{
1608         \stex_file_resolve:Nx \l__stex_refs_file
1609         { \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str }

```

```

1608     \str_set:Nx \l__stex_refs_uri_str { file:/ \stex_file_use:N \l__stex_refs_file }
1609   }
1610   ){
1611     \stex_require_archive:o \l_stex_key_archive_str
1612     \prop_set_eq:Nc \l__stex_refs_prop { c_stex_mathhub_\l_stex_key_archive_str _manifest
1613     \__stex_refs_find_uri_in_prop_file:N \l__stex_refs_prop
1614   }
1615 }
1616 }
1617
1618 \cs_new_protected:Nn \__stex_refs_find_uri_in_prop_file:N {
1619   \str_set:Nx \l__stex_refs_uri_str {
1620     \stex_file_use:N \c_stex_mathhub_file /
1621     \prop_item:Nn #1 {id} /
1622     source / \l_stex_key_file_str .sref
1623   }
1624   \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_uri_str
1625   \stex_uri_from_repo_file:NNNn \l__stex_refs_uri #1
1626   \l__stex_refs_file {narr}
1627   \str_set:Nx \l__stex_refs_uri_str {\stex_uri_use:N \l__stex_refs_uri}
1628 }
1629
1630 \cs_new_protected:Nn \__stex_refs_find_uri_in_file:nnn {
1631   \stex_debug:nn{sref}{Checking~file~#2}
1632   \seq_map_inline:cn{g__stex_refs_#2_seq}{
1633     \str_if_eq:nnT{#1}{##1}{
1634       \str_set:Nx \l__stex_refs_uri_str {\stex_uri_use:N \l_stex_current_doc_uri}
1635       \stex_debug:nn{sref}{Found.}
1636       #3
1637     }
1638   }
1639 }

```

Doing the actual referencing:

```

1640 \cs_new_protected:Nn \__stex_refs_do_autoref:n {
1641   \cs_if_exist:cTF{autoref}{
1642     \exp_args:Nx\autoref{sref@#1}
1643   }{
1644     \exp_args:Nx\ref{sref@#1}
1645   }
1646 }
1647
1648 \cs_new_protected:Nn \__stex_refs_do_sref:nn {
1649   \str_if_empty:NTF \l__stex_refs_uri_str {
1650     \str_if_empty:NTF \l_stex_key_file_str {
1651       \stex_debug:nn{sref}{autoref~on~\stex_uri_use:N \l_stex_current_doc_uri?#1}
1652       \exp_args:Ne \__stex_refs_do_autoref:n{\stex_uri_use:N \l_stex_current_doc_uri ? #1}
1653     }{
1654       \stex_debug:nn{sref}{srefin~on~#1}
1655       \__stex_refs_set_keys_b:n{ #2 }
1656       \__stex_refs_do_sref_in:n{#1}
1657     }
1658   }{
1659     \exp_args:NNo \seq_if_in:NnTF \g__stex_refs_files_seq \l__stex_refs_uri_str {

```

```

1660 \stex_debug:nn{sref}{Using~ref~file~\l__stex_refs_uri_str}
1661 \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref\_l__stex_refs_uri_str_seq}{\detokenize{#1}}{
1662 \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref~on~\l__stex_refs_uri_str?}
1663 \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1664 }{
1665 \str_if_empty:NTF \l_stex_key_file_str {
1666 \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
1667 \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1668 }{
1669 \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l__stex_refs_uri_str?#1}
1670 \__stex_refs_set_keys_b:n{ #2 }
1671 \__stex_refs_do_sref_in:n{#1}
1672 }
1673 }
1674 }{
1675 \stex_debug:nn{sref}{No~ref~file~found~for~\l__stex_refs_uri_str}
1676 \str_if_empty:NTF \l_stex_key_file_str {
1677 \stex_debug:nn{sref}{in~empty;~autoref~on~\l__stex_refs_uri_str?#1}
1678 \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1679 }{
1680 \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l__stex_refs_uri_str?#1}
1681 \__stex_refs_set_keys_b:n{ #2 }
1682 \__stex_refs_do_sref_in:n{#1}
1683 }
1684 }
1685 }
1686 }
1687
1688 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1689 \stex_debug:nn{sref}{In: \l_stex_key_file_str^^JRepo:\l_stex_key_archive_str}
1690 \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1691 \tl_if_exist:cTF{r@sref@\l__stex_refs_uri_str?#1}{
1692 \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1693 }{
1694 %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1695 \group_begin:\catcode13=9\relax\catcode10=9\relax
1696 \str_if_empty:NTF \l_stex_key_archive_str {
1697 \prop_if_exist:NTF \l_stex_current_archive_prop {
1698 \str_set:Nx \l__stex_refs_file_str {
1699 \stex_file_use:N \c_stex_mathhub_file /
1700 \prop_item:Nn \l_stex_current_archive_prop { id }
1701 / source / \l_stex_key_file_str .sref
1702 }
1703 }{
1704 \str_set:Nx \l__stex_refs_file_str {
1705 \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str .sref
1706 }
1707 }
1708 }{
1709 \str_set:Nx \l__stex_refs_file_str {
1710 \stex_file_use:N \c_stex_mathhub_file / \l_stex_key_archive_str
1711 / source / \l_stex_key_file_str .sref
1712 }
1713 }

```

```

1714 \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_file_str
1715 \str_set:Nx \l__stex_refs_file_str {\stex_file_use:N \l__stex_refs_file }
1716 \stex_debug:nn{sref}{File: \l__stex_refs_file_str }
1717 \exp_args:NNNx \exp_args:No \str_if_eq:nnTF \l__stex_refs_file_str {\stex_file_use:N\
1718 \__stex_refs_do_autoref:n{
1719 \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1720 }
1721 }{
1722 \exp_args:No \IfFileExists \l__stex_refs_file_str {
1723 \tl_clear:N \l__stex_refs_return_tl
1724 \str_set:Nn \l__stex_refs_id_str {#1}
1725 \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nmnnn
1726 \use:c{@ @ input}{\l__stex_refs_file_str}
1727 \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1728 \exp_args:Nnno \msg_warning:nmnn{stex}{warning/smslabelmissing}\l__stex_refs_fil
1729 \__stex_refs_do_autoref:n{
1730 \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1731 }
1732 }{
1733 \l__stex_refs_return_tl
1734 }
1735 }{
1736 \exp_args:Nnno \msg_warning:nmnn{stex}{warning/smsmissing}\l__stex_refs_file_str
1737 \__stex_refs_do_autoref:n{
1738 \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1739 }
1740 }
1741 }
1742 \group_end:
1743 }
1744 }
1745
1746 \cs_new_protected:Nn \__stex_refs_do_return:nmnn {
1747 \tl_set:Nn \l__stex_refs_return_tl {
1748 \stex_annotate:nn{shtml:sref={#4},shtml:srefin={\l__stex_refs_file_str}}{
1749 \use:c{#3autorefname}~#1\tl_if_empty:nF{#2}{~(#2)}
1750 \tl_if_empty:NF\l_stex_key_title_tl{
1751 {}~in~\l_stex_key_title_tl
1752 }
1753 }
1754 }
1755 }
1756
1757 \cs_new_protected:Nn \__stex_refs_restore_target:nmnnn {
1758 \str_if_empty:NTF \l__stex_refs_uri_str {
1759 \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1760 \__stex_refs_do_return:nmnn{#4}{#5}{#3}{#1?#2}
1761 }
1762 }{
1763 \stex_debug:nn{sref}{\l__stex_refs_uri_str{}}~ == ~ #1 ~ ?}
1764 \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1765 \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1766 \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1767 \stex_debug:nn{sref}{success!}

```



```

1768     \_stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
1769     \endinput
1770   }
1771 }
1772 }
1773 }

```

The actual macros:

```

1774 \NewDocumentCommand \sref { 0{} m 0{} }{
1775   \stex_keys_set:nn { sref / 1 }{ #1 }
1776   \_stex_refs_find_uri:n { #2 }
1777   \_stex_refs_do_sref:nn{#2}{#3}
1778 }
1779 \NewDocumentCommand \extref { 0{} m m }{
1780   \stex_keys_set:nn { sref / 1 }{ #1 }
1781   \_stex_refs_find_uri:n { #2 }
1782   \_stex_refs_set_keys_b:n{ #3 }
1783   \str_if_empty:NT \l_stex_key_file_str {
1784     \msg_error:nn{stex}{error/extrefmissing}
1785   }
1786   \_stex_refs_do_sref_in:n{#2}
1787 }

```

(End of definition for `\sref` and `\extref`. These functions are documented on page 84.)

`\stex_ref_new_sym_target:n`

```

1788 \cs_new_protected:Nn \stex_ref_new_symbol:n {
1789   \cs_if_exist:cF{r@sref@sym@\tl_to_str:n{#1}}{
1790     \_stex_refs_new_symbol:n{#1}
1791   }
1792 }
1793
1794 \cs_new_protected:Nn \_stex_sref_do_aux:n {
1795   #1 \iow_now:Nn \@auxout {#1}
1796 }
1797
1798 \cs_new_protected:Nn \_stex_refs_new_symbol:n {
1799   \prop_if_exist:NTF \l_stex_current_archive_prop {
1800     \prop_get:NnNTF \l_stex_current_archive_prop {docurl} \l__stex_refs_str {
1801       \exp_args:Ne \_stex_sref_do_aux:n {
1802         \STEXInternalSetSrefSymURL{#1}{\l__stex_refs_str / symbol? #1}
1803       }
1804     }{
1805       \_stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{ } }
1806     }
1807   }{
1808     \_stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{ } }
1809   }
1810 }
1811
1812 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1813   \cs_if_exist:NT \hypertarget{
1814     \exp_args:Ne \hypertarget{\tl_to_str:n{sref@sym@ #1}}{ }
1815     \str_gset:cx{\tl_to_str:n{r@sref@sym@ #1}}{\tl_to_str:n{sref@sym@ #1}}
1816   }

```

```

1817 }
1818
1819 \cs_new_protected:Nn \stex_ref_new_sym_target:nn {
1820   \str_if_eq:nnTF{#1}{#2}{
1821     \stex_ref_new_sym_target:n{#1}
1822   }{
1823     \str_gset:cn{g_stex_sref_sym_ #1 _label}{#2}
1824   }
1825 }

```

(End of definition for `\stex_ref_new_sym_target:n`. This function is documented on page 121.)

### `\srefsym`

```

1826 \NewDocumentCommand \srefsym { m m }{
1827   \stex_get_symbol:n { #1 }
1828   \exp_args:Ne
1829   \__stex_refs_sym_aux:nn{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
1830 }
1831
1832 \cs_new_protected:Npn \__stex_refs_do_internal_link:nn #1 {
1833   \cs_if_exist:NTF \hyperlink {
1834     \hyperlink{#1}
1835   }\use:n
1836 }
1837
1838 \cs_new_protected:Npn \__stex_refs_do_url_link:nn {
1839   \cs_if_exist:NTF \href \href \use_ii:nn
1840 }
1841
1842 \cs_new_protected:Npn \__stex_refs_sym_aux:nn #1 {
1843   \cs_if_exist:cTF{\tl_to_str:n{r@sref@sym@#1}}{
1844     \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1845   }{
1846     \str_if_exist:cTF{g_stex_sref_sym_#1_label}{
1847       \exp_args:Ne \__stex_refs_sym_aux:nn{\use:c{g_stex_sref_sym_#1_label}}
1848     }{
1849       \str_if_empty:cTF{g_stex_sref_sym_#1_target}{
1850         \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1851       }{
1852         \exp_args:Ne \__stex_refs_do_url_link:nn{\use:c{g_stex_sref_sym_#1_target}}
1853       }
1854     }
1855   }
1856 }

```

(End of definition for `\srefsym`. This function is documented on page 89.)

### `\srefsymuri`

```

1857 \cs_new_protected:Npn \srefsymuri #1 {
1858   \__stex_refs_sym_aux:nn{#1}
1859 }

```

(End of definition for `\srefsymuri`. This function is documented on page 89.)

### 13.4.4 Inputs

1860 <@@=stex\_inputs>

**\stex\_resolve\_path\_pair:Nnn**  
**\stex\_resolve\_path\_pair:Nxx**

```

1861 \cs_new_protected:Nn \stex_resolve_path_pair:Nnn {
1862   \stex_debug:nn{resolving-path}{#3-in-[#2]}
1863   \str_if_empty:nTF{#2}{
1864     \prop_if_exist:NTF \l_stex_current_archive_prop {
1865       \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1866         \prop_item:Nn \l_stex_current_archive_prop { id } / source /
1867         #3}
1868       \stex_debug:nn{resolving-path}{In-current-archive-
1869         \prop_item:Nn \l_stex_current_archive_prop { id }
1870         ;-result:~#1}
1871     }{
1872       \str_set:Nx #1 {\stex_file_use:N \c_stex_pwd_file / .. / #3 }
1873       \stex_debug:nn{resolving-path}{No-current-archive;-result:~#1}
1874     }
1875   }{
1876     \stex_require_archive:n { #2 }
1877     \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1878       \prop_item:cn {c_stex_mathhub_#2_manifest_prop} { id } / source /
1879       #3}
1880     \stex_debug:nn{resolving-path}{result:~#1}
1881   }
1882 }
1883 \cs_generate_variant:Nn \stex_resolve_path_pair:Nnn {Nxx}

```

(End of definition for \stex\_resolve\_path\_pair:Nnn. This function is documented on page 134.)

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

```

1884 \newif \ifinputref \inputreffalse
1885
1886 \cs_new_protected:Nn \__stex_inputs_mhinput:nn {
1887   \stex_in_archive:nn {#1} {
1888     \ifinputref
1889       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1890     \else
1891       \inputreftrue
1892       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1893     \inputreffalse
1894   \fi
1895 }
1896 }
1897
1898 \NewDocumentCommand \mhinput { 0 } m>{
1899   \exp_args:NNx\exp_args:Nnx\__stex_inputs_mhinput:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2} }
1900 }
1901
1902 \cs_new_protected:Nn \__stex_inputs_inputref_html:nn {
1903   \str_clear:N \l_tmpa_str
1904   \prop_get:NnNF \l_stex_current_archive_prop { narr } \l_tmpa_str {
1905     \prop_get:NnNF \l_stex_current_archive_prop { ns } \l_tmpa_str {}
1906   }

```

```

1907 \tl_if_empty:nTF{ #1 }{
1908   \IfFileExists{#2}{
1909     \ifvmode\noindent\fi\stex_annotate_invisible:nn{shtml:inputref={
1910       \l_tmpa_str / #2
1911     }}{ }
1912   }{
1913     \stex_input_with_hooks:n{#2}
1914   }
1915 }{
1916   \IfFileExists{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }{
1917     \ifvmode\noindent\fi\stex_annotate_invisible:nn{shtml:inputref={
1918       \l_tmpa_str / #2
1919     }}{ }
1920   }{
1921     \input{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1922   }
1923 }
1924 }
1925
1926 \cs_new_protected:Nn \__stex_inputs_inputref_pdf:nn {
1927   \begingroup
1928     \inputreftrue
1929     \tl_if_empty:nTF{ #1 }{
1930       \stex_input_with_hooks:n{#2}
1931     }{
1932       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1933     }
1934   \endgroup
1935 }
1936
1937 \cs_new_protected:Nn \__stex_inputs_inputref:nn {
1938   \stex_in_archive:nn {#1} {
1939     \stex_if_html_backend:TF
1940       \__stex_inputs_inputref_html:nn
1941       \__stex_inputs_inputref_pdf:nn
1942     {##1}{#2}
1943   }
1944 }
1945
1946 \NewDocumentCommand \inputref { 0{} m}{
1947   \exp_args:NNx \exp_args:Nnx \__stex_inputs_inputref:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2} }
1948 }

```

(End of definition for `\inputref`, `\mhinput`, and `\ifinputref`. These functions are documented on page 83.)

### `\addmhbibresource`

```

1949 \cs_new_protected:Nn \__stex_inputs_bibresource:n {
1950   \__stex_inputs_up_archive:nn{#1}{bib}
1951   \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
1952     \msg_error:nxx{stex}{error/nofile}{\exp_not:N\addmhbibresource}{#1.bib}
1953   }{
1954     \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
1955       \addbibresource{ ##1 }

```

```

1956     }
1957   }
1958 }
1959 \newcommand\addmhbibresource[2][]{
1960   \tl_if_empty:nTF{#1}{
1961     \__stex_inputs_bibresource:n{#2}
1962   }{
1963     \stex_in_archive:nn{#1}{\__stex_inputs_bibresource:n{#2}}
1964   }
1965 }

```

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

### \IfInputref

```

1966 \stex_if_html_backend:TF{
1967   \newcommand \IfInputref[2]{
1968     \stex_annotate:nn{shtml:ifinputref=true}{#1}
1969     \stex_annotate:nn{shtml:ifinputref=false}{#2}
1970   }
1971 }{
1972   \newcommand \IfInputref[2]{
1973     \ifinputref #1 \else #2 \fi
1974   }
1975 }

```

(End of definition for \IfInputref. This function is documented on page 83.)

### \libinput

```

1976 \cs_new_protected:Nn \__stex_inputs_up_archive:nn {
1977   \prop_if_exist:NF \l_stex_current_archive_prop {
1978     \msg_error:nnn{stex}{error/notinarchive}\libinput
1979   }
1980   \prop_get:NnNF \l_stex_current_archive_prop {id} \l__stex_inputs_id_str {
1981     \msg_error:nnn{stex}{error/notinarchive}\libinput
1982   }
1983   \seq_clear:N \l__stex_inputs_libinput_files_seq
1984   \seq_set_eq:NN \l__stex_inputs_path_seq \c_stex_mathhub_file
1985   \seq_set_split:NnV \l__stex_inputs_id_seq / \l__stex_inputs_id_str
1986
1987   \bool_while_do:nn { ! \seq_if_empty_p:N \l__stex_inputs_id_seq }{
1988     \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / meta-
1989     \IfFileExists{ \l__stex_inputs_path_str }{
1990       \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_s
1991       \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
1992     }
1993   }{ }
1994   \seq_pop_left:NN \l__stex_inputs_id_seq \l__stex_inputs_path_str
1995   \seq_put_right:No \l__stex_inputs_path_seq \l__stex_inputs_path_str
1996 }
1997
1998 \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / lib / #1
1999 \IfFileExists{ \l__stex_inputs_path_str }{
2000   \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
2001   \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
2002 }

```

```

2003   }{ }
2004 }
2005
2006 \cs_new_protected:Nn \__stex_inputs_libinput:n {
2007   \__stex_inputs_up_archive:nn{#1}{tex}
2008   \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
2009     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
2010   }{
2011     \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
2012       \input{ ##1 }
2013     }
2014   }
2015 }
2016
2017 \newcommand \libinput [2] [] {
2018   \tl_if_empty:nTF{#1}{
2019     \__stex_inputs_libinput:n{#2}
2020   }{
2021     \stex_in_archive:nn{#1}{\__stex_inputs_libinput:n{#2}}
2022   }
2023 }

```

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

### `\libusepackage`

```

2024 \newcommand\libusepackage[2] [] {
2025   \__stex_inputs_up_archive:nn{#2}{sty}
2026   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2027     \str_set:Nx \l__stex_inputs_tmp_str {\seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
2028     \exp_args:Nne \use:n {\usepackage[#1]} {
2029       \str_range:Nnn\l__stex_inputs_tmp_str 1 {-5}
2030     }
2031   }{
2032     \stex_fatal_error:nnn{error/nofile}{\libusepackage}{#1.sty}
2033   }
2034 }

```

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

### `\mhgraphics`

#### `\cmhgraphics`

#### `\lstinputmhlising`

#### `\clstinputmhlising`

#### `\mhtikzinput`

#### `\cmhtikzinput`

```

2035 \str_new:N \l__stex_inputs_gin_repo_str
2036 \ltx@ifpackageloaded{graphicx}{\use:n}{\AtEndOfPackageFile{graphicx}}{
2037   \define@key{Gin}{archive}{
2038     \tl_set:Nx\Gin@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2039   }
2040   \providecommand\mhgraphics[2] [] {
2041     \tl_set:Nx\Gin@mhrepos{
2042       \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id}
2043     }
2044     \setkeys{Gin}{#1}
2045     \includegraphics[#1]{ \Gin@mhrepos #2 }
2046   }
2047   \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
2048 }
2049

```

```

2050 \ltx@ifpackageloaded{listings}{\use:n}{\AtEndOfPackageFile{listings}}{
2051   \define@key{lst}{archive}{
2052     \def\lst@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2053   }
2054   \newcommand\lstinputmhlisting[2] []{%
2055     \def\lst@mhrepos{
2056       \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id
2057     }
2058     \setkeys{lst}{#1}%
2059     \lstinputlisting[#1]{\lst@mhrepos #2}}
2060   \newcommand\clstinputmhlisting[2] []{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
2061 }
2062
2063 \ltx@ifpackageloaded{tikzinput}{\use:n}{\AtEndOfPackageFile{tikzinput}}{
2064   \define@key{Gin}{archive}{
2065     \str_set:Nn \l__stex_inputs_gin_repo_str {#1}
2066     \def\Gin@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2067   }
2068   \newcommand\mhtikzinput[2] []{%
2069     \str_clear:N \l__stex_inputs_gin_repo_str
2070     \def\Gin@mhrepos{
2071       \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id
2072     }
2073     \setkeys{Gin}{#1}%
2074     \exp_args:No \stex_in_archive:nn \l__stex_inputs_gin_repo_str {
2075       \tikzinput[#1]{\Gin@mhrepos #2}
2076     }
2077   }
2078   \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
2079 }

```

(End of definition for \mhgraphics and others. These functions are documented on page 83.)

### \libusetikzlibrary

```

2080 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary_i:nn {
2081   \pgfkeys@spdef\pgf@temp{#1}
2082   \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
2083   \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfutil
2084   \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
2085   \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
2086   \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
2087   \catcode'\@=11
2088   \catcode'\|=12
2089   \catcode'\$=3
2090   \pgfutil@InputIfFileExists{#2}{-}{-}
2091   \catcode'\@=\csname tikz@library@#1@atcode\endcsname
2092   \catcode'\|=\csname tikz@library@#1@barcode\endcsname
2093   \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
2094 }
2095
2096 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary:n{
2097   \__stex_inputs_up_archive:nn{tikzlibrary#1}{code.tex}
2098   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2099     \exp_args:Nne \__stex_inputs_usetikzlibrary_i:nn{#1}{ \seq_item:Nn \l__stex_inputs_libin

```

```

2100   }{
2101     \stex_fatal_error:nnn{error/nofile}{\libusetikzlibrary}{tikzlibrary#1.code.tex}
2102   }
2103 }
2104
2105 \newcommand \libusetikzlibrary [2] [] {
2106   \cs_if_exist:NF \usetikzlibrary {
2107     \msg_error:nnx{stex}{error/notikz}{\tl_to_str:n{\libusetikzlibrary}}
2108   }
2109   \tl_if_empty:nTF{#1}{
2110     \__stex_inputs_usetikzlibrary:n{#2}
2111   }{
2112     \stex_in_archive:nn{#1}{\__stex_inputs_usetikzlibrary:n{#2}}
2113   }
2114 }

```

(End of definition for \libusetikzlibrary. This function is documented on page 118.)

## 13.5 SMS Mode

```

2115 <@@=stex_smsmode>
      Macros and environments allowed in sms mode:
2116 \tl_new:N \g__stex_smsmode_allowed_tl
2117 \tl_new:N \g__stex_smsmode_allowed_escape_tl
2118 \seq_new:N \g__stex_smsmode_allowedenvs_seq
      \stex_sms_allow:N
\stex_sms_allow_escape:N
\stex_sms_allow_env:n
2119 \cs_new_protected:Nn \stex_sms_allow:N {
2120   \tl_gput_right:Nn \g__stex_smsmode_allowed_tl {#1}
2121 }
2122
2123 \cs_new_protected:Nn \stex_sms_allow_escape:N {
2124   \tl_gput_right:Nn \g__stex_smsmode_allowed_escape_tl {#1}
2125 }
2126
2127 \cs_new_protected:Nn \stex_sms_allow_env:n {
2128   \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowedenvs_seq {\tl_to_str:n{#1}}
2129 }

```

(End of definition for \stex\_sms\_allow:N, \stex\_sms\_allow\_escape:N, and \stex\_sms\_allow\_env:n. These functions are documented on page 135.)

Some initial allowed macros:

```

2130 \stex_sms_allow:N \makeatletter
2131 \stex_sms_allow:N \makeatother
2132 \stex_sms_allow:N \ExplSyntaxOn
2133 \stex_sms_allow:N \ExplSyntaxOff
2134 \stex_sms_allow:N \rustexBREAK
      \stex_if_smsmode_p:
\stex_if_smsmode:TF
2135 \bool_new:N \g__stex_smsmode_bool
2136 \bool_set_false:N \g__stex_smsmode_bool
2137 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2138   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2139 }

```



(End of definition for `\stex_if_smsmode:TF`. This function is documented on page 135.)

`\stex_sms_allow_import:Nn`

`\stex_sms_allow_import_env:nn`

```

2140 \tl_new:N \g__stex_smsmode_allowed_import_tl
2141 \seq_new:N \g__stex_smsmode_allowed_import_env_seq
2142 \cs_new_protected:Nn \stex_sms_allow_import:Nn {
2143   \tl_gput_right:Nn \g__stex_smsmode_allowed_import_tl {#1}
2144   \tl_gset:cn{\tl_to_str:n{#1}---smsmode} {#2}
2145 }
2146 \cs_new_protected:Nn \stex_sms_allow_import_env:nn {
2147   \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowed_import_env_seq {\tl_to_str:n{#1}}
2148   \tl_gset:cn{\tl_to_str:n{#1}---env---smsmode} {#2}
2149 }
2150
2151 \tl_new:N \g_stex_sms_import_code

```

(End of definition for `\stex_sms_allow_import:Nn` and `\stex_sms_allow_import_env:nn`. These functions are documented on page 136.)

`\stex_file_in_smsmode:nn`

`\stex_file_in_smsmode:on`

```

2152 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:n { \stex_suppress_html:n {
2153   \vbox_set:Nn \l_tmpa_box {
2154     \bool_set_true:N \g__stex_smsmode_bool
2155     \bool_set_false:N \stex_html_do_output_bool
2156     #1
2157   }
2158   %\box_clear:N \l_tmpa_box
2159 } }
2160
2161 \quark_new:N \q__stex_smsmode_break
2162
2163 \cs_new_protected:Nn \__stex_smsmode_start_smsmode:n {
2164   \everyeof{\q__stex_smsmode_break\exp_not:N}
2165   \let\stex_smsmode_do:\__stex_smsmode_smsmode_do:
2166   \exp_after:wN \exp_after:wN \exp_after:wN
2167   \stex_smsmode_do:
2168   \cs:w @ @ input\cs_end: "#1" \relax
2169 }
2170
2171 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2172   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2173   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2174   \tl_clear:N \g_stex_sms_import_code
2175   \group_begin:
2176     \let \l_stex_metatheory_uri \c_stex_default_metatheory
2177     \cs_set:Npn \stex_check_term:n ##1 {}
2178     \seq_clear:N \l_stex_all_modules_seq
2179     \str_clear:N \l_stex_current_module_str
2180     #2
2181     \stex_filestack_push:n{#1}
2182     \__stex_smsmode_in_smsmode:n {
2183       \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_imports:N
2184       \tl_map_inline:Nn \g__stex_smsmode_allowed_import_tl {
2185         \use:c{\tl_to_str:n{##1}---smsmode}

```

```

2186     }
2187     \seq_map_inline:Nn \g__stex_smsmode_allowed_import_env_seq {
2188       \use:c{\tl_to_str:n{##1}~---env~---smsmode}
2189     }
2190     \__stex_smsmode_start_smsmode:n{#1}
2191   }
2192   \__stex_smsmode_in_smsmode:n \g_stex_sms_import_code
2193   \__stex_smsmode_in_smsmode:n {
2194     \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_normal:N
2195     \__stex_smsmode_start_smsmode:n{#1}
2196   }
2197   \stex_filestack_pop:
2198 \group_end:
2199 }
2200 \cs_generate_variant:Nn \stex_file_in_smsmode:nn {on}

```

(End of definition for `\stex_file_in_smsmode:nn`. This function is documented on page 135.)

`\stex_smsmode_do:`

```

2201 \cs_new_protected:Nn \__stex_smsmode_smsmode_do: {
2202   %\stex_if_smsmode:T {
2203     \__stex_smsmode_do:w
2204   %}
2205 }
2206 \let\stex_smsmode_do:\relax
2207
2208
2209 \cs_new:Nn \__stex_smsmode_check_cs:NNn {
2210   \exp_after:wN\if\exp_after:wN\relax\exp_not:N#3
2211   \exp_after:wN#1\exp_after:wN#3\else
2212   \exp_after:wN#2\fi
2213 }
2214
2215 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2216   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2217     \__stex_smsmode_check_cs:NNn \__stex_smsmode_do_aux:N \__stex_smsmode_do:w { #1 }
2218   }{
2219     \__stex_smsmode_do:w
2220   }
2221 }
2222
2223 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2224   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
2225     \__stex_smsmode_do_aux_curr:N #1
2226   }
2227 }
2228
2229 \cs_new_protected:Nn \__stex_smsmode_do_aux_imports:N {
2230   % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~import}
2231   \tl_if_in:NnTF \g__stex_smsmode_allowed_import_tl {#1} {
2232     \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}~in~import}
2233     #1
2234   }{
2235     \cs_if_eq:NNTF \begin #1 {

```

```

2236     \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowed_import_env_seq
2237   }{
2238     \cs_if_eq:NNTF \end #1 {
2239       \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowed_import_env_seq
2240     }{
2241       \__stex_smsmode_do:w
2242     }
2243   }
2244 }
2245 }
2246
2247 \cs_new_protected:Nn \__stex_smsmode_do_aux_normal:N {
2248   % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~sms~mode}
2249   \tl_if_in:NnTF \g__stex_smsmode_allowed_tl {#1} {
2250     \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}}
2251     #1\__stex_smsmode_do:w
2252   }{
2253     \tl_if_in:NnTF \g__stex_smsmode_allowed_escape_tl {#1} {
2254       \stex_debug:nn{sms}{Executing~escaped~\tl_to_str:n{#1}}
2255       #1
2256     }{
2257       \cs_if_eq:NNTF \begin #1 {
2258         \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowedenvs_seq
2259       }{
2260         \cs_if_eq:NNTF \end #1 {
2261           \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowedenvs_seq
2262         }{
2263           \__stex_smsmode_do:w
2264         }
2265       }
2266     }
2267   }
2268 }
2269
2270 \cs_new_protected:Nn \__stex_smsmode_check_begin:Nn {
2271   % \stex_debug:nn{sms}{Checking~environment~#2}
2272   \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2273     \stex_debug:nn{sms}{Environment~#2}
2274     \begin{#2}
2275   }{
2276     \__stex_smsmode_do:w
2277   }
2278 }
2279 \cs_new_protected:Nn \__stex_smsmode_check_end:Nn {
2280   % \stex_debug:nn{sms}{Checking~end~environment~#2}
2281   \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2282     \stex_debug:nn{sms}{End~Environment~#2}
2283     \end{#2}\__stex_smsmode_do:w
2284   }{
2285     %\str_if_eq:nnTF{#2}{document} \endinput
2286     \__stex_smsmode_do:w
2287   }
2288 }

```

(End of definition for `\stex_smsmode_do:`. This function is documented on page 136.)

## 13.6 Modules

### 13.6.1 The smodule-environment

```
2289 <@@=stex_modules>
```

`\l_stex_current_module_str` The current module:

```
2290 \str_new:N \l_stex_current_module_str
```

(End of definition for `\l_stex_current_module_str`. This variable is documented on page 122.)

`\l_stex_all_modules_seq` Stores all modules currently in scope

```
2291 \seq_new:N \l_stex_all_modules_seq
```

(End of definition for `\l_stex_all_modules_seq`. This variable is documented on page 122.)

`\stex_every_module:n`

```
2292 <@@=stex_module_setup>
```

```
2293 \tl_clear:N \g_stex_every_module_tl {
```

```
2294 }
```

```
2295 \cs_new_protected:Nn \stex_every_module:n {
```

```
2296   \tl_gput_right:Nn \g_stex_every_module_tl { #1 }
```

```
2297 }
```

(End of definition for `\stex_every_module:n`. This function is documented on page 122.)

`\stex_module_setup:n` Sets up a new module:

```
2298 \cs_new_protected:Npn \stex_module_setup:n {
```

```
2299   \stex_if_in_module:TF \__stex_module_setup_setup_nested:n \__stex_module_setup_setup_top:n
```

```
2300 }
```

```
2301
```

```
2302 \cs_new_protected:Nn \__stex_module_setup_setup_top:n {
```

```
2303   \__stex_module_setup_get_uri_str:n{#1}
```

```
2304   \stex_debug:nn{module}{Module~URI:~\l__stex_module_setup_ns_str?#1}
```

```
2305   \str_if_empty:NTF \l_stex_key_sig_str
```

```
2306   \stex_module_setup_top_nosig:n \__stex_module_setup_setup_top_sig:n {\l__stex_module_setu
```

```
2307   \stex_metagroup_new:o \l_stex_current_module_str
```

```
2308   \g_stex_every_module_tl
```

```
2309   \stex_execute_in_module:x {
```

```
2310     \stex_do_deprecation:n{#1}
```

```
2311   }
```

```
2312   \__stex_module_setup_load_meta:
```

```
2313 }
```

```
2314
```

```
2315 \cs_new_protected:Nn \stex_module_setup_top_nosig:n {
```

```
2316   \stex_if_module_exists:nTF{#1}{
```

```
2317     \stex_debug:nn{modules}{(already exists)}
```

```
2318   }{
```

```
2319     \tl_gclear:c{c_stex_module_ #1 _code}
```

```
2320     \prop_gclear:c{c_stex_module_ #1 _morphisms_prop }
```

```
2321     \prop_gclear:c{c_stex_module_ #1 _symbols_prop }
```

```
2322     \prop_gclear:c{c_stex_module_ #1 _notations_prop }
```

```
2323   }
```

```
2324   \str_set:Nx \l_stex_current_module_str {#1}
```

```
2325   \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
```

```

2326 }
2327
2328 \cs_new_protected:Nn \__stex_module_setup_setup_top_sig:n {
2329   \stex_if_module_exists:nTF{#1}{
2330     \stex_debug:nn{modules}{(already exists)}
2331   }{
2332     \stex_debug:nn{modules}{(needs loading)}
2333     \__stex_module_setup_load_sig:
2334   }
2335   %\stex_if_smsmode:F { % WHY?
2336   \stex_activate_module:x {
2337     #1
2338   }
2339   %}
2340   \str_set:Nx\l_stex_current_module_str{#1}
2341 }
2342
2343 \cs_new_protected:Nn \__stex_module_setup_load_sig: {
2344   \stex_file_split_off_ext:NN \l__stex_module_setup_sigfile \g_stex_current_file
2345   \stex_file_split_off_lang:NN \l__stex_module_setup_sigfile \l__stex_module_setup_sigfile
2346   \exp_args:Ne \stex_file_in_smsmode:nn {
2347     \stex_file_use:N \l__stex_module_setup_sigfile . \l_stex_key_sig_str . tex
2348   }{}
2349 }
2350
2351 \cs_new_protected:Nn \__stex_module_setup_setup_nested:n {
2352   \exp_after:wN
2353     \__stex_module_setup_split_module:n \l_stex_current_module_str \__stex_module_setup_end:
2354   \stex_debug:nn{module}{Nested~Module~URI:~\l_stex_current_module_str}
2355   \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
2356   \stex_metagroup_new:o \l_stex_current_module_str
2357 }
2358
2359
2360 \cs_new_protected:Nn \__stex_module_setup_get_uri_str:n {
2361   \str_clear:N \l__stex_module_setup_ns_str
2362   \stex_map_uri:Nnnnn \l_stex_current_ns_uri {
2363     \str_set:Nx \l__stex_module_setup_ns_str{##1\c_colon_str/}
2364   }{
2365     \seq_set_split:Nnn \l__stex_module_setup_seq / {##1}
2366     \seq_pop_right:NN \l__stex_module_setup_seq \l__stex_module_setup_seg
2367     \exp_args:No \str_if_eq:nnF \l__stex_module_setup_seg {#1} {
2368       \seq_put_right:No \l__stex_module_setup_seq \l__stex_module_setup_seg
2369     }
2370     \tl_put_right:Nx \l__stex_module_setup_ns_str {\seq_use:Nn \l__stex_module_setup_seq /}
2371   }{}{}
2372 }
2373
2374 \cs_new_protected:Npn \__stex_module_setup_split_module:n #1?#2 \__stex_module_setup_end: #3
2375   \stex_module_setup_top_nosig:n { #1 ? #2 / #3}
2376 }
2377
2378 \bool_new:N \l_stex_in_meta_bool
2379 \bool_set_false:N \l_stex_in_meta_bool

```

```

2380
2381 \cs_new_protected:Nn \__stex_module_setup_load_meta: {
2382   \tl_if_empty:NF \l_stex_metatheory_uri {
2383     \stex_execute_in_module:x{
2384       \stex_pseudogroup_with:nn{\l_stex_in_meta_bool}{
2385         \stex_activate_module:n {\stex_uri_use:N \l_stex_metatheory_uri }
2386       }
2387     }
2388   }
2389 }
2390
2391 <@@=stex_modules>

```

(End of definition for \stex\_module\_setup:n. This function is documented on page 122.)

**\stex\_close\_module:**

```

2392 \cs_new:Nn \stex_close_module: {
2393   \bool_if:NT \c_stex_persist_write_mode_bool \__stex_modules_persist_module:
2394   \stex_debug:nn{module}{
2395     Closing~module~\l_stex_current_module_str^^J
2396     Code:~\expandafter\meaning\csname c_stex_module_\l_stex_current_module_str_code\endcsname
2397     Imports:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_str_
2398     Declarations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_
2399     Notations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_str
2400   }
2401 }
2402
2403 \cs_new_protected:Nn \__stex_modules_persist_module: {
2404   \stex_persist:e {
2405     \__stex_modules_restore_module:nnnn {\l_stex_current_module_str}{
2406       \exp_after:wN \prop_to_keyval:N \cs:w
2407         c_stex_module_\l_stex_current_module_str_morphisms_prop
2408       \cs_end:
2409     }{
2410       \exp_after:wN \prop_to_keyval:N \cs:w
2411         c_stex_module_\l_stex_current_module_str_symbols_prop
2412       \cs_end:
2413     }{
2414       \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
2415       \exp_after:wN \exp_after:wN \exp_after:wN
2416       { \cs:w c_stex_module_\l_stex_current_module_str_code \cs_end: }
2417     }{}
2418     \prop_map_function:cN{c_stex_module_\l_stex_current_module_str_notations_prop}
2419     \__stex_modules_persist_notsi:nn
2420     \exp_not:N \STEXRestoreNotsiEnd {}
2421   }
2422 }
2423
2424 \cs_new_protected:Nn \__stex_modules_restore_module:nnnn {
2425   \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_morphisms_prop}{#2}
2426   \cs_set:Npn \__stex_modules_tl {#3}
2427   \exp_args:Nno \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}\__stex_
2428   \prop_map_inline:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}{
2429     \stex_ref_new_symbol:n{#1?#1}

```

```

2430 }
2431 \cs_gset:cpn{c_stex_module_\tl_to_str:n{#1}_code}{#4}
2432 \prop_gclear:c{c_stex_module_\tl_to_str:n{#1}_notations_prop}
2433 \str_set:Nn \l__stex_modules_restore_mod_str {#1}
2434 \group_begin:
2435   \catcode'_=8\relax
2436   \catcode':=12\relax
2437   \__stex_modules_restore_not:n
2438 }
2439
2440 \cs_new:Nn \__stex_modules_persist_not:n {
2441   \exp_not:n{#2}
2442 }
2443
2444 \quark_new:N \STEXRestoreNotEnd
2445
2446 \cs_new_protected:Nn \__stex_modules_restore_not:n {
2447   \__stex_modules_restore_not:n_i:n
2448 }
2449
2450 \cs_new_protected:Nn \__stex_modules_restore_not:n_i:n {
2451   \tl_if_eq:nnTF{#1}{\STEXRestoreNotEnd}{
2452     \group_end:
2453   }{
2454     \__stex_modules_restore_not_ii:nnnnn {#1}
2455   }
2456 }
2457
2458 \cs_new_protected:Nn \__stex_modules_restore_not_ii:nnnnn {
2459   \cs_set:Npn \l__stex_modules_tl_{{#4}{#5}}
2460   \exp_args:NNe\use:nn\prop_gput:cnn{
2461     {c_stex_module_\l__stex_modules_restore_mod_str_notations_prop}
2462     {\tl_to_str:n{#1!#2}}{
2463       {\tl_to_str:n{#1}}{\tl_to_str:n{#2}}{#3}
2464       \exp_args:No \exp_not:n \l__stex_modules_tl
2465     }
2466   }
2467   \__stex_modules_restore_not:n_i:n
2468 }

```

(End of definition for `\stex_close_module:`. This function is documented on page 122.)

`\l_stex_metatheory_uri`

```

2469 \tl_new:N \l_stex_metatheory_uri

```

(End of definition for `\l_stex_metatheory_uri`. This variable is documented on page ??.)

`\setmetatheory`

```

2470 \cs_new_protected:Nn \__stex_modules_set_metatheory:nn {
2471   \group_begin:
2472     \stex_debug:nn{metatheory}{Setting~metatheory~[#1]#2}
2473     \stex_import_module_uri:nn { #1 } { #2 }
2474     \stex_debug:nn{metatheory}{Here^^J
2475       \l_stex_import_archive_str^^J
2476       \l_stex_import_path_str^^J

```

```

2477     \l_stex_import_name_str^^J
2478   }
2479   \stex_import_require_module:ooo
2480     \l_stex_import_archive_str
2481     \l_stex_import_path_str
2482     \l_stex_import_name_str
2483     \stex_debug:nn{metatheory}{Found:~\l_stex_import_ns_str}
2484     \exp_args:Nne \use:nn {
2485       \group_end: \stex_uri_resolve:Nn \l_stex_metatheory_uri
2486     }{\l_stex_import_ns_str}
2487   }
2488
2489   \NewDocumentCommand \setmetatheory {0{ } m}{
2490     \__stex_modules_set_metatheory:nn { #1 }{ #2 }
2491     \stex_smsmode_do:
2492   }
2493   \stex_sms_allow_escape:N \setmetatheory

```

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

Keys and key handling:

```

2494 \stex_keys_define:nnnn{smodule}{
2495   \str_clear:N \l_stex_key_sig_str
2496 }{
2497   meta          .code:n      = {
2498     \str_if_empty:nTF {#1}{
2499       \tl_clear:N \l_stex_metatheory_uri
2500     }{
2501       \stex_uri_resolve:Nx \l_stex_metatheory_uri { #1 }
2502     }
2503   },
2504   ns            .code:n      = {
2505     \stex_uri_resolve:Nx \l_stex_current_ns_uri { #1 }
2506   } ,
2507   lang          .code:n      = {
2508     \stex_set_language:n { #1 }
2509   } ,
2510   sig           .str_set_x:N  = \l_stex_key_sig_str ,
2511   creators      .code:n      = {} , % todo ?
2512   contributors  .code:n      = {} , % todo ?
2513   srccite       .code:n      = {} % todo ?
2514 }{id, title, style, deprecate}

```

smodule (env.)

```

2515 \stex_new_stylable_env:nnnnnn {module} {0{ } m}{
2516   \stex_keys_set:nn { smodule }{ #1 }
2517   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
2518   \tl_if_empty:NF \thistitle {
2519     \exp_args:No \stexdoctitle \thistitle
2520   }
2521   \exp_args:Nx \stex_module_setup:n { \tl_to_str:n{ #2 } }
2522
2523   \stex_if_do_html:T {
2524     \exp_args:Nne \begin{stex_annotate_env} {
2525       shtml:theory={\l_stex_current_module_str},

```



```

2526     shtml:language={ \l_stex_current_language_str},
2527     shtml:signature={\l_stex_key_sig_str}
2528     \tl_if_empty:NF \l_stex_metatheory_uri {,
2529         shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
2530     }
2531 }
2532 \stex_annotate_invisible:n{}
2533 }
2534 \stex_if_smsmode:F {
2535     \str_set_eq:NN \thismoduleuri \l_stex_current_module_str
2536     \tl_set:Nn \thismodulename {#2}
2537     \stex_style_apply:
2538 }
2539 \stex_smsmode_do:
2540 ){
2541     \stex_close_module:
2542     \stex_if_smsmode:F \stex_style_apply:
2543     \stex_if_do_html:T{ \end{stex_annotate_env} }
2544 }{}{}{s}
2545
2546 \stex_sms_allow_env:n{smodule}

```

`\stex_if_in_module_p:` Are we currently in a module?

```

\stex_if_in_module:TF
2547 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
2548     \str_if_empty:NTF \l_stex_current_module_str
2549     \prg_return_false: \prg_return_true:
2550 }

```

*(End of definition for \stex\_if\_in\_module:TF. This function is documented on page 122.)*

`\stex_if_module_exists_p:n` Does a module with this URI exist?

```

\stex_if_module_exists:nTF
2551 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
2552     \tl_if_exist:cTF { c_stex_module_#1_code }
2553     \prg_return_true: \prg_return_false:
2554 }

```

*(End of definition for \stex\_if\_module\_exists:nTF. This function is documented on page 122.)*

`\stex_do_up_to_module:n` Execute code in the current module (i.e. as if the `\begin{smodule}` was the current  
`\stex_do_up_to_module:x` tex group)

```

2555 \cs_new_protected:Nn \stex_do_up_to_module:n {
2556     \exp_args:No \stex_metagroup_do_in:n \l_stex_current_module_str {#1}
2557 }
2558 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}

```

*(End of definition for \stex\_do\_up\_to\_module:n. This function is documented on page 123.)*

`\stex_module_add_code:n`

```

\stex_module_add_code:x
2559 \cs_new_protected:Nn \stex_module_add_code:n {
2560     \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
2561 }
2562 \cs_generate_variant:Nn \stex_module_add_code:n {x}

```

*(End of definition for \stex\_module\_add\_code:n. This function is documented on page 123.)*

`\stex_execute_in_module:n`

`\stex_execute_in_module:x`

```
2563 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:TF {
2564   \stex_module_add_code:n { #1 }
2565   \stex_do_up_to_module:n { #1 }
2566 }{ #1 }}
2567 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
```

(End of definition for `\stex_execute_in_module:n`. This function is documented on page 122.)

`\STEXexport`

```
2568 \NewDocumentCommand \STEXexport {} {
2569   \ExplSyntaxOn
2570   \__stex_modules_export:n
2571 }
2572 \cs_new_protected:Nn \__stex_modules_export:n {
2573   \stex_ignore_spaces_and_pars:#1\ExplSyntaxOff
2574   \stex_module_add_code:n { \stex_ignore_spaces_and_pars:#1}
2575   \stex_smsmode_do:
2576 }
```

Only allowed in modules, and allowed (escaped) in sms mode:

```
2577 \stex_deactivate_macro:Nn \STEXexport {module-environments}
2578 \stex_sms_allow_escape:N \STEXexport
2579 \stex_every_module:n {\stex_reactivate_macro:N \STEXexport}
```

(End of definition for `\STEXexport`. This function is documented on page 86.)

`\stex_module_add_morphism:nmn`

`\stex_module_add_morphism:nonn`

`\stex_module_add_morphism:ooox`

```
2580 \cs_new_protected:Nn \stex_module_add_morphism:nmn {
2581   \exp_args:Nne \prop_gput:cnx{c_stex_module_\l_stex_current_module_str _morphisms_prop}{
2582     \tl_if_empty:nTF{#1}{[#2]}{#1}
2583   }{#1}{#2}{#3}{#4}}
2584 }
2585 \cs_generate_variant:Nn \stex_module_add_morphism:nmn {nonn,ooox}
```

(End of definition for `\stex_module_add_morphism:nmn`. This function is documented on page 130.)

`\stex_module_add_symbol:nnnnnN`

**#1** :  $\langle \text{Macro name} \rangle$

**#2** :  $\langle \text{Name} \rangle$

**#3** :  $\langle \text{arity} \rangle$

**#4** :  $\{(\langle \text{Arg num} \rangle)\langle \text{Arg str} \rangle\}^*$

**#5** : Definiens

**#6** : type

**#7** : Return

**#8** : Command

```
2586 \cs_new_protected:Nn \stex_module_add_symbol:nnnnnN {
2587   \stex_debug:nn{declaration}{New-declaration:~\l_stex_current_module_str?#2^^J
2588     Macro:#1^^JArity:#3~(#4)^^J
2589     Def:~\tl_to_str:n{#5}^^J
2590     Type:~\tl_to_str:n{#6}^^J
2591     Returns:~\tl_to_str:n{#7}
2592 }
2593 %\prop_gput:cnx{c_stex_module_\l_stex_current_module_str _symbols_prop}
2594 %{#2}{\exp_not:n{#1}{#2}{#3}{#4}{#5}{#6}}{#7}\exp_not:n{#8}}
```

```

2595 \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
2596 {#2}{#{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}
2597 \tl_if_empty:nF{#1}{
2598   \stex_execute_in_module:n {
2599     \__stex_modules_activate_sym:n {#2}
2600   }
2601 }
2602 }
2603
2604 \cs_new_protected:Nn \__stex_modules_activate_sym:n {
2605   \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _symbols_prop}{
2606     \str_if_eq:nnT{#1}{##1}{
2607       \__stex_modules_activate_i:nnnnnnn ##2
2608     }
2609   }
2610 }
2611 \cs_new_protected:Nn \__stex_modules_activate_i:nnnnnnn {
2612   \stex_debug:nn{activating}{#1:\l_stex_current_module_str^^J
2613     \tl_to_str:n{#{#2}{#3}{#4}{#5}{#6}{#7}{#8}}
2614   }
2615   \cs_set:cpx{#1} {
2616     \stex_invoke_symbol:nnnnnnN
2617     {\l_stex_current_module_str}
2618     \exp_not:n{#{#2}{#3}{#4}{#5}{#6}{#7}{#8}}
2619   }
2620   \stex_debug:nn{activating}{done}
2621   \prop_map_break:
2622 }

```

(End of definition for `\stex_module_add_symbol:nnnnnN`. This function is documented on page ??.)

```

\stex_module_add_notation:nnnnn #1 : URI
\stex_module_add_notation:eoeeo #2 : variant
\stex_set_notation_macro:nnnnn #3 : arity
                                #4 : macro body
                                #5 : op

```

```

2623 \cs_new_protected:Nn \stex_module_add_notation:nnnnn {
2624   \stex_debug:nn{notations}{Adding~notation:^^J
2625     #1~\c_hash_str#2~#3^^J
2626     to~\l_stex_current_module_str
2627   }
2628   \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _notations_prop}
2629   {#1!#2}{#{#1}{#2}{#3}{#4}{#5}}
2630   \stex_execute_in_module:n {
2631     \__stex_modules_activate_not:nn{#1}{#2}
2632   }
2633 }
2634 \cs_generate_variant:Nn \stex_module_add_notation:nnnnn {eoeeo}
2635
2636
2637 \cs_new_protected:Nn \__stex_modules_activate_not:nn {
2638   \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _notations_prop}{
2639     \str_if_eq:nnT{#1!#2}{##1}{
2640       \prop_map_break:n{\stex_set_notation_macro:nnnnn ##2 }

```

```

2641     }
2642   }
2643 }

```

(End of definition for `\stex_module_add_notation:nnnnn` and `\stex_set_notation_macro:nnnnn`. These functions are documented on page 125.)

`\stex_set_notation_macro:nnnnn`

`\stex_set_notation_macro:eoexo`

```

2644 \cs_new_protected:Nn \stex_set_notation_macro:nnnnn {
2645   \tl_set:cn {l_stex_notation_#1_#2_cs}{#4}
2646   \cs_if_exist:cF{l_stex_notation_#1_cs}{
2647     \tl_set:cn {l_stex_notation_#1_cs}{#4}
2648   }
2649   \tl_if_empty:nF{#5}{
2650     \tl_set:cn{l_stex_notation_#1_op_#2_cs}{#5}
2651     \cs_if_exist:cF{l_stex_notation_#1_op_cs}{
2652       \cs_set_eq:cc {l_stex_notation_#1_op_cs}{l_stex_notation_#1_op_#2_cs}
2653     }
2654   }
2655 }
2656 \cs_generate_variant:Nn \stex_set_notation_macro:nnnnn {eoexo}

```

(End of definition for `\stex_set_notation_macro:nnnnn`. This function is documented on page 126.)

`\stex_activate_module:n`

`\stex_activate_module:o`

`\stex_activate_module:x`

```

2657 \cs_new_protected:Nn \stex_activate_module:n {
2658   \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
2659     \stex_debug:nn{modules}{Activating-module~#1^J\expandafter\meaning\csname c_stex_module
2660     \seq_put_right:Nn \l_stex_all_modules_seq { #1 }
2661     \stex_pseudogroup:nn{
2662       \str_set:Nn \l_stex_current_module_str {#1}
2663       \use:c{ c_stex_module_#1_code }
2664     }{
2665       \stex_pseudogroup_restore:N \l_stex_current_module_str
2666     }
2667   }
2668 }
2669 \cs_generate_variant:Nn \stex_activate_module:n {o,x}

```

(End of definition for `\stex_activate_module:n`. This function is documented on page 122.)

Iterating:

```

2670 <@@=stex_iterate>

```

`\stex_iterate_symbols:n`

```

2671 \cs_new_protected:Nn \stex_iterate_symbols:n {
2672   \stex_pseudogroup_with:nn{\_stex_iterate_sym_cs:nnnnnnnN\stex_iterate_break:\stex_iterate
2673   \cs_set:Npn \_stex_iterate_sym_cs:nnnnnnnN
2674   ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
2675   \cs_set:Npn \stex_iterate_break: {
2676     \prop_map_break:n{\seq_map_break:}
2677   }
2678   \cs_set:Npn \stex_iterate_break:n ##1 {
2679     \prop_map_break:n{\seq_map_break:n{##1}}
2680   }

```

```

2681 \seq_map_inline:Nn \l_stex_all_modules_seq {
2682 \prop_map_inline:cn{c_stex_module_##1_symbols_prop}{
2683 \__stex_iterate_sym_cs:nnnnnnnN {##1} ###2
2684 }
2685 }
2686 }
2687 }

```

(End of definition for `\stex_iterate_symbols:n`. This function is documented on page 123.)

`\stex_iterate_symbols:nn`

```

2688 \cs_new_protected:Nn \stex_iterate_symbols:nn {
2689 \seq_clear:N \l__stex_iterate_mods_seq
2690 \stex_pseudogroup_with:nn{\__stex_iterate_sym_cs:nnnnnnnN}{
2691 \cs_set:Npn \__stex_iterate_sym_cs:nnnnnnnN
2692 ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #2 }
2693 \clist_map_function:nN {#1} \__stex_iterate_it_decl_i:n
2694 }
2695 }
2696
2697 \cs_new_protected:Nn \__stex_iterate_it_decl_i:n {
2698 \seq_if_in:NnF \l__stex_iterate_mods_seq {#1} {
2699 \seq_put_left:Nn \l__stex_iterate_mods_seq {#1}
2700 \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2701 \__stex_iterate_it_decl_check:nnnn ##2
2702 }
2703 \prop_map_inline:cn{c_stex_module_#1_symbols_prop}{
2704 \__stex_iterate_sym_cs:nnnnnnnN {#1} ##2
2705 }
2706 }
2707 }
2708 \cs_new_protected:Nn \__stex_iterate_it_decl_check:nnnn {
2709 \tl_if_empty:nT{#1}{
2710 \__stex_iterate_it_decl_i:n {#2}
2711 }
2712 }

```

(End of definition for `\stex_iterate_symbols:nn`. This function is documented on page 123.)

`\stex_iterate_notations:nn`

```

2713 \cs_new_protected:Nn \stex_iterate_notations:nn {
2714 \seq_clear:N \l__stex_iterate_mods_seq
2715 \stex_pseudogroup_with:nn{\__stex_iterate_not_cs:nnnnn}{
2716 \cs_set:Npn \__stex_iterate_not_cs:nnnnn
2717 ##1 ##2 ##3 ##4 ##5 { #2 }
2718 \clist_map_function:nN {#1} \__stex_iterate_it_not_i:n
2719 }
2720 }
2721
2722 \cs_new_protected:Nn \__stex_iterate_it_not_i:n {
2723 \seq_if_in:NnF \l__stex_iterate_mods_seq {#1} {
2724 \seq_put_left:Nn \l__stex_iterate_mods_seq {#1}
2725 \prop_map_inline:cn{c_stex_module_#1_notations_prop}{
2726 \__stex_iterate_not_cs:nnnnn ##2
2727 }

```

```

2728     \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2729       \__stex_iterate_it_not_check:nmmm ##2
2730     }
2731   }
2732 }
2733 \cs_new_protected:Nn \__stex_iterate_it_not_check:nmmm {
2734   \tl_if_empty:nT{#1}{
2735     \__stex_iterate_it_not_i:n {#2}
2736   }
2737 }

```

(End of definition for `\stex_iterate_notations:nn`. This function is documented on page 126.)

`\stex_iterate_morphisms:nn`

```

2738 \cs_new_protected:Nn \stex_iterate_morphisms:nn {
2739   \seq_clear:N \l__stex_iterate_mods_seq
2740   \bool_set_true:N \l__stex_iterate_continue_bool
2741   \cs_set:Npn \__stex_iterate_morphism_cs:nmmm ##1 ##2 ##3 ##4 ##5 {
2742     #2
2743     \bool_if:NT \l__stex_iterate_continue_bool {
2744       \str_if_eq:nnTF{##1}{##2}{
2745         \tl_put_right:Nn \l__stex_iterate_todo_tl {
2746           \__stex_iterate_iterate_morphism:nn{##5}{##2}
2747         }
2748       }{
2749         \tl_put_right:Nn \l__stex_iterate_todo_tl {
2750           \__stex_iterate_iterate_morphism:nn{##5 / ##1}{##2}
2751         }
2752       }
2753     }
2754   }
2755   \cs_set:Npn \stex_iterate_break:n ##1 {
2756     \bool_set_false:N \l__stex_iterate_continue_bool
2757     \prop_map_break:n{##1}
2758   }
2759   \__stex_iterate_iterate_morphism:nn{}{##1}
2760 }
2761
2762 \cs_new_protected:Nn \__stex_iterate_iterate_morphism:nn {
2763   \tl_clear:N \l__stex_iterate_todo_tl
2764   \seq_if_in:NnF \l__stex_iterate_mods_seq {#1 #2}{
2765     \seq_put_right:Nn \l__stex_iterate_mods_seq {#1 #2}
2766     \prop_map_inline:cn{c_stex_module_#2_morphisms_prop}{
2767       \__stex_iterate_morphism_cs:nmmm ##2 {#1}
2768       % TODO
2769       % ##1: name or [mpath]
2770       % ##2 = {####1}{####2}{####3}{####4}
2771       % ####1 = name
2772       % ####2 = mpath
2773       % ####3 = type
2774       % ####4 = {origname}{newname}*
2775     }
2776     \bool_if:NT \l__stex_iterate_continue_bool \l__stex_iterate_todo_tl
2777   }
2778 }

```

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

## 13.6.2 Structural Features

2779 `<@@=stex_features>`

`\stex_structural_feature_module:nn`  
`\stex_structural_feature_module_end:`

```

2780 \cs_new_protected:Nn \stex_structural_feature_module:nn {
2781   \stex_if_do_html:TF {
2782     \exp_args:Nne \begin{stex_annotate_env} {
2783       shtml:feature-#2={
2784         \l_stex_current_module_str/#1}
2785       \str_if_empty:NF \l_stex_macroname_str {,
2786         shtml:macroname={\l_stex_macroname_str}
2787       }
2788     }
2789     \stex_annotate_invisible:n{#1}
2790   }\group_begin:
2791   \stex_module_setup:n {#1-module}
2792 }
2793
2794 \cs_new_protected:Nn \stex_structural_feature_module_end: {
2795   \tl_gset_eq:NN \g_stex_last_feature_str \l_stex_current_module_str
2796   \stex_close_module:
2797   \stex_if_do_html:TF{
2798     \end{stex_annotate_env}
2799   }\group_end:
2800 }
```

(End of definition for `\stex_structural_feature_module:nn` and `\stex_structural_feature_module_end:`. These functions are documented on page 128.)

`\stex_structural_feature_morphism:nnn`  
`\stex_structural_feature_morphism_end:`

```

2801 \bool_new:N \l__stex_features_implicit_bool
2802 \cs_new_protected:Nn \stex_structural_feature_morphism:nnnn {
2803   \str_clear:N \l_stex_current_domain_str
2804   \tl_if_empty:nT{#3}{
2805     \stex_get_mathstructure:n{#4}
2806     \str_set_eq:NN \l_stex_current_domain_str \l_stex_get_structure_module_str
2807   }
2808   \str_if_empty:NT \l_stex_current_domain_str {
2809     \stex_import_module_uri:nn { #3 }{ #4 }
2810     \group_begin:
2811     \stex_import_require_module:ooo
2812     \l_stex_import_archive_str
2813     \l_stex_import_path_str
2814     \l_stex_import_name_str
2815     \exp_args:Nnx \use:nn \group_end: {
2816       \str_set:Nn \exp_not:N\l_stex_current_domain_str {\l_stex_import_ns_str}
2817     }
2818   }
2819   \tl_if_empty:nTF{#1}{
2820     \bool_set_true:N \l__stex_features_implicit_bool
2821     \str_set:Nx \l_tmpa_str {\l_stex_current_domain_str}
2822   }{
```

```

2823   \bool_set_false:N \l__stex_features_implicit_bool
2824   \str_set:Nn \l_tmpa_str {#1}
2825 }
2826
2827 \stex_if_do_html:TF {
2828   \begin{stex_annotate_env} {
2829     shtml:feature-#2={\l_stex_current_module_str?\l_tmpa_str},
2830     shtml:domain={\l_stex_current_domain_str}
2831     #5
2832   }
2833   \stex_annotate_invisible:n{}
2834 } \group_begin:
2835 \str_set:Nn \l__stex_features_feature_str {#2}
2836 \str_set_eq:NN \l_stex_feature_name_str \l_tmpa_str
2837 \__stex_features_setup:
2838 \__stex_features_reactivate:
2839 %^A\stex_activate_module:o \l_stex_current_domain_str
2840 \exp_args:Ne \stex_metagroup_new:n {\l_stex_current_module_str / \l_stex_feature_name_str}
2841 }
2842
2843 \cs_new_protected:Nn \__stex_features_do_for_list: {
2844   \seq_clear:N \l_stex_fors_seq
2845   \clist_map_inline:Nn \l_stex_key_for_clist {
2846     \exp_args:Ne \stex_get_in_morphism:n{\tl_to_str:n{##1}}
2847     \seq_put_right:Nx \l_stex_fors_seq
2848       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
2849   }
2850 }
2851
2852 \cs_new_protected:Nn \__stex_features_add_definiens:nn {
2853   \__stex_features_set_definiens_macros: #1 \__stex_features_break:
2854   \stex_assign_do:n{#2}
2855   #2
2856 }
2857 \cs_new_protected:Npn \__stex_features_set_definiens_macros: #1?#2?#3 \__stex_features_break:
2858   \str_set:Nn \l_stex_get_symbol_mod_str {#1?#2}
2859   \str_set:Nn \l_stex_get_symbol_name_str {#3}
2860   \exp_args:Nne \use:nn{\__stex_features_set_definiens_macros_i:nnnnnn}{
2861     \prop_item:Nn \l_stex_morphism_symbols_prop {[#1?#2]/[#3]}
2862   }
2863 }
2864 \cs_new_protected:Nn \__stex_features_set_definiens_macros_i:nnnnnn {
2865   \tl_set:Nn \l_stex_get_symbol_def_tl{#4}
2866 }
2867
2868 \cs_new_protected:Nn \stex_structural_feature_morphism_end: {
2869   \str_gset_eq:NN \l_stex_feature_name_str \l_stex_feature_name_str
2870   \str_gset_eq:NN \l_stex_current_domain_str \l_stex_current_domain_str
2871   \seq_gset_eq:NN \l_stex_morphism_symbols_prop \l_stex_morphism_symbols_prop
2872   \seq_gset_eq:NN \l_stex_morphism_renames_prop \l_stex_morphism_renames_prop
2873   \seq_gset_eq:NN \l_stex_morphism_morphisms_seq \l_stex_morphism_morphisms_seq
2874   \__stex_features_do_elaboration:
2875   \stex_if_do_html:TF{
2876     \end{stex_annotate_env}

```



```

2877   }\group_end:
2878 }
2879
2880 \cs_new_protected:Nn \__stex_features_setup: {
2881   \prop_clear:N \l_stex_morphism_symbols_prop
2882   \prop_clear:N \l_stex_morphism_renames_prop
2883   \seq_clear:N \l_stex_morphism_morphisms_seq
2884   \__stex_features_do_decls:
2885   \exp_args:No \__stex_features_do_morphisms:n \l_stex_current_domain_str
2886 }
2887
2888 \cs_new_protected:Nn \__stex_features_rename_all: {
2889
2890 }
2891
2892 \cs_new:Npn \__stex_features_clean:nw [#1] / #2 \stex_end: {
2893   [#1]/[#2]
2894 }
2895
2896 \cs_new_protected:Nn \__stex_features_do_decls: {
2897   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_domain_str {
2898     \stex_str_if_starts_with:nnTF{##3}[]{
2899       \exp_args:NNe \prop_put:Nnn \l_stex_morphism_symbols_prop {
2900         \__stex_features_clean:nw ##3 \stex_end:
2901       }
2902     }{
2903       \prop_put:Nnn \l_stex_morphism_symbols_prop
2904         {[#1]/[#3]}
2905     }{
2906       {##2}{##4}{##5}{##6}{##7}{##8}##9
2907     }
2908   }
2909 }
2910
2911 \cs_new_protected:Nn \stex_structural_feature_morphism_check_total: {
2912   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2913     \__stex_features_total_check: ##1 ##2
2914   }
2915 }
2916
2917 \cs_new_protected:Npn \__stex_features_total_check: [#1]/[#2] #3 #4 #5 #6 #7 #8 #9 {
2918   \tl_if_empty:nT{#6}{
2919     \msg_error:nxxx{stex}{error/needsdefiniens}{#1?#2}{total~morphism}
2920   }
2921 }
2922
2923 \cs_new:Npn \__stex_features_split_qm:w #1 ? #2 ? #3 { #3 }
2924 \cs_new_protected:Nn \__stex_features_do_elaboration: {
2925   \stex_debug:nn{morphisms}{
2926     Elaborating:^^J\prop_to_keyval:N \l_stex_morphism_symbols_prop
2927     ^^J
2928     Renamings:^^J
2929     \prop_to_keyval:N \l_stex_morphism_renames_prop
2930   }

```

```

2931 \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2932   \__stex_features_elab_check: ##1 ##2
2933 }
2934 \exp_args:No\stex_iterate_notations:nn\l_stex_current_domain_str{
2935   \prop_get:NnNTF \l_stex_morphism_renames_prop {##1}\l_stex_features_tmp {
2936     \exp_args:Ne \stex_module_add_notation:nnnnn
2937     {\l_stex_current_module_str ? \exp_after:wN \use_ii:nn \l_stex_features_tmp}
2938   }{
2939     \exp_args:Ne \stex_module_add_notation:nnnnn
2940     {\l_stex_current_module_str ? \l_stex_feature_name_str
2941     / \__stex_features_split_qm:w ##1}
2942   }{##2}{##3}{##4}{##5}
2943 }
2944 \stex_module_add_morphism:ooox
2945   \l_stex_feature_name_str
2946   \l_stex_current_domain_str
2947   \l_stex_features_feature_str
2948   {\prop_map_function:NN \l_stex_morphism_renames_prop \__stex_features_rename:nn}
2949 }
2950
2951 \cs_new:Nn \__stex_features_rename:nn{
2952   {#1}{\use_ii:nn#2}
2953 }
2954
2955 \cs_new_protected:Npn \__stex_features_elab_check: [#1]/[#2] #3 {
2956   \prop_get:NnNTF \l_stex_morphism_renames_prop {#1?#2} \l_stex_features_tmp {
2957     \stex_debug:nn{morphisms}{Generating~\l_stex_features_tmp}
2958     \exp_after:wN \stex_module_add_symbol:nnnnnnN \l_stex_features_tmp
2959   }{
2960     \bool_if:NTF \l_stex_features_implicit_bool {
2961       \stex_debug:nn{morphisms}{Generating-#3:~\l_stex_feature_name_str / #2}
2962       \exp_args:Nno \stex_module_add_symbol:nnnnnnN {#3}{\l_stex_feature_name_str / #2}
2963     }{
2964       \stex_debug:nn{morphisms}{Generating~\l_stex_feature_name_str / #2}
2965       \exp_args:Nno \stex_module_add_symbol:nnnnnnN {}{\l_stex_feature_name_str / #2}
2966     }
2967   }
2968 }
2969
2970 \cs_new_protected:Nn \__stex_features_do_morphisms:n {
2971   \prop_map_inline:cn {c_stex_module_#1_morphisms_prop}{
2972     \__stex_features_do_morph:nnnn ##2
2973   }
2974 }
2975
2976 \cs_new_protected:Nn \__stex_features_do_morph:nnnn {
2977   \tl_if_empty:nF{#3}{
2978     \seq_put_right:Nn \l_stex_morphism_morphisms_seq {{#1}{#2}{#3}}
2979   }
2980   \__stex_features_do_morphisms:n{#2}
2981 }
2982
2983 \cs_new_protected:Npn \__stex_features_reactivate: {
2984   \stex_deactivate_macro:Nn \symdecl {module~environments}

```

```

2985 \stex_deactivate_macro:Nn \textsymdecl {module~environments}
2986 \stex_deactivate_macro:Nn \symdef {module~environments}
2987 \stex_deactivate_macro:Nn \notation {module~environments}
2988 \stex_deactivate_macro:Nn \importmodule {module~environments}
2989 \stex_deactivate_macro:Nn \requiremodule {module~environments}
2990 \stex_deactivate_macro:Nn \smodule {outside~of~morphisms}
2991 \stex_reactivate_macro:N \assign
2992 \stex_reactivate_macro:N \assignMorphism
2993 \stex_reactivate_macro:N \renamedecl
2994 \cs_set_eq:NN \stex_do_for_list: \stex_features_do_for_list:
2995 \cs_set_eq:NN \stex_add_definiens:nn \stex_features_add_definiens:nn
2996 }

```

(End of definition for \stex\_structural\_feature\_morphism:nnn and \stex\_structural\_feature\_morphism\_end:. These functions are documented on page ??.)

**\stex\_get\_in\_morphism:n**

```

2997 \cs_new_protected:Nn \stex_get_in_morphism:n {
2998   \str_clear:N \l_stex_get_symbol_name_str
2999   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
3000     \exp_args:Nx \stex_features_get_check:nnnn{\tl_to_str:n{#1}}##1##2
3001   }
3002   \str_if_empty:NT \l_stex_get_symbol_name_str {
3003     \prop_map_inline:Nn \l_stex_morphism_renames_prop {
3004       \stex_features_renamed_check:nnnn{#1}##1##2
3005     }
3006     \str_if_empty:NT \l_stex_get_symbol_name_str {
3007       \msg_error:nnxx{stex}{error/unknownsymbolin}{#1}{
3008         morphism~\l_stex_feature_name_str
3009       }
3010     }
3011   }
3012 }
3013
3014 \cs_new_protected:Npn \stex_features_renamed_check:nnnn #1#2#3#4=#5#6 {
3015   \str_if_eq:nnTF{#1}{#5}{
3016     \exp_args:Nnx \use:nn{\stex_features_check_break:nnnnnnnn{#2#3}{#4}}{
3017       \prop_item:Nn \l_stex_morphism_symbols_prop {[#2#3]/[#4]}
3018     }
3019   }{
3020     \str_if_eq:nnT{#1}{#6}{
3021       \exp_args:Nnx \use:nn{\stex_features_check_break:nnnnnnnn{#2#3}{#4}}{
3022         \prop_item:Nn \l_stex_morphism_symbols_prop {[#2#3]/[#4]}
3023       }
3024     }
3025   }
3026 }
3027
3028 \cs_new_protected:Npn \stex_features_get_check:nnnn #1[#2]/[#3]#4 {
3029   \str_if_eq:nnTF{#1}{#3}{
3030     \stex_features_check_break:nnnnnnnn{#2}{#3}{#4}
3031   }{
3032     \str_if_eq:nnTF{#1}{#4}{
3033       \stex_features_check_break:nnnnnnnn{#2}{#3}{#4}

```

```

3034     }{
3035     \use_none:nnnnnn
3036     }
3037   }
3038 }
3039
3040 \cs_new_protected:Nn \__stex_features_check_break:nnnnnnnnn {
3041   \prop_map_break:n{
3042     \str_set:Nn \l_stex_get_symbol_mod_str{#1}
3043     \str_set:Nn \l_stex_get_symbol_name_str{#2}
3044     \str_set:Nn \l_stex_get_symbol_macro_str{#3}
3045     \int_set:Nn \l_stex_get_symbol_arity_int {#4}
3046     \tl_set:Nn \l_stex_get_symbol_args_tl {#5}
3047     \tl_set:Nn \l_stex_get_symbol_def_tl {#6}
3048     \tl_set:Nn \l_stex_get_symbol_type_tl {#7}
3049     \tl_set:Nn \l_stex_get_symbol_return_tl {#8}
3050     \tl_set:Nn \l_stex_get_symbol_invoke_cs {#9}
3051   }
3052 }

```

(End of definition for `\stex_get_in_morphism:n`. This function is documented on page 131.)

## 13.7 Inheritance

### 13.7.1 `\importmodule/\usemodule`

```

3053 <@=stex_importmodule>

```

`\usemodule`

```

3054 \stex_new_stylable_cmd:nnnn {usemodule} { 0{ } m } {
3055   \stex_import_module_uri:nn { #1 }{ #2 }
3056   \stex_import_require_module:ooo
3057   \l_stex_import_archive_str
3058   \l_stex_import_path_str
3059   \l_stex_import_name_str
3060   \stex_if_do_html:T {
3061     \hbox{\stex_annotate_invisible:nn
3062       {shtml:usemodule=\l_stex_import_ns_str} {}}
3063   }
3064   \stex_if_smsmode:F{
3065     \group_begin:
3066     \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3067     \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3068     \tl_clear:N \thisstyle
3069     \stex_style_apply:
3070     \group_end:
3071   }
3072 }{}

```

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

`\stex_import_module_uri:nn`

```

3073 \cs_new_protected:Nn \stex_import_module_uri:nn {
3074   \stex_debug:nn{importmodule}{URI:~>#1<~>#2<}

```

```

3075 \exp_args:NNnx \seq_set_split:Nnn \__stex_importmodule_seq ? { \tl_to_str:n{ #2 } }
3076 \seq_pop_right:NN \__stex_importmodule_seq \l_stex_import_name_str
3077 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \__stex_importmodule_seq ? }
3078 \tl_if_empty:nTF { #1 } {
3079   \stex_debug:nn{importmodule}{No~archive}
3080   \prop_if_exist:NTF \l_stex_current_archive_prop {
3081     \stex_debug:nn{importmodule}{Picking~current~archive}
3082     \str_set:Nx \l_stex_import_archive_str {
3083       \prop_item:Nn \l_stex_current_archive_prop { id }
3084     }
3085   }{
3086     \str_clear:N \l_stex_import_archive_str
3087     \str_set:Nn \l_stex_import_uri_str {file:}
3088     \str_if_empty:NTF \l_stex_import_path_str {
3089       \stex_debug:nn{importmodule}{Empty~Path}
3090       \stex_file_split_off_ext:NN \l__stex_importmodule_path_seq \g_stex_current_file
3091       \stex_file_split_off_lang:NN \l__stex_importmodule_path_seq \l__stex_importmodule_pa
3092       \str_set:Nx \l_stex_import_path_str {
3093         \stex_file_use:N \l__stex_importmodule_path_seq
3094       }
3095     }{
3096       \stex_debug:nn{importmodule}{Resolving~path~\l_stex_import_path_str~relative~to~\ste
3097       \stex_file_resolve:Nx \l__stex_importmodule_seq { \stex_file_use:N \g_stex_current_f
3098       \str_set:Nx \l_stex_import_path_str {
3099         \stex_file_use:N \l__stex_importmodule_seq
3100       }
3101       \stex_debug:nn{importmodule}{...yields~\l_stex_import_path_str}
3102     }
3103   }
3104 } {
3105   \stex_debug:nn{importmodule}{Archive~#1}
3106   \str_set:Nx \l_stex_import_archive_str { #1 }
3107   \stex_require_archive:o \l_stex_import_archive_str
3108   \str_set:Nx \l_stex_import_uri_str { \prop_item:cn{ c_stex_mathhub_ \l_stex_import_archi
3109 }
3110 }

```

(End of definition for `\stex_import_module_uri:nn`. This function is documented on page 130.)

`\stex_import_require_module:nnn`

`\stex_import_require_module:ooo`

```

3111 \cs_new_protected:Npn \stex_import_require_module:nnn #1 {
3112   \tl_if_empty:nTF { #1 } {
3113     \str_clear:N \l__stex_importmodule_archive_str
3114     \str_set:Nn \l_stex_import_uri_str {file:}
3115     \__stex_importmodule_get_module:nnn {}
3116   }{
3117     \stex_require_archive:n { #1 }
3118     \str_set:Nx \l__stex_importmodule_archive_str {#1}
3119     \str_set:Nx \l_stex_import_uri_str { \prop_item:cn{ c_stex_mathhub_ #1 _manifest_prop}{
3120     \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
3121     \exp_args:No \__stex_importmodule_get_module:nnn \l__stex_importmodule_str
3122   }
3123 }
3124 \cs_generate_variant:Nn \stex_import_require_module:nnn {ooo}

```

```

3125
3126 \cs_new_protected:Npn \stex_import_require_module_safe:nnn #1 {
3127   \tl_if_empty:nTF { #1 } {
3128     \str_clear:N \l_stex_importmodule_archive_str
3129     \str_set:Nn \l_stex_import_uri_str {file:}
3130     \__stex_importmodule_get_module_safe:nnn {}
3131   }{
3132     \stex_require_archive:n { #1 }
3133     \str_set:Nx \l__stex_importmodule_archive_str {#1}
3134     \str_set:Nx \l_stex_import_uri_str { \prop_item:cn{ c_stex_mathhub_ #1 _manifest_prop}{
3135     \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
3136     \exp_args:No \__stex_importmodule_get_module_safe:nnn \l__stex_importmodule_str
3137   }
3138 }
3139
3140 \cs_new_protected:Nn \__stex_importmodule_get_module_uri:nnn {
3141   \tl_if_empty:nF {#2}{
3142     \str_set:Nx \l_stex_import_uri_str {\l_stex_import_uri_str / #2}
3143   }
3144   \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
3145     Current~file:\stex_file_use:N \g_stex_current_file^^J
3146     Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3147   }
3148   \stex_if_module_exists:nTF {\l_stex_import_uri_str?#3} {
3149     \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3150   }{
3151     \stex_if_module_exists:nTF{\stex_uri_use:N \l_stex_current_ns_uri ? #3}{
3152       \str_set:Nx \l_stex_import_ns_str {\stex_uri_use:N \l_stex_current_ns_uri ? #3}
3153     }{
3154       \__stex_importmodule_get_from_file:nnn{#1}{#2}{#3}
3155       \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3156     }
3157   }
3158 }
3159 \cs_new_protected:Nn \__stex_importmodule_get_module_uri_safe:nnn {
3160   \tl_if_empty:nF {#2}{
3161     \str_set:Nx \l_stex_import_uri_str {\l_stex_import_uri_str / #2}
3162   }
3163   \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
3164     Current~file:\stex_file_use:N \g_stex_current_file^^J
3165     Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3166   }
3167   \stex_if_module_exists:nTF {\l_stex_import_uri_str?#3} {
3168     \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3169   }{
3170     \stex_if_module_exists:nTF{\stex_uri_use:N \l_stex_current_ns_uri ? #3}{
3171       \str_set:Nx \l_stex_import_ns_str {\stex_uri_use:N \l_stex_current_ns_uri ? #3}
3172     }{
3173       \__stex_importmodule_get_from_file_safe:nnn{#1}{#2}{#3}
3174       \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3175     }
3176   }
3177 }
3178

```

```

3179 \cs_new_protected:Nn \__stex_importmodule_get_module:nnn {
3180   \stex_debug:nn{importmodule}{Requiring->[#1]#2?#3<}
3181   \__stex_importmodule_get_module_uri:nnn{#1}{#2}{#3}
3182   \stex_activate_module:o \l_stex_import_ns_str
3183 }
3184
3185 \cs_new_protected:Nn \__stex_importmodule_get_module_safe:nnn {
3186   \stex_debug:nn{importmodule}{Requiring->[#1]#2?#3<}
3187   \__stex_importmodule_get_module_uri_safe:nnn{#1}{#2}{#3}
3188 }
3189
3190 \cs_new_protected:Nn \__stex_importmodule_get_from_file:nnn {
3191   \stex_file_resolve:Nx \l__stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3192   \str_set:Nx \l__stex_importmodule_str {\stex_file_use:N \l__stex_importmodule_seq}
3193   \stex_debug:nn{imports}{Looking-for~\l_stex_import_uri_str?#3...}
3194   \__stex_importmodule_check_file:nn{ /#3.tex }{
3195     \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
3196       \__stex_importmodule_check_file:nn{/#3.en.tex}{
3197         \__stex_importmodule_check_file:nn{.tex}{
3198           \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex}{
3199             \__stex_importmodule_check_file:nn{.en.tex}{
3200               \msg_error:nnx{stex}{error/unknownmodule}{\l_stex_import_uri_str?#3}
3201             }
3202           }
3203         }
3204       }
3205     }
3206   }
3207   \stex_if_smsmode:TF{
3208     \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
3209       \stex_debug:nn{imports}{Skipping-current-file}
3210     }{
3211       \__stex_importmodule_load_file:n{#3}
3212     }
3213   }{
3214     \__stex_importmodule_load_file:n{#3}
3215   }
3216 }
3217 \cs_new_protected:Nn \__stex_importmodule_get_from_file_safe:nnn {
3218   \stex_file_resolve:Nx \l__stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3219   \str_set:Nx \l__stex_importmodule_str {\stex_file_use:N \l__stex_importmodule_seq}
3220   \stex_debug:nn{imports}{Looking-for~\l_stex_import_uri_str?#3...}
3221   \__stex_importmodule_check_file:nn{ /#3.tex }{
3222     \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
3223       \__stex_importmodule_check_file:nn{/#3.en.tex}{
3224         \__stex_importmodule_check_file:nn{.tex}{
3225           \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex}{
3226             \__stex_importmodule_check_file:nn{.en.tex}{
3227               }
3228           }
3229         }
3230       }
3231     }
3232   }

```

```

3233 \stex_if_smsmode:TF{
3234   \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
3235   \stex_debug:nn{imports}{Skipping~current~file}
3236   }{
3237     \IfFileExists{ \l__stex_importmodule_str }{
3238       \__stex_importmodule_load_file:n{#3}
3239     }{}
3240   }
3241 }{
3242   \IfFileExists{ \l__stex_importmodule_str }{
3243     \__stex_importmodule_load_file:n{#3}
3244   }{}
3245 }
3246 }
3247
3248 \cs_new_protected:Nn \__stex_importmodule_load_file:n {
3249   \stex_file_in_smsmode:on \l__stex_importmodule_str {
3250     \str_if_empty:NF \l__stex_importmodule_archive_str {
3251       \stex_set_current_archive:n \l__stex_importmodule_archive_str
3252     }
3253     \stex_debug:nn{modules}{Loading~\l__stex_importmodule_str}
3254   }
3255   \stex_if_module_exists:nF {\l_stex_import_uri_str?#1} {
3256     \msg_error:nnx{stex}{error/unknownmodule}{\l_stex_import_uri_str?#1}
3257   }
3258 }
3259
3260 \cs_new_protected:Npn \__stex_importmodule_check_file:nn #1 {
3261   \stex_debug:nn{imports}{Checking~ \l__stex_importmodule_str #1}
3262   \IfFileExists{ \l__stex_importmodule_str #1 }{
3263     \stex_debug:nn{imports}{Success}
3264     \str_set:Nx \l__stex_importmodule_str { \l__stex_importmodule_str #1 }
3265   }
3266 }

```

(End of definition for `\stex_import_require_module:nnn`. This function is documented on page 130.)

## `\importmodule`

```

3267 \stex_new_stylable_cmd:nnnn{importmodule} { 0{} m } {
3268   \__stex_importmodule_import_module:nn {#1}{#2}
3269   \stex_smsmode_do:
3270 }{}
3271 \stex_deactivate_macro:Nn \importmodule {module~environments}
3272
3273 \cs_new_protected:Nn \__stex_importmodule_import_module:nn {
3274   \stex_import_module_uri:nn { #1 }{ #2 }
3275   \stex_import_require_module:ooo
3276     \l_stex_import_archive_str
3277     \l_stex_import_path_str
3278     \l_stex_import_name_str
3279   \stex_execute_in_module:x{
3280     \stex_activate_module:n{\l_stex_import_ns_str}
3281   }
3282   \stex_module_add_morphism:nonn

```



```

3283   {}{\l_stex_import_ns_str}{import}{}
3284 \stex_if_do_html:T {
3285   \stex_annotate_invisible:nn
3286   {shtml:import=\l_stex_import_ns_str} {}
3287 }
3288 \stex_if_smsmode:F{
3289   \group_begin:
3290   \tl_set:Nn \thisarchive {#1}
3291   \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3292   \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3293   \tl_clear:N \thisstyle
3294   \stex_style_apply:
3295   \group_end:
3296 }
3297 }
3298
3299 \cs_new_protected:Nn \__stex_importmodule_import_module_presms:nn {
3300   \stex_import_module_uri:nn { #1 }{ #2 }
3301   \tl_gput_right:Nx \g_stex_sms_import_code {
3302     \stex_import_require_module_safe:nnn
3303     {\l_stex_import_archive_str}
3304     {\l_stex_import_path_str}
3305     {\l_stex_import_name_str}
3306   }
3307 }
3308
3309 \stex_sms_allow_escape:N \importmodule
3310 \stex_every_module:n {\stex_reactivate_macro:N \importmodule}
3311 \stex_sms_allow_import:Nn \importmodule {
3312   \stex_reactivate_macro:N \importmodule
3313   \let \__stex_importmodule_import_module:nn \__stex_importmodule_import_module_presms:nn
3314 }
3315
3316 \stex_new_stylable_cmd:nnnn{requiremodule} { 0{ } m } {
3317   \stex_import_module_uri:nn { #1 }{ #2 }
3318   \stex_import_require_module:ooo
3319   \l_stex_import_archive_str
3320   \l_stex_import_path_str
3321   \l_stex_import_name_str
3322   \stex_do_up_to_module:x{
3323     \stex_activate_module:n{\l_stex_import_ns_str}
3324   }
3325   \stex_if_do_html:T {
3326     \stex_annotate_invisible:nn
3327     {shtml:import=\l_stex_import_ns_str} {}
3328   }
3329   \stex_if_smsmode:F{
3330     \group_begin:
3331     \tl_set:Nn \thisarchive {#1}
3332     \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3333     \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3334     \tl_clear:N \thisstyle
3335     \stex_style_apply:
3336     \group_end:

```

```

3337   }
3338   \stex_smsmode_do:
3339 }{}
3340 \stex_deactivate_macro:Nn \requiremodule {module-environments}

```

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

## 13.7.2 Theory Morphisms

```

3341 <@@=stex_morphisms>
3342 \stex_new_stylable_cmd:nnnn {assign} { m m }{
3343   \stex_get_in_morphism:n{#1}
3344   \_stex_assign_do:n{#2}
3345   \stex_smsmode_do:
3346 }{}
3347 \stex_sms_allow_escape:N\assign
3348
3349 \cs_new_protected:Nn \_stex_assign_do:n{
3350   \stex_debug:nn{assign}{Assigning~\l_stex_get_symbol_name_str~to~\tl_to_str:n{#1}}
3351   \tl_if_empty:NF \l_stex_get_symbol_def_tl {
3352     \%msg_error:nnxx{stex}{error/symbolalreadydefined}{\l_stex_get_symbol_name_str}{
3353       % morphism~\l_stex_feature_name_str
3354     \%}
3355   }
3356   \stex_check_term:n{#1}
3357   \stex_debug:nn{HERE!}{
3358     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str^^J
3359     \tl_to_str:n{#1}
3360   }
3361   \stex_if_do_html:T{
3362     \stex_annotate_invisible:nn{shtml:assign={\l_stex_get_symbol_mod_str?\l_stex_get_symbol
3363       \_stex_annotate_force_break:n{
3364         \mode_if_math:T\hbox{${\stex_annotate:nn{shtml:definiens={}}{#1}$}
3365       }
3366     }
3367   }
3368   \exp_args:Ne \stex_metagroup_do_in:nx{
3369     \l_stex_current_module_str / \l_stex_feature_name_str
3370 }{
3371   \prop_put:Nnn \exp_not:N \l_stex_morphism_symbols_prop
3372   {[\l_stex_get_symbol_mod_str]/[\l_stex_get_symbol_name_str]}
3373   {
3374     {\l_stex_get_symbol_macro_str}
3375     {\int_use:N \l_stex_get_symbol_arity_int}
3376     {\l_stex_get_symbol_args_tl}
3377     {\exp_not:n{#1}}
3378     {\exp_args:No\exp_not:n\l_stex_get_symbol_type_tl}
3379     {\exp_args:No\exp_not:n\l_stex_get_symbol_return_tl}
3380     {\l_stex_get_symbol_invoke_cs}
3381   }
3382 }
3383 }
3384
3385

```

```

3386 \stex_new_stylable_cmd:nnnn {renamedecl} { m 0{ } m }{
3387 \stex_get_in_morphism:n{#1}
3388 \_stex_renamedecl_do:nn{#2}{#3}
3389 \stex_smsmode_do:
3390 }{}
3391 \stex_sms_allow_escape:N\renamedecl
3392
3393 \cs_new_protected:Nn \_stex_renamedecl_do:nn {
3394 \stex_debug:nn{renamedecl}{Renaming~\l_stex_get_symbol_name_str~to~[#1]{#2}}
3395 \stex_if_do_html:T{
3396 \exp_args:Ne \stex_annotate_invisible:nn{
3397 shtml:rename={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str},
3398 shtml:macroname={#2}
3399 \str_if_empty:nF{#1}{ ,shtml:to={#1} }
3400 }{}
3401 }
3402 \exp_args:Ne \stex_metagroup_do_in:nx{
3403 \l_stex_current_module_str / \l_stex_feature_name_str
3404 }{
3405 \prop_put:Nnn \exp_not:N \l_stex_morphism_renames_prop
3406 {\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str}{#2}{
3407 \tl_if_empty:nTF{#1}{\l_stex_feature_name_str/\l_stex_get_symbol_name_str}{#1}
3408 }}
3409 }
3410 }
3411
3412 \stex_new_stylable_cmd:nnnn {assignMorphism} { m m }{
3413 \str_clear:N \l__stex_morphisms_morphism_dom_str
3414 \stex_iterate_morphisms:nn\l_stex_current_domain_str{
3415 \stex_debug:nn{assignMorphism}{
3416 Checking:~#1~vs:^^J##1^^J##2^^J##3^^J##4
3417 }
3418 \str_if_eq:nnTF{#1}{##1}{
3419 \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3420 }{
3421 \stex_str_if_ends_with:nnT{##2}{#1}{
3422 \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3423 }
3424 }
3425 }
3426 \str_if_empty:NT \l__stex_morphisms_morphism_dom_str {
3427 \msg_error:nnn{stex}{error/nomorphism}{#1}
3428 }
3429 \bool_set_false:N \l_tmpa_bool
3430 \stex_iterate_morphisms:nn \l_stex_current_module_str {
3431 \stex_debug:nn{assignMorphism}{
3432 Checking:~#2~vs:^^J##1^^J##2^^J##3^^J##4
3433 }
3434 \str_if_eq:nnTF{#2}{##1}{
3435 \stex_debug:nn{assignMorphism}{match!}
3436 \stex_iterate_break:n{
3437 \stex_annotate_invisible:nn{
3438 shtml:assignMorphismFrom={\l__stex_morphisms_morphism_dom_str}
3439 ahtml:assignMorphismTo={\l_stex_current_module_str?##1}

```

```

3440     }{}
3441     \bool_set_true:N \l_tmpa_bool
3442   }
3443   }{
3444   \stex_str_if_ends_with:nnT{##2}{#2}{
3445     \stex_debug:nn{assignMorphism}{match!}
3446     \stex_iterate_break:n{
3447       \stex_annotate_invisible:nn{
3448         shtml:assignMorphismFrom={\l_stex_morphisms_morphism_dom_str},
3449         shtml:assignMorphismTo={\l_stex_current_module_str?##1}
3450       }{}
3451       \bool_set_true:N \l_tmpa_bool
3452     }
3453   }
3454 }
3455 }
3456 \bool_if:NF \l_tmpa_bool {
3457   \msg_error:nnn{stex}{error/nomorphism}{#2}
3458 }
3459 }{}
3460 \cs_new_protected:Nn \_stex_morphisms_do_morph_assign:nnn {
3461   \stex_iterate_break:n{
3462     \str_set:Nx \l_stex_morphisms_morphism_dom_str { \l_stex_current_domain_str ? #1 }
3463     \stex_debug:nn{assignMorphism}{match!}
3464     \stex_iterate_symbols:nn{#2}{
3465       \stex_debug:nn{assignMorphism}{removing~##1?##3}
3466       % TODO: non-trivial assignments
3467       \prop_remove:Nn \l_stex_morphism_symbols_prop {
3468         [##1]/[##3]
3469       }
3470     }
3471   }
3472 }
3473
3474 \stex_deactivate_macro:Nn \assign {morphism-environments}
3475 \stex_deactivate_macro:Nn \renamedecr {morphism-environments}
3476 \stex_deactivate_macro:Nn \assignMorphism {morphism-environments}
3477 \stex_new_stylable_env:nnnnnn {copymodule}{m 0{ m}{
3478
3479   \stex_structural_feature_morphism:nnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3480
3481   \stex_if_smsmode:F {
3482     \tl_set:Nn \thiscopynome { #1 }
3483     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3484     \stex_style_apply:
3485   }
3486   \stex_smsmode_do:
3487 }{
3488   \stex_if_smsmode:F {
3489     \stex_style_apply:
3490   }
3491   \stex_structural_feature_morphism_end:
3492 }{}{}{}
3493 \stex_deactivate_macro:Nn \copymodule {module-environments}

```

```

3494 \stex_every_module:n {
3495   \stex_reactivate_macro:N \copymodule
3496 }
3497 \stex_sms_allow_env:n{copymodule}
3498
3499 \stex_new_stylable_env:nnnnnn {interpretmodule}{m 0{ } m}{
3500   \stex_structural_feature_morphism:nnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3501   \stex_if_smsmode:F {
3502     \tl_set:Nn \thiscopyname { #1 }
3503     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3504     \stex_style_apply:
3505   }
3506   \stex_smsmode_do:
3507 }{
3508   \stex_structural_feature_morphism_check_total:
3509   \stex_if_smsmode:F {
3510     \stex_style_apply:
3511   }
3512   \stex_structural_feature_morphism_end:
3513 }{}{}{}
3514 \stex_deactivate_macro:Nn \interpretmodule {module~environments}
3515 \stex_every_module:n {
3516   \stex_reactivate_macro:N \interpretmodule
3517 }
3518 \stex_sms_allow_env:n{interpretmodule}
3519 \stex_new_stylable_env:nnnnnn {realization}{0{ } m}{
3520
3521   \stex_structural_feature_morphism:nnnn{}{morphism}{#1}{#2}{,shtml:total=true}
3522   %\stex_execute_in_module:x{
3523   % \stex_activate_module:n{\l_stex_current_domain_str}
3524   %}
3525   \stex_if_smsmode:F {
3526     \tl_set:Nn \thiscopyname { #2 }
3527     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3528     \stex_style_apply:
3529   }
3530   \stex_smsmode_do:
3531 }{
3532   \stex_structural_feature_morphism_check_total:
3533   \stex_if_smsmode:F {
3534     \stex_style_apply:
3535   }
3536   \stex_structural_feature_morphism_end:
3537 }{}{}{}
3538 \stex_deactivate_macro:Nn \realization {module~environments}
3539 \stex_every_module:n {
3540   \stex_reactivate_macro:N \realization
3541 }
3542 \stex_sms_allow_env:n{realization}
3543
3544 \cs_new_protected:Nn \__stex_morphisms_parse_assign:n {
3545   \str_clear:N \l__stex_morphisms_name_str
3546   \str_clear:N \l__stex_morphisms_newname_str
3547   \tl_clear:N \l__stex_morphisms_ass_tl

```

```

3547 \stex_debug:nn{morphisms}{Parsing~#1}
3548 \exp_args:NNe \seq_set_split:Nnn \l__stex_morphisms_seq {\tl_to_str:n{0}} {#1}
3549 \int_compare:nNnTF {\seq_count:N \l__stex_morphisms_seq} = 1 {
3550   \stex_debug:nn{morphisms}{No~0}
3551   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_next_tl
3552 }{
3553   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3554   \stex_debug:nn{morphisms}{Name:~\l__stex_morphisms_name_str}
3555   \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3556   \tl_set:Nx \l__stex_morphisms_next_tl {\seq_use:Nn \l__stex_morphisms_seq @}
3557 }
3558 \exp_args:NNNo \seq_set_split:Nnn \l__stex_morphisms_seq = \l__stex_morphisms_next_tl
3559 \str_if_empty:NTF \l__stex_morphisms_name_str {
3560   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3561   \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3562   \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3563 }{
3564   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_newname_str
3565   \exp_args:NNo \str_set:Nn \l__stex_morphisms_newname_str \l__stex_morphisms_newname_str
3566   \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3567 }
3568 \__stex_morphisms_do_parsed_assign:
3569 }
3570
3571 \cs_new_protected:Nn \__stex_morphisms_do_parsed_assign: {
3572   \exp_args:No \stex_get_in_morphism:n \l__stex_morphisms_name_str
3573   \str_if_empty:NF \l__stex_morphisms_newname_str {
3574     \exp_after:wN \__stex_morphisms_do_parsed_newname: \l__stex_morphisms_newname_str \__stex
3575   }
3576   \tl_if_empty:NF \l__stex_morphisms_ass_tl {
3577     \exp_args:No \stex_assign_do:n \l__stex_morphisms_ass_tl
3578   }
3579 }
3580
3581 \cs_new_protected:Nn \__stex_morphisms_do_parsed_newname: {
3582   \peek_charcode:NTF [ {
3583     \__stex_morphisms_do_parsed_newname:w
3584   }{
3585     \__stex_morphisms_do_parsed_newname:w []
3586   }
3587 }
3588
3589 \cs_new_protected:Npn \__stex_morphisms_do_parsed_newname:w [#1] #2 \__stex_morphisms_end: {
3590   \stex_renamedec1_do:nn{#1}{#2}
3591 }
3592
3593 \stex_new_stylable_cmd:nnnn{copymod}{m 0{} m m}{
3594   \stex_structural_feature_morphism:nnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3595
3596   \clist_map_function:nN{#4}\__stex_morphisms_parse_assign:n
3597
3598   \stex_if_smsmode:F {
3599     \tl_set:Nn \thiscopyname { #1 }
3600     \tl_set_eq:NN \thismoduleuri \l__stex_current_domain_str

```

```

3601     \stex_style_apply:
3602   }
3603   \stex_structural_feature_morphism_end:
3604   \stex_smsmode_do:
3605 }{}
3606 \stex_deactivate_macro:Nn \copymod {module-environments}
3607 \stex_every_module:n {
3608   \stex_reactivate_macro:N \copymod
3609 }
3610 \stex_sms_allow_escape:N\copymod
3611
3612
3613 \stex_new_stylable_cmd:nmmm{interpretmod}{m 0{} m m}{
3614   \stex_structural_feature_morphism:nmmmm{#1}{morphism}{#2}{#3}{,shtml:total=true}
3615
3616   \clist_map_function:nN{#4}\_stex_morphisms_parse_assign:n
3617
3618   \stex_if_smsmode:F {
3619     \tl_set:Nn \thiscopyname { #1 }
3620     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3621     \stex_style_apply:
3622   }
3623   \stex_structural_feature_morphism_check_total:
3624   \stex_structural_feature_morphism_end:
3625   \stex_smsmode_do:
3626 }{}
3627 \stex_deactivate_macro:Nn \interpretmod {module-environments}
3628 \stex_every_module:n {
3629   \stex_reactivate_macro:N \interpretmod
3630 }
3631 \stex_sms_allow_escape:N\interpretmod
3632
3633
3634 \stex_new_stylable_cmd:nmmn{realize}{0{} m m}{
3635   \stex_structural_feature_morphism:nmmmm{}{morphism}{#1}{#2}{,shtml:total=true}
3636
3637   \clist_map_function:nN{#3}\_stex_morphisms_parse_assign:n
3638
3639   \stex_if_smsmode:F {
3640     \tl_set:Nn \thiscopyname { #1 }
3641     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3642     \stex_style_apply:
3643   }
3644   \stex_structural_feature_morphism_check_total:
3645   \stex_structural_feature_morphism_end:
3646   \stex_smsmode_do:
3647 }{}
3648 \stex_deactivate_macro:Nn \realize {module-environments}
3649 \stex_every_module:n {
3650   \stex_reactivate_macro:N \realize
3651 }
3652 \stex_sms_allow_escape:N\realize

```

## 13.8 Symbols

### 13.8.1 Declarations

```
3653 <@@=stex_symdecl>
```

Some setup:

```
\stex_if_check_terms_p:
```

```
\stex_if_check_terms:TF
```

```
3654 \stex_if_html_backend:TF {
3655   \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3656     \prg_return_false:
3657   }
3658 }{
3659   \stex_get_env:Nn\__stex_symdecl_env_str{STEX_CHECKTERMS}
3660   \str_if_empty:NF\__stex_symdecl_env_str{
3661     \exp_args:No \str_if_eq:nnF \__stex_symdecl_env_str{false}{
3662       \bool_set_true:N \c_stex_check_terms_bool
3663     }
3664   }
3665   \bool_if:NTF \c_stex_check_terms_bool {
3666     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3667       \prg_return_true:
3668     }
3669   }{
3670     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3671       \prg_return_false:
3672     }
3673   }
3674 }
```

(End of definition for `\stex_if_check_terms:TF`. This function is documented on page 124.)

```
\stex_check_term:n
```

```
3675 \stex_if_check_terms:TF{
3676   \cs_new_protected:Nn \stex_check_term:n {
3677     \hbox_set:Nn \l_tmpa_box {
3678       \group_begin:
3679         $#1$
3680       \group_end:
3681     }
3682   }
3683 }{
3684   \cs_new_protected:Nn \stex_check_term:n {}
3685 }
```

(End of definition for `\stex_check_term:n`. This function is documented on page 124.)

symdecl arguments:

```
3686 \stex_keys_define:nnnn{symargs}{
3687   \str_clear:N \l_stex_key_args_str
3688   \str_clear:N \l_stex_key_role_str
3689   \str_clear:N \l_stex_key_reorder_str
3690   \str_clear:N \l_stex_key_assoc_str
3691 }{
3692   args      .str_set:N = \l_stex_key_args_str ,
3693   reorder   .str_set:N = \l_stex_key_reorder_str ,
```



```

3694   assoc      .choices:nn   = {bin,binl,binr,pre,conj,pwconj}
3695   {\str_set:Nx \l_stex_key_assoc_str \l_keys_choice_tl},
3696   role       .str_set:N     = \l_stex_key_role_str
3697 }{}
3698
3699 \stex_keys_define:n{decl}{
3700   \str_clear:N \l_stex_key_name_str
3701   \str_clear:N \l_stex_key_args_str
3702   \tl_clear:N \l_stex_key_type_tl
3703   \tl_clear:N \l_stex_key_def_tl
3704   \tl_clear:N \l_stex_key_return_tl
3705   \str_clear:N \l_stex_key_wikidata_str
3706   \clist_clear:N \l_stex_key_argtypes_clist
3707 }{
3708   name       .str_set:N     = \l_stex_key_name_str ,
3709
3710   return     .tl_set:N      = \l_stex_key_return_tl ,
3711   argtypes   .clist_set:N   = \l_stex_key_argtypes_clist ,
3712   wikidata   .str_set:N     = \l_stex_key_wikidata_str ,
3713
3714   type       .tl_set:N      = \l_stex_key_type_tl ,
3715   def        .tl_set:N      = \l_stex_key_def_tl ,
3716
3717   align      .code:n        = {},
3718   gfc        .code:n        = {}
3719 }{style,deprecate,symargs}
3720 % \stex_do_deprecation:n{#2}

```

## \symdecl

```

3721 \str_new:N \l_stex_macroname_str
3722 \stex_new_stylable_cmd:n{nnnn {symdecl} { s m 0}} {
3723   \stex_keys_set:nn{decl}{#3}
3724   \IfBooleanTF #1 {
3725     \str_clear:N \l_stex_macroname_str
3726   }{
3727     \str_set:Nx \l_stex_macroname_str { #2 }
3728   }
3729   \stex_symdecl_top:n{#2}
3730
3731   \stex_if_smsmode:F{
3732     \group_begin:
3733     \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3734     \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3735     \tl_set_eq:NN \thistype \l_stex_key_type_tl
3736     \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
3737     \tl_set_eq:NN \thisargs \l_stex_key_args_str
3738     \tl_clear:N \thisstyle
3739     \stex_style_apply:
3740     \group_end:
3741   }
3742   \stex_smsmode_do:
3743 }{}
3744 \stex_deactivate_macro:Nn \symdecl {module-environments}
3745 \stex_every_module:n {\stex_reactivate_macro:N \symdecl}

```

```
3746 \stex_sms_allow_escape:N \symdecl
```

(End of definition for `\symdecl`. This function is documented on page 87.)

```
\stex_symdecl_top:n
```

```
3747 \cs_new_protected:Nn \stex_symdecl_top:n {
3748   \str_if_empty:NT \l_stex_key_name_str {
3749     \str_set:Nx \l_stex_key_name_str { #1 }
3750   }
3751   \stex_symdecl_do:
3752   \_stex_symdecl_check_terms:
3753   \_stex_symdecl_add_decl:
3754   \stex_if_do_html:T {
3755     \_stex_symdecl_html:
3756   }
3757 }
3758
3759 \cs_new_protected:Nn \_stex_symdecl_add_decl: {
3760   \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnN}{
3761     {\l_stex_macroname_str}
3762     {\l_stex_key_name_str}
3763     {\int_use:N \l_stex_get_symbol_arity_int}
3764     {\l_stex_get_symbol_args_tl}
3765     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }%{\exp_args:No \exp_not:n \l_stex_key_def_tl}
3766     {}%{\exp_args:No \exp_not:n \l_stex_key_type_tl}
3767     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
3768     \stex_invoke_symbol:
3769   }
3770   \exp_args:Ne \stex_ref_new_symbol:n
3771     {\l_stex_current_module_str?\l_stex_key_name_str}
3772 }
3773
3774 \cs_new:Nn \_stex_return_args:nn {
3775   {\svar{ARGUMENT_#1}\_stex_eat_exclamation_point:}
3776 }
3777
3778 \cs_new_protected:Nn \_stex_symdecl_html: {
3779   \exp_args:Ne \stex_annotate_invisible:nn {
3780     shtml:symdecl = {\l_stex_current_module_str ? \l_stex_key_name_str},
3781     shtml:args = {\l_stex_key_args_str}
3782     \str_if_empty:NF \l_stex_macroname_str {,
3783       shtml:macroname={\l_stex_macroname_str}
3784     }
3785     \str_if_empty:NF \l_stex_key_wikidata_str {,
3786       shtml:wikidata={\l_stex_key_wikidata_str}
3787     }
3788     \str_if_empty:NF \l_stex_key_assoc_str {,
3789       shtml:asstype={\l_stex_key_assoc_str}
3790     }
3791     \str_if_empty:NF \l_stex_key_reorder_str {,
3792       shtml:reorderargs={\l_stex_key_reorder_str}
3793     }
3794     \str_if_empty:NF \l_stex_key_role_str {,
3795       shtml:role={\l_stex_key_role_str}
```

```

3796   }
3797   }{\hbox\bgroup\_stex_annotate_force_break:n{
3798     \bool_set_true:N \stex_in_invisible_html_bool
3799     \tl_if_empty:NF \l_stex_key_type_tl {
3800       $\stex_annotate:nn{shtml:type={}}{\l_stex_key_type_tl}$
3801     }
3802     \tl_if_empty:NF \l_stex_key_def_tl {
3803       $\stex_annotate:nn{shtml:definiens={}}{\l_stex_key_def_tl}$
3804     }
3805     \tl_if_empty:NF \l_stex_key_return_tl{
3806       \exp_args:Nno \use:n{
3807         \cs_generate_from_arg_count:NNnn \l__stex_symdecl_cs
3808         \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
3809         \tl_set:Nx \l__stex_symdecl_args_tl {\_stex_map_args:N \_stex_return_args:nn}
3810         $\stex_annotate:nn{shtml:returtype={}}{
3811           \exp_after:wN \l__stex_symdecl_cs \l__stex_symdecl_args_tl!
3812         }$
3813       }
3814       \clist_if_empty:NF \l_stex_key_argtypes_clist {
3815         \stex_annotate:nn{shtml:argtypes={}}{\_stex_annotate_force_break:n{
3816           \clist_map_inline:Nn \l_stex_key_argtypes_clist {
3817             $\stex_annotate:nn{shtml:type={}}{##1}$
3818           }
3819         }}
3820       }
3821     }\egroup}
3822 }

```

(End of definition for `\stex_symdecl_top:n`. This function is documented on page 125.)

**`\stex_symdecl_do:`** Requires the above keys and `\l_stex_macroname_str` to be set first

```

3823 \cs_new_protected:Nn \stex_symdecl_do: {
3824   \_stex_do_deprecation:n \l_stex_key_name_str
3825   \__stex_symdecl_parse_arity:
3826   \__stex_symdecl_do_args:
3827 }
3828
3829 \int_new:N \l_stex_assoc_args_count
3830
3831 \cs_new_protected:Nn \__stex_symdecl_parse_arity: {
3832   \int_zero:N \l_stex_get_symbol_arity_int
3833   \int_zero:N \l_stex_assoc_args_count
3834   \str_map_inline:Nn \l_stex_key_args_str {
3835     \str_case:nnF ##1 {
3836       0 { \str_map_break: }
3837       1 { \str_map_break:n{
3838         \int_set:Nn \l_stex_get_symbol_arity_int {1}
3839         \str_set:Nn \l_stex_key_args_str {i}
3840       } }
3841       2 { \str_map_break:n{
3842         \int_set:Nn \l_stex_get_symbol_arity_int {2}
3843         \str_set:Nn \l_stex_key_args_str {ii}
3844       } }
3845       3 { \str_map_break:n{

```

```

3846     \int_set:Nn \l_stex_get_symbol_arity_int {3}
3847     \str_set:Nn \l_stex_key_args_str {iii}
3848   } }
3849   4 { \str_map_break:n{
3850     \int_set:Nn \l_stex_get_symbol_arity_int {4}
3851     \str_set:Nn \l_stex_key_args_str {iiii}
3852   } }
3853   5 { \str_map_break:n{
3854     \int_set:Nn \l_stex_get_symbol_arity_int {5}
3855     \str_set:Nn \l_stex_key_args_str {iiiii}
3856   } }
3857   6 { \str_map_break:n{
3858     \int_set:Nn \l_stex_get_symbol_arity_int {6}
3859     \str_set:Nn \l_stex_key_args_str {iiiiii}
3860   } }
3861   7 { \str_map_break:n{
3862     \int_set:Nn \l_stex_get_symbol_arity_int {7}
3863     \str_set:Nn \l_stex_key_args_str {iiiii}
3864   } }
3865   8 { \str_map_break:n{
3866     \int_set:Nn \l_stex_get_symbol_arity_int {8}
3867     \str_set:Nn \l_stex_key_args_str {iiiiiii}
3868   } }
3869   9 { \str_map_break:n{
3870     \int_set:Nn \l_stex_get_symbol_arity_int {9}
3871     \str_set:Nn \l_stex_key_args_str {iiiiiiii}
3872   } }
3873   i {\int_incr:N \l_stex_get_symbol_arity_int}
3874   b {\int_incr:N \l_stex_get_symbol_arity_int}
3875   a {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3876   B {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3877   ){
3878     \msg_error:nxxx{stex}{error/wrongargs}{
3879       \l_stex_current_module_str ? \l_stex_key_name_str
3880     }{##1}
3881   }
3882 }
3883 }
3884
3885 \cs_new_protected:Nn \__stex_symdecl_do_args: {
3886   \tl_clear:N \l_stex_get_symbol_args_tl
3887   \int_step_inline:nn \l_stex_get_symbol_arity_int {
3888     \tl_put_right:Nn \l_stex_get_symbol_args_tl {##1}
3889     \tl_put_right:Nx \l_stex_get_symbol_args_tl {
3890       \str_item:Nn \l_stex_key_args_str {##1}
3891     }
3892   }
3893 }

```

(End of definition for `\stex_symdecl_do:`. This function is documented on page 124.)

`\stex_symdecl_check_terms:`

```

3894 \cs_new_protected:Nn \stex_symdecl_check_terms: {
3895   \stex_check_term:n{

```

```

3896 \stex_debug:nn{check_terms}{Checking~type...}
3897 \group_begin:\l_stex_key_type_tl\group_end:
3898 \stex_debug:nn{check_terms}{Checking~definiens...}
3899 \group_begin:\l_stex_key_def_tl\group_end:
3900 \stex_debug:nn{check_terms}{Checking~return...}
3901 \group_begin:\l_stex_key_return_tl!\group_end:
3902 \stex_debug:nn{check_terms}{Checking~argument~types...}
3903 \group_begin:\l_stex_key_argtypes_clist\group_end:
3904 }
3905 }

```

(End of definition for `\_stex_symdecl_check_terms:`. This function is documented on page 125.)

### `\textsymdecl`

```

3906
3907 \stex_keys_define:nnnn{textsymdecl}{
3908 \str_clear:N \l_stex_key_name_str
3909 \tl_clear:N \l_stex_key_type_tl
3910 \tl_clear:N \l_stex_key_def_tl
3911 }{
3912 name .str_set:N = \l_stex_key_name_str ,
3913 type .tl_set:N = \l_stex_key_type_tl ,
3914 def .tl_set:N = \l_stex_key_def_tl
3915 }{style,deprecate}
3916
3917 \stex_new_stylable_cmd:nnnn {textsymdecl} {m O{} m} {
3918 \stex_keys_set:nn{symdef}{}
3919 \stex_keys_set:nn{textsymdecl}{#2}
3920 \str_set:Nx \l_stex_macroname_str { #1 }
3921 \str_if_empty:NT \l_stex_key_name_str {
3922 \str_set:Nn \l_stex_key_name_str {#1}
3923 }%{
3924 % \str_set:Nx \l_stex_key_name_str {\l_stex_key_name_str-sym}
3925 %}
3926 \str_set:Nn \l_stex_key_role_str {textsymdecl}
3927
3928 \stex_symdecl_do:
3929 \_stex_symdecl_check_terms:
3930 \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnN}{
3931 {\l_stex_macroname_str}
3932 {\l_stex_key_name_str}
3933 {0}{}
3934 {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
3935 }% type
3936 {\use:c{#1name_nospace}}% return
3937 \stex_invoke_text_symbol:
3938 }
3939 \exp_args:Ne \stex_ref_new_symbol:n
3940 {\l_stex_current_module_str?\l_stex_key_name_str}
3941 \stex_if_do_html:T {
3942 \_stex_symdecl_html:
3943 }
3944
3945 \int_set:Nn \l_stex_get_symbol_arity_int 0

```

```

3946 \tl_clear:N \l_stex_key_op_tl
3947 \str_clear:N \l_stex_key_intent_str
3948 \str_clear:N \l_stex_key_prec_str
3949 \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
3950 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
3951 \stex_notation_parse:n{\hbox{#3}}
3952 \_stex_notation_add:
3953 \stex_if_do_html:T {
3954   \def\comp{\_comp}
3955   \stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
3956 }
3957 \stex_execute_in_module:x{
3958   \_stex_symdecl_set_textsymdecl_macro:nnn{#1}{\l_stex_current_module_str?\l_stex_key_name_str}
3959   \exp_not:n{#3}
3960 }
3961
3962 \stex_if_smsmode:F{
3963   \group_begin:
3964   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3965   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3966   \tl_clear:N \thisstyle
3967   \stex_style_apply:
3968   \group_end:
3969 }
3970 \stex_smsmode_do:
3971 }{}
3972 \stex_deactivate_macro:Nn \textsymdecl {module~environments}
3973 \stex_every_module:n {\stex_reactivate_macro:N \textsymdecl}
3974 \stex_sms_allow_escape:N \textsymdecl
3975
3976 \cs_new_protected:Nn \_stex_symdecl_set_textsymdecl_macro:nnn {
3977   \cs_set_protected:cpn{#1name_nospace}{#3}
3978   \cs_set_protected:cpn{#1name}{
3979     \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3980     \mode_if_math:T{\hbox{\let\xspace\relax #3}}
3981     \mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3982   }
3983 }
3984
3985 \cs_new_protected:Nn \stex_invoke_text_symbol: {
3986   \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3987   \_stex_term_oms_or_omv:nnn{}{}{\maincomp{\let\xspace\relax\l_stex_current_return_tl}}
3988   \group_end:\mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3989 }

```

(End of definition for \textsymdecl. This function is documented on page 88.)

### \stex\_get\_symbol:n

```

3990 \cs_new_protected:Nn \stex_get_symbol:n {
3991   \_stex_get_symbol:n{ #1 }
3992   \str_if_empty:NT \l_stex_get_symbol_name_str {
3993     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3994   }
3995 }

```

```

3996
3997 \cs_new_protected:Nn \_stex_get_symbol:n {
3998   \str_clear:N \l_stex_get_symbol_mod_str
3999   \str_clear:N \l_stex_get_symbol_name_str
4000   \cs_if_exist:cTF { #1 }{
4001     \cs_set_eq:Nc \l__stex_symdecl_cs { #1 }
4002     % command name
4003     \exp_args:Ne \tl_if_empty:nTF { \cs_argument_spec:N \l__stex_symdecl_cs }{
4004       % ...that takes no arguments
4005       \exp_args:Ne \cs_if_eq:NNTF {\tl_head:N \l__stex_symdecl_cs}
4006         \_stex_invoke_symbol:nnnnnnN
4007         \__stex_symdecl_get_symbol_from_cs:
4008         {\__stex_symdecl_get_symbol_from_string:n { #1 }}
4009     }{
4010       \__stex_symdecl_get_symbol_from_string:n { #1 }
4011     }
4012   }{
4013     \__stex_symdecl_get_symbol_from_string:n { #1 }
4014   }
4015 }
4016
4017 \int_new:N \l_stex_get_symbol_arity_int
4018 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
4019   \stex_debug:nn{symbols}{Getting-from-cs...}
4020   \stex_pseudogroup_with:nn{\_stex_invoke_symbol:nnnnnnN}{
4021     \cs_set:Npn \_stex_invoke_symbol:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {
4022       \str_set:Nn \l_stex_get_symbol_mod_str {##1}
4023       \str_set:Nn \l_stex_get_symbol_name_str {##2}
4024       \int_set:Nn \l_stex_get_symbol_arity_int {##3}
4025       \tl_set:Nn \l_stex_get_symbol_args_tl {##4}
4026       \tl_set:Nn \l_stex_get_symbol_def_tl {##5}
4027       \tl_set:Nn \l_stex_get_symbol_type_tl {##6}
4028       \tl_set:Nn \l_stex_get_symbol_return_tl {##7}
4029       \tl_set:Nn \l_stex_get_symbol_invoke_cs {##8}
4030     }
4031     \l__stex_symdecl_cs
4032   }
4033 }
4034
4035 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
4036   \stex_debug:nn{symbols}{Getting-from-string-#1...}
4037   \seq_set_split:Nnn \l__stex_symdecl_seq ? {#1}
4038   \seq_pop_right:NN \l__stex_symdecl_seq \l__stex_symdecl_name
4039   \seq_if_empty:NTF \l__stex_symdecl_seq {
4040     \exp_args:No \__stex_symdecl_get_from_one_string:n {#1}
4041   }{
4042     \exp_args:NNe \exp_args:Nno \__stex_symdecl_get_symbol_from_modules:nn {
4043       \seq_use:Nn \l__stex_symdecl_seq ?
4044     } \l__stex_symdecl_name
4045   }
4046 }
4047
4048 \cs_new_protected:Nn \__stex_symdecl_sym_from_str_i:nnnn {
4049   \bool_lazy_any:nTF{

```

```

4050     {\str_if_eq_p:nn{#2}{#3}}
4051     {\str_if_eq_p:nn{#2}{#4}}
4052     {\stex_str_if_ends_with_p:nn{#4}{/#2}}
4053   }{
4054     \__stex_symdecl_sym_i_finish:nnnnnnN{#1}{#4}
4055   }{
4056     \__stex_symdecl_sym_i_gobble:nnnnnn
4057   }
4058 }
4059 \cs_new_protected:Nn \__stex_symdecl_sym_i_gobble:nnnnnn {}
4060
4061 \cs_new_protected:Nn \__stex_symdecl_sym_i_finish:nnnnnnN {
4062   \prop_map_break:n{\seq_map_break:n{
4063     \str_set:Nn \l_stex_get_symbol_mod_str {#1}
4064     \str_set:Nn \l_stex_get_symbol_name_str {#2}
4065     \int_set:Nn \l_stex_get_symbol_arity_int {#3}
4066     \tl_set:Nn \l_stex_get_symbol_args_tl {#4}
4067     \tl_set:Nn \l_stex_get_symbol_def_tl {#5}
4068     \tl_set:Nn \l_stex_get_symbol_type_tl {#6}
4069     \tl_set:Nn \l_stex_get_symbol_return_tl {#7}
4070     \tl_set:Nn \l_stex_get_symbol_invoke_cs {#8}
4071   }}
4072 }
4073
4074 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_modules:nn {
4075   \stex_debug:nn{symbols}{Getting~#2~in~#1...}
4076   \seq_map_inline:Nn \l_stex_all_modules_seq {
4077     \stex_str_if_ends_with:nnT{##1}{#1}{
4078       \prop_map_inline:cn{c_stex_module_##1_symbols_prop}{
4079         \__stex_symdecl_sym_from_str_i:nnnn{##1}{#2} ####2
4080       }
4081     }
4082   }
4083 }
4084
4085 \cs_new_protected:Nn \__stex_symdecl_get_from_one_string:n {
4086   \stex_debug:nn{symbols}{Getting~#1~anywhere...}
4087   \stex_iterate_symbols:n{
4088     %\stex_debug:nn{symbols}{>#1==##2~|~#1==##3<...}
4089     \bool_lazy_any:nT{
4090       {\str_if_eq_p:nn{#1}{##2}}
4091       {\str_if_eq_p:nn{#1}{##3}}
4092       {\stex_str_if_ends_with_p:nn{##3}{/#1}}
4093     }{
4094       \stex_iterate_break:n{
4095         \str_set:Nn \l_stex_get_symbol_mod_str {##1}
4096         \str_set:Nn \l_stex_get_symbol_name_str {##3}
4097         \int_set:Nn \l_stex_get_symbol_arity_int {##4}
4098         \tl_set:Nn \l_stex_get_symbol_args_tl {##5}
4099         \tl_set:Nn \l_stex_get_symbol_def_tl {##6}
4100         \tl_set:Nn \l_stex_get_symbol_type_tl {##7}
4101         \tl_set:Nn \l_stex_get_symbol_return_tl {##8}
4102         \tl_set:Nn \l_stex_get_symbol_invoke_cs {##9}
4103       }

```



```

4104     }
4105   }
4106 }

```

(End of definition for `\stex_get_symbol:n`. This function is documented on page 124.)

## 13.8.2 Notations

```

4107 <@@=stex_notations>

```

```

\stex_map_args:N
\stex_map_notation_args:N
4108 \cs_new:Nn \stex_map_args:N {
4109   \tl_if_empty:NF \l_stex_get_symbol_args_tl {
4110     \exp_after:wN \stex_notations_map_args_i:w \exp_after:wN
4111     #1 \l_stex_get_symbol_args_tl \stex_notations_args_end:
4112   }
4113 }
4114 \cs_new:Npn \stex_notations_map_args_i:w #1 #2 #3 #4 \stex_notations_args_end: {
4115   #1 #2 #3
4116   \tl_if_empty:nF{#4}{
4117     \stex_notations_map_args_i:w #1 #4 \stex_notations_args_end:
4118   }
4119 }
4120
4121 \cs_new:Nn \stex_map_notation_args:N {
4122   \tl_if_empty:NF \l_stex_notation_args_tl {
4123     \exp_after:wN \stex_notations_map_args_ii:w \exp_after:wN
4124     #1 \l_stex_get_symbol_args_tl \stex_notations_args_end:
4125   }
4126 }
4127 \cs_new:Npn \stex_notations_map_args_ii:w #1 #2 #3 #4 #5 #6 \stex_notations_args_end: {
4128   #1 #2 #3 #4 #5
4129   \tl_if_empty:nF{#6}{
4130     \stex_notations_map_args_ii:w #1 #6 \stex_notations_args_end:
4131   }
4132 }

```

(End of definition for `\stex_map_args:N` and `\stex_map_notation_args:N`. These functions are documented on page ??.)

notation arguments:

```

4133 \stex_keys_define:nnnn{notation}{
4134   \str_clear:N \l_stex_key_variant_str
4135   \str_clear:N \l_stex_key_prec_str
4136   \str_clear:N \l_stex_key_op_tl
4137   \str_clear:N \l_stex_key_intent_str
4138   \clist_clear:N \l_stex_key_intent_args_clist
4139 }{
4140   variant   .str_set_x:N = \l_stex_key_variant_str ,
4141   prec      .str_set_x:N = \l_stex_key_prec_str ,
4142   op        .tl_set:N    = \l_stex_key_op_tl ,
4143   intent    .str_set:N    = \l_stex_key_intent_str ,
4144   argnames  .clist_set:N  = \l_stex_key_intent_args_clist ,
4145   unknown   .code:n      = {
4146     \str_if_empty:NTF \l_keys_key_str {
4147       \str_set:Nx \l_stex_key_variant_str {\l_keys_key_tl}

```

```

4148     }{
4149     \str_set_eq:NN \l_stex_key_variant_str \l_keys_key_str
4150     }
4151   }
4152 }{style}

```

## `\notation`

```

4153 \stex_new_stylable_cmd:nmmm {notation} { s m 0{} m} {
4154   \stex_keys_set:nn{notation}{#3}
4155   \stex_get_symbol:n{#2}
4156   \stex_notation_parse:n{#4}
4157   \stex_if_check_terms:T{ \_stex_notation_check: }
4158   \_stex_notation_add:
4159   \stex_if_do_html:T {
4160     \def\comp{\_comp}
4161     \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4162   }
4163   \IfBooleanTF#1{
4164     \_stex_notation_set_default:n{
4165       \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4166     }
4167   }{}
4168   \stex_if_smsmode:F{
4169     \group_begin:
4170     \_stex_notations_styledefs:
4171     \stex_style_apply:
4172     \group_end:
4173   }
4174   \stex_smsmode_do:
4175 }{}
4176
4177 \cs_new_protected:Nn \_stex_notations_styledefs: {
4178   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4179   \str_set:Nn \thisdeclname \l_stex_get_symbol_name_str
4180   \tl_set:Nx \thisdecluri {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4181   \def\thisnotation{
4182     $
4183     \tl_set_eq:NN \l_stex_current_symbol_str\thisdecluri
4184     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{
4185       \_stex_notation_make_args:
4186     }$
4187   }
4188 }
4189
4190 \stex_deactivate_macro:Nn \notation {module-environments}
4191 \stex_every_module:n {\stex_reactivate_macro:N \notation}
4192 \stex_sms_allow_escape:N \notation

```

*(End of definition for `\notation`. This function is documented on page 90.)*

`\stex_notation_parse:n` requires the above keys, `\l_stex_get_symbol_arity_int`, and `\l_stex_get_symbol_arity_tl`

```

4193 \cs_new_protected:Nn \stex_notation_parse:n {
4194   \tl_if_empty:NF \l_stex_key_op_tl {

```

```

4195     \tl_set:Nx \l_stex_key_op_tl { \exp_not:N\maincomp {
4196       \exp_args:No \exp_not:n \l_stex_key_op_tl
4197     } }
4198   }
4199   \seq_clear:N \l__stex_notations_precs_seq
4200   \tl_clear:N \l_stex_notation_args_tl
4201   \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0 {
4202     \__stex_notations_const_precs:
4203     \tl_if_empty:NT \l_stex_key_op_tl {
4204       \tl_set:Nn \l_stex_key_op_tl { \maincomp{#1} }
4205     }
4206   }{
4207     \__stex_notations_fun_precs:
4208     \str_set:Nn \l__stex_notations_missing_str {#1}
4209     \tl_clear:N \l__stex_notations_missing_tl
4210     \stex_map_args:N \__stex_notations_add_missing_args:nn
4211     \tl_if_empty:NT \l_stex_key_op_tl {
4212       \hbox_set:Nn \l_tmpa_box {
4213         \str_set:Nn \l_stex_current_symbol_str {}
4214         \cs_set:Npn \l_tmpa_cs ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
4215         \cs_set:Npn \maincomp ##1 {
4216           \tl_gset:Nn \l_stex_key_op_tl { \maincomp{##1} }
4217           ##1
4218         }
4219         \cs_set:Npn \argsep ##1 ##2 {##1 ##2}
4220         \cs_set:Npn \argmap ##1 ##2 ##3 {##1 ##3}
4221         \cs_set:Npn \argarraymap ##1 ##2 ##3 ##4 {
4222           ##1 ##2
4223         }
4224         \stex_suppress_html:n{${\l_tmpa_cs abcdefghj$}
4225       }
4226     }
4227   }
4228   \exp_args:NNe
4229   \tl_set:Nn \l_stex_notation_macrocode_cs {
4230     \STEXInternalNotation
4231     { \l_stex_key_variant_str }
4232     { \l__stex_notations_opprec_tl }
4233     { \l_stex_key_intent_str }
4234     { \l_stex_notation_args_tl }
4235     {
4236       \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0
4237       { \exp_not:n { \maincomp{ #1 } } }
4238       { \exp_not:n { #1 } \l__stex_notations_missing_tl }
4239     }
4240   }
4241   \stex_debug:nn{notation}{Notation:~\meaning\l_stex_notation_macrocode_cs}
4242 }
4243
4244 \cs_new_protected:Nn \__stex_notations_const_precs: {
4245   \str_if_empty:NTF \l_stex_key_prec_str {
4246     \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
4247   }{
4248     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{

```

```

4249     \tl_set:No \l__stex_notations_opprec_tl { \neginfpref }
4250   }{
4251     \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
4252   }
4253 }
4254 }
4255
4256 \cs_new_protected:Nn \__stex_notations_fun_precs: {
4257   \str_if_empty:NTF \l_stex_key_prec_str {
4258     \tl_set:No \l__stex_notations_opprec_tl { \neginfpref }
4259   }{
4260     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
4261       \tl_set:No \l__stex_notations_opprec_tl { \neginfpref }
4262     }{
4263       \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
4264     }
4265   }
4266   \str_if_empty:NTF \l_stex_key_prec_str {
4267     \tl_set:Nn \l__stex_notations_opprec_tl { 0 }
4268     \int_step_inline:nn \l_stex_get_symbol_arity_int {
4269       \seq_put_right:Nn \l__stex_notations_precs_seq {0}
4270     }
4271   }{
4272     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
4273       \stex_debug:nn{notation}{No~brackets}
4274       \tl_set:No \l__stex_notations_opprec_tl { \neginfpref }
4275       \int_step_inline:nn \l_stex_get_symbol_arity_int {
4276         \exp_args:NNo \seq_put_right:Nn \l__stex_notations_precs_seq \infpref
4277       }
4278     } \__stex_notations_parse_precs:
4279   }
4280   \__stex_notations_do_argnames:
4281 }
4282
4283 \cs_new_protected:Nn \__stex_notations_parse_precs: {
4284   \stex_debug:nn{notation}{parsing~precedence~\l_stex_key_prec_str}
4285   \seq_set_split:NnV \l__stex_notations_seq ; \l_stex_key_prec_str
4286   \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4287     \tl_set_eq:NN \l__stex_notations_opprec_tl \l__stex_notations_str
4288     \seq_pop_left:NNT \l__stex_notations_seq \l__stex_notations_str {
4289       \exp_args:NNo \seq_set_split:NnV \l__stex_notations_seq
4290         {\tl_to_str:n{x}} \l__stex_notations_str
4291     }
4292   }{
4293     \tl_set:No \l__stex_notations_opprec_tl { 0 }
4294   }
4295   \int_step_inline:nn \l_stex_get_symbol_arity_int {
4296     \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4297       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_str
4298     }{
4299       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_opprec_tl
4300     }
4301   }
4302 }

```

```

4303
4304 \cs_new_protected:Nn \__stex_notations_do_argnames: {
4305   \tl_clear:N \l_stex_notation_args_tl
4306   \stex_map_args:N \__stex_notations_do_argname:nn
4307 }
4308
4309 \cs_new_protected:Nn \__stex_notations_do_argname:nn {
4310   \clist_if_empty:NTF \l_stex_key_intent_args_clist {
4311     \tl_put_right:Nx \l_stex_notation_args_tl {
4312       #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}{
4313         \str_if_empty:NF \l_stex_key_intent_str {#1}
4314       }
4315     }
4316   }{
4317     \tl_put_right:Nx \l_stex_notation_args_tl {
4318       #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}
4319       {\c_dollar_str\clist_item:Nn \l_stex_key_intent_args_clist 1}
4320     }
4321     \clist_pop:NN \l_stex_key_intent_args_clist \l_tmpa_tl
4322   }
4323 }
4324
4325 \cs_new:Nn \__stex_notations_add_missing_args:nn {
4326   \exp_args:NNe \str_if_in:NnF \l__stex_notations_missing_str {\c_hash_str\c_hash_str#1}{
4327     \tl_put_right:Nn \l__stex_notations_missing_tl{\STEXinvisible{## #1}}
4328   }
4329 }

```

*(End of definition for \stex\_notation\_parse:n. This function is documented on page ??.)*

```

\stex_notation_check:
  \stex_notation_add:
\stex_notation_do_html:n
\stex_notation_make_args:
4330 \cs_new_protected:Nn \stex_notation_check: {
4331   \stex_check_term:n{
4332     \str_set:Nn \l_stex_current_symbol_str {test}
4333     \cs_set:Npn \comp ##1 {##1}
4334     \stex_debug:nn{check_terms}{Checking~notation...}
4335     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{
4336       \stex_notation_make_args:
4337     }
4338   }
4339 }
4340
4341 \cs_new_protected:Nn \stex_notation_add: {
4342   \stex_module_add_notation:eoeoo{
4343     \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4344     }\l_stex_key_variant_str
4345     {\int_use:N \l_stex_get_symbol_arity_int}
4346     \l_stex_notation_macrocode_cs
4347     \l_stex_key_op_tl
4348   }
4349
4350 \cs_new_protected:Nn \stex_notation_do_html:n {
4351   \hbox{\stex_annotate_invisible:nn {
4352     shtml:notation={#1},

```

```

4353     shtml:notationfragment={\l_stex_key_variant_str},
4354     shtml:precedence={\l__stex_notations_opprec_tl},
4355     shtml:argprecs={\seq_use:Nn \l__stex_notations_precs_seq ,}
4356   }{
4357     \cs_set_protected:Npn \argsep ##1 ##2 {
4358       \stex_annotate:nn{shtml:argsep={}}{
4359         ##1 ##2
4360       }
4361     }
4362     \cs_set_protected:Npn \argmap ##1 ##2 ##3 {
4363       \cs_set:Npn \__stex_notations_map_cs: ####1 { ##2 }
4364       \stex_annotate:nn{shtml:argmap={}}{
4365         \__stex_notations_map_cs:{##1} ##3
4366       }
4367     }
4368     \cs_set_protected:Npn \maincomp {
4369       \_do_comp:nNn {maincomp}\compemph@uri
4370     }
4371     $
4372     \str_set:Nx \l_stex_current_symbol_str {#1}
4373     \stex_annotate:nn{shtml:notationcomp={}}{
4374       \exp_args:Nne \use:nn {
4375         \l_stex_notation_macrocode_cs {}
4376       }{
4377         \stex_map_args:N \__stex_notations_make_arg_html:nn
4378       }
4379     }
4380     $
4381     \tl_if_empty:NF \l_stex_key_op_tl {
4382       $
4383       \str_set:Nx \l_stex_current_symbol_str {#1}
4384       \stex_annotate:nn{shtml:notationopcomp={}}{
4385         \stex_term_oms:nnn{\l_stex_key_variant_str}{}\l_stex_key_op_tl}
4386       }
4387       $
4388     }
4389   }}
4390 }
4391
4392 \cs_new:Nn \__stex_notations_make_arg_html:nn {
4393 % \str_case:nnF #2 {
4394 %   a {{
4395 %     \stex_annotate:nn{shtml:argnum=#1a}{x},
4396 %     \stex_annotate:nn{shtml:argnum=#1b}{x}
4397 %   }}
4398 %   B {{
4399 %     \stex_annotate:nn{shtml:argnum=#1a}{x},
4400 %     \stex_annotate:nn{shtml:argnum=#1b}{x}
4401 %   }}
4402 % }{
4403 % {
4404 %   \stex_annotate:nn{shtml:argnum=#1}{x}
4405 % }
4406 % }

```

```

4407 }
4408
4409 \cs_new:Nn \_stex_notation_make_args: {
4410   \_stex_map_notation_args:N \_stex_notations_make_arg:nnnn
4411 }
4412
4413
4414 \cs_new:Nn \_stex_notations_make_arg:nnnn {
4415   \str_case:nnF #2 {
4416     a {{
4417       a\c_math_subscript_token{#1,1},
4418       a\c_math_subscript_token{#1,2}
4419     }}
4420     B {{
4421       B\c_math_subscript_token{#1,1},
4422       B\c_math_subscript_token{#1,2}
4423     }}
4424   }{
4425     \_stex_term_arg:nnnn{#1}{#2}{#3}{#4}
4426     {#2}\c_math_subscript_token{#1}}
4427   }
4428 }

```

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

### **\setnotation**

```

\_stex_notation_set_default:n 4429 \cs_new_protected:Npn \setnotation #1 #2 {
4430   \stex_get_symbol:n{#1}
4431   \cs_if_exist:cTF{l_stex_notation_
4432     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4433     _#2_cs
4434   }{
4435     \tl_set_eq:Nc \l_stex_notation_macrocode_cs {l_stex_notation_
4436       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4437       _#2_cs
4438     }
4439     \cs_if_exist:cTF{l_stex_notation_
4440       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4441       _op_#2_cs
4442     }{
4443       \tl_set_eq:Nc \l_stex_key_op_tl {l_stex_notation_
4444         \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4445         _op_#2_cs
4446       }
4447     }{
4448       \tl_clear:N \l_stex_key_op_tl
4449     }
4450     \_stex_notation_set_default:n{
4451       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4452     }
4453   }{
4454     \msg_error:nnxx{stex}{unknownnotation}{#2}{
4455       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str

```

```

4456     }
4457   }
4458 }
4459
4460 \cs_new_protected:Nn \_stex_notation_set_default:n{
4461   \stex_if_in_module:TF{
4462     \stex_module_add_notation:eoeoo{#1}{ }
4463     {\int_use:N \l_stex_get_symbol_arity_int}
4464     \l_stex_notation_macrocode_cs
4465     \l_stex_key_op_tl
4466   }{
4467     \cs_set_eq:cN {l_stex_notation_
4468       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4469       _cs}\l_stex_notation_macrocode_cs
4470     \tl_if_empty:NF \l_stex_key_op_tl {
4471       \cs_set_eq:cN{l_stex_notation_
4472         \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4473         _op__cs}\l_stex_key_op_tl
4474     }
4475   }
4476 }

```

(End of definition for `\setnotation` and `\_stex_notation_set_default:n`. These functions are documented on page 91.)

#### `\varnotation`

```

4477 \stex_new_stylable_cmd:nnnn {varnotation} { s m 0{} m} {
4478   \stex_keys_set:nn{notation}{#3}
4479   \stex_get_var:n{#2}
4480   \str_set_eq:NN \l_stex_key_name_str \l_stex_get_symbol_name_str
4481   \stex_notation_parse:n{#4}
4482   \stex_if_check_terms:T{ \_stex_notation_check: }
4483   \_stex_vardecl_notation_macro:
4484   \IfBooleanTF#1{
4485     \_stex_notation_set_default:n{\l_stex_get_symbol_name_str}
4486   }{ }
4487   \group_begin:
4488   \tl_set_eq:NN \thisvarname \l_stex_get_symbol_name_str
4489   \tl_clear:N \thisstyle
4490   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4491   \def\thisnotation{
4492     $\let\l_stex_current_symbol_str\thisvarname
4493     \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{ }{
4494       \_stex_notation_make_args:
4495     }$
4496   }
4497   \stex_style_apply:
4498   \group_end:
4499 }{ }

```

(End of definition for `\varnotation`. This function is documented on page 95.)

#### `\symdef`

```

4500 \stex_keys_define:nnnn{symdef}{ }{ }{decl,notation}
4501

```



```

4502 \cs_new_protected:Nn \_stex_symdef_styledefs: {
4503   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
4504   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
4505   \tl_set_eq:NN \thistype \l_stex_key_type_tl
4506   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
4507   \tl_set_eq:NN \thisargs \l_stex_key_args_str
4508   \tl_clear:N \thisstyle
4509   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4510   \def\thisnotation{
4511     $\let\l_stex_current_symbol_str\thisdecluri
4512     \def\comp{\_comp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{
4513       \_stex_notation_make_args:
4514     }$
4515   }
4516 }
4517
4518 \stex_new_stylable_cmd:nnnn {symdef} { m 0{ } m} {
4519   \stex_keys_set:nn{symdef}{#2}
4520   \str_set:Nx \l_stex_macroname_str { #1 }
4521   \stex_symdecl_top:n{#1}
4522   \stex_debug:nn{symdef}{Doing~\l_stex_current_module_str ? \l_stex_key_name_str}
4523   \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
4524   \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
4525   \stex_notation_parse:n{#3}
4526   \stex_debug:nn{Here!}{\meaning\l_stex_notation_args_tl}
4527   \_stex_notation_check:
4528   \_stex_notation_add:
4529   \stex_if_do_html:T{
4530     \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4531   }
4532   \stex_if_smsmode:F{
4533     \group_begin:
4534     \_stex_symdef_styledefs:
4535     \stex_style_apply:
4536     \group_end:
4537   }
4538   \stex_smsmode_do:
4539 }{}
4540
4541 \stex_deactivate_macro:Nn \symdef {module~environments}
4542 \stex_every_module:n {\stex_reactivate_macro:N \symdef}
4543 \stex_sms_allow_escape:N \symdef

```

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

`\stex_do_default_notation_op:`

```

4544 \cs_new_protected:Nn \stex_do_default_notation: {
4545   \stex_do_default_notation_op:
4546   \tl_if_empty:NTF \l_stex_current_args_tl {
4547     \tl_clear:N \l__stex_notations_args_tl
4548   }{
4549     \__stex_notations_make_name:
4550     \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
4551     \tl_set:Nx \l__stex_notations_args_tl {

```

```

4552     \stex_map_args:N \stex_notations_augment_arg:nn
4553   }
4554   \tl_put_right:Nn \l_stex_default_notation {\comp()}
4555   \seq_clear:N \l_tmpa_seq
4556   \int_step_inline:nn \l_stex_current_arity_str {
4557     \seq_put_right:Nn \l_tmpa_seq {#### #1}
4558   }
4559   \tl_put_right:Nx \l_stex_default_notation {
4560     \seq_use:Nn \l_tmpa_seq {\mathpunct{\comp{,}}}
4561   }
4562   \tl_put_right:Nn \l_stex_default_notation {\comp}}
4563 }
4564 \tl_set:Nx \l_stex_default_notation {\STEXInternalNotation{}{0}{}\l_stex_notations_args}
4565 \exp_args:No \exp_not:n \l_stex_default_notation
4566 }}
4567 }
4568
4569 \cs_new:Nn \stex_notations_augment_arg:nn {
4570   #1#2{0}{}
4571 }
4572
4573 \cs_new_protected:Nn \stex_notations_make_name: {
4574   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq ? \l_stex_current_symbol_str
4575   \seq_pop_right:NN \l_tmpa_seq \l_stex_notations_name_str
4576   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq / \l_stex_notations_name_str
4577   \seq_pop_right:NN \l_tmpa_seq \l_stex_notations_name_str
4578 }
4579
4580 \cs_new_protected:Nn \stex_do_default_notation_op: {
4581   \stex_notations_make_name:
4582   \tl_set:Nx \l_stex_default_notation {\exp_not:N \maincomp{ \exp_not:N \mathrm {\l_stex_no
4583 }

```

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

## `\STEXInternalNotation`

```

4584 % 1: variant 2: operator precedence 3: intent 4: arguments 5: code 6: next
4585
4586 \cs_new_protected:Npn \STEXInternalNotation #1 #2 #3 #4 #5 #6 {
4587   \stex_notations_process_notation:nnnnn{#1}{#2}{#3}{#4}{#5}{
4588     \l_stex_notations_code_tl
4589     #6
4590   }
4591 }
4592
4593 \cs_new_protected:Npn \stex_notations_process_notation:nnnnn #1 #2 #3 #4 {
4594   \tl_if_empty:nTF{#4}{
4595     \stex_notations_simple:nnnn{#1}{#2}{#3}
4596   }{
4597     \stex_notations_complex:nnnnn{#1}{#2}{#3}{#4}
4598   }
4599 }
4600
4601 \cs_new_protected:Nn \stex_notations_simple:nnnn {

```

```

4602 \stex_debug:nn{Notation~code}{\tl_to_str:n{#4}}
4603 \tl_set:Nn \l__stex_notations_code_tl {
4604   \cs_set:Npn \l__stex_notations_cs {
4605     \stex_maybe_brackets:nn{#2}{
4606       \stex_term_oms_or_omv:nnn{#1}{#3}{#4}
4607     }
4608   }
4609   \l__stex_notations_cs
4610 }
4611 \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4612 #5
4613 }
4614
4615 \cs_new_protected:Nn \__stex_notations_complex:nnnnn {
4616   \stex_debug:nn{Notation~code}{\tl_to_str:n{#5}}
4617   \int_zero:N \l_tmpa_int
4618   \tl_set:Nn \l__stex_notations_pre_tl {\cs_set_eq:NN \stex_term_oma_or_omb:nnn \stex_term_
4619   \tl_set:Nn \l__stex_notations_code_tl {
4620     \cs_generate_from_arg_count:NNnn \l__stex_notations_cs \cs_set:Npn \l_tmpa_int
4621     {
4622       \stex_maybe_brackets:nn{#2}{
4623         \stex_term_oma_or_omb:nnn{#1}{#3}{
4624           \bool_set_false:N \l_stex_brackets_dones_bool
4625           #5
4626         }
4627       }
4628     }
4629     \l__stex_notations_cs
4630   }
4631   \tl_set:Nn \l__stex_notations_after_tl{
4632     \exp_args:NNo
4633     \tl_put_left:Nn \l__stex_notations_code_tl \l__stex_notations_pre_tl
4634     \tl_put_left:Nx \l__stex_notations_code_tl {
4635       \int_set:Nn \l_tmpa_int {\int_use:N \l_tmpa_int}
4636     }
4637     \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4638     #6
4639   }
4640   \__stex_notations_parse_notation_args:nnnw #4 \__stex_notations_args_end:
4641 }
4642
4643 \cs_new_protected:Npn \__stex_notations_parse_notation_args:nnnw #1 #2 #3 #4 #5 \__stex_not
4644 \tl_if_empty:nTF{#5}{
4645   \__stex_notations_add_last:nnnn{#1}{#2}{#3}{#4}
4646 }{
4647   \__stex_notations_add_next:nnnnn{#1}{#2}{#3}{#4}{#5}
4648 }
4649 }
4650
4651 \cs_new_protected:Nn \__stex_notations_add_next:nnnnn {
4652   \__stex_notations_add:nnnn{#1}{#2}{#3}{#4}{#6}
4653   \__stex_notations_parse_notation_args:nnnw #5 \__stex_notations_args_end:
4654 }
4655

```

```

4656 \cs_new_protected:Nn \l__stex_notations_add_last:nnnnn {
4657   \l__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#5}
4658   \l__stex_notations_after_tl
4659 }
4660
4661 \cs_new_protected:Nn \l__stex_notations_add:nnnnn {
4662   \int_incr:N \l_tmpa_int
4663   \str_case:nn{#2}{
4664     i {
4665       \tl_put_right:Nn \l__stex_notations_code_tl {
4666         {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4667       }
4668     }
4669     b {
4670       \tl_set:Nn \l__stex_notations_pre_tl {
4671         \cs_set_eq:NN \_stex_term_oma_or_omb:nnn \_stex_term_omb:nnn
4672       }
4673       \tl_put_right:Nn \l__stex_notations_code_tl {
4674         {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4675       }
4676     }
4677     a {
4678       \tl_put_right:Nn \l__stex_notations_code_tl {
4679         {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4680       }
4681     }
4682     B {
4683       \tl_set:Nn \l__stex_notations_pre_tl {
4684         \cs_set_eq:NN \_stex_term_oma_or_omb:nnn \_stex_term_omb:nnn
4685       }
4686       \tl_put_right:Nn \l__stex_notations_code_tl {
4687         {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4688       }
4689     }
4690   }
4691 }

```

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

## a/B-mode argument handling

`\argsep`

```

4692 \cs_new_protected:Nn \l__stex_notations_check_aB_arg:Nn {
4693   \exp_args:Ne \cs_if_eq:NNF {\tl_head:n{#2}}
4694   \_stex_term_arg_aB:nnnnn {
4695     \msg_error:nnx{stex}{error/assocarg}{\tl_to_str:n{#1}}
4696   }
4697 }
4698
4699 \cs_new_protected:Npn \argsep #1 #2 {
4700   \l__stex_notations_check_aB_arg:Nn\argsep{#1}
4701   \stex_pseudogroup_with:nn{\_stex_term_do_aB_clist:}{
4702     \tl_set:Nn \_stex_term_do_aB_clist: {
4703       \seq_use:Nn \l_stex_aB_args_seq {#2}

```

```

4704     }
4705     #1
4706   }
4707 }

```

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

### `\argmap`

```

4708 \cs_new_protected:Npn \argmap #1 #2 #3 {
4709   \__stex_notations_check_aB_arg:Nn\argmap{#1}
4710   \stex_pseudogroup_with:nn{
4711     \stex_term_do_aB_clist:
4712     \__stex_notations_map_cs:
4713   }{
4714     \cs_set:Npn \__stex_notations_map_cs: ##1 { #2 }
4715     \tl_set:Nn \stex_term_do_aB_clist: {
4716       \seq_clear:N \l_tmpa_seq
4717       \seq_map_inline:Nn \l_stex_aB_args_seq {
4718         \tl_if_eq:nnTF{##1}{\ellipses}{
4719           \seq_put_right:Nn \l_tmpa_seq \ellipses
4720         }{
4721           \seq_put_right:Nx \l_tmpa_seq {
4722             \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
4723           }
4724         }
4725       }
4726       \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4727       \seq_use:Nn \l_stex_aB_args_seq {#3}
4728     }
4729     #1
4730   }
4731 }

```

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

### `\argarraymap`

```

4732 \int_new:N \l__stex_notations_clist_count_int
4733 \cs_new_protected:Npn \argarraymap #1 #2 #3 #4 {
4734   \__stex_notations_check_aB_arg:Nn\argarraymap{#1}
4735   \stex_pseudogroup_with:nn{
4736     \stex_term_do_aB_clist:
4737     \__stex_notations_map_cs:
4738   }{
4739     \cs_set:Npn \__stex_notations_map_cs: ##1 { #3 }
4740     \int_set:Nn \l__stex_notations_clist_count_int {\exp_args:No\clist_count:n{\tl_to_str:n{
4741       \tl_set:Nn \stex_term_do_aB_clist: {
4742         \tl_clear:N \l_tmpa_tl
4743         \int_zero:N \l_tmpa_int
4744         \seq_map_inline:Nn \l_stex_aB_args_seq {
4745           \int_incr:N \l_tmpa_int
4746           \int_compare:nNnT \l_tmpa_int > \l__stex_notations_clist_count_int {
4747             \int_set:Nn \l_tmpa_int 1
4748           }
4749           \tl_put_right:Nx \l_tmpa_tl {
4750             \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }

```

```

4751         \clist_item:nn{#4}\l_tmpa_int
4752     }
4753 }
4754 \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4755 \begin{array}{#2}
4756     \l_tmpa_tl
4757 \end{array}
4758 }
4759 #1
4760 }
4761 }

```

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

### 13.8.3 Variables

```

4762 <@@=stex_vars>

```

`\vardef`

```

4763 \tl_new:N \l_stex_variables_prop
4764 \bool_new:N \l__stex_vars_bind_bool
4765 \cs_new_protected:Nn \_stex_variable:nnnnnnN {}
4766 \stex_keys_define:nnnn{\vardef}{
4767     \bool_set_false:N \l__stex_vars_bind_bool
4768 }{
4769     bind .bool_set:N = \l__stex_vars_bind_bool
4770 }{symdef}
4771
4772 \stex_new_stylable_cmd:nnnn {\vardef} { m 0{ } m } {
4773     \stex_keys_set:nn{\vardef}{#2}
4774     \str_set:Nx \l_stex_macroname_str { #1 }
4775     \str_if_empty:NT \l_stex_key_name_str {
4776         \str_set:Nx \l_stex_key_name_str { #1 }
4777     }
4778
4779     \stex_symdecl_do:
4780     \_stex_symdecl_check_terms:
4781     \__stex_vars_add:
4782     \__stex_vars_macro:
4783     \stex_if_do_html:T \__stex_vars_html:
4784
4785     \int_set:Nn \l_stex_get_symbol_arity_int {\l_stex_get_symbol_arity_int}
4786     \stex_debug:nn{\vardef}{Doing~\l_stex_key_name_str}
4787     \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4788     \stex_notation_parse:n{#3}
4789     \stex_if_check_terms:T{ \_stex_notation_check: }
4790     \_stex_vardecl_notation_macro:
4791     \stex_if_do_html:T {
4792         \def\comp{\_varcomp}
4793         \_stex_notation_do_html:n \l_stex_key_name_str
4794     }
4795     \group_begin:
4796     \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4797     \tl_clear:N \thisstyle
4798     \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str

```

```

4799 \def\thisnotation{
4800   $\let\l_stex_current_symbol_str\thisvarname
4801   \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}{
4802     \_stex_notation_make_args:
4803   }$
4804 }
4805 \stex_style_apply:
4806 \group_end:\ignorespaces
4807 }{}
4808
4809 \cs_new_protected:Nn \__stex_vars_add: {
4810   \exp_args:NNNo \exp_args:NNnx
4811   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4812     {\l_stex_macroname_str}
4813     {\l_stex_key_name_str}
4814     {\int_use:N \l_stex_get_symbol_arity_int}
4815     {\l_stex_get_symbol_args_tl}
4816     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4817     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4818     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4819     \stex_invoke_symbol:
4820   }
4821 }
4822
4823 \cs_new_protected:Nn \__stex_vars_macro: {
4824   \tl_set:cx{\l_stex_macroname_str}{
4825     \_stex_invoke_variable:nnnnnN
4826     {\l_stex_key_name_str}
4827     {\int_use:N \l_stex_get_symbol_arity_int}
4828     {\l_stex_get_symbol_args_tl}
4829     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4830     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4831     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4832     \stex_invoke_symbol:
4833   }
4834 }
4835
4836 \cs_new_protected:Nn \__stex_vars_html: {
4837   \stex_if_do_html:T {
4838     \hbox\bgroup\exp_args:Ne \stex_annotate_invisible:nn {
4839       shtml:vardef = {\l_stex_key_name_str},
4840       shtml:args = {\l_stex_key_args_str}
4841       \str_if_empty:NF \l_stex_macroname_str {,
4842         shtml:macroname={\l_stex_macroname_str}
4843       }
4844       \str_if_empty:NF \l_stex_key_assoc_str {,
4845         shtml:assoctype={\l_stex_key_assoc_str}
4846       }
4847       \str_if_empty:NF \l_stex_key_role_str {,
4848         shtml:role={\l_stex_key_role_str}
4849       }
4850       \str_if_empty:NF \l_stex_key_reorder_str {,
4851         shtml:reorderargs={\l_stex_key_reorder_str}
4852       }

```

```

4853     \bool_if:NT \l__stex_vars_bind_bool {,
4854         shtml:bind={}
4855     }
4856 }{
4857     \_stex_annotate_force_break:n{
4858         \bool_set_true:N \stex_in_invisible_html_bool
4859         \tl_if_empty:NF \l_stex_key_type_tl {
4860             \stex_annotate:nn{shtml:type={}}{ $\l_stex_key_type_tl$ }
4861         }
4862         \tl_if_empty:NF \l_stex_key_def_tl {
4863             \stex_annotate:nn{shtml:definens={}}{ $\l_stex_key_def_tl$ }
4864         }
4865         \tl_if_empty:NF \l_stex_key_return_tl{
4866             \exp_args:Nno \use:n{
4867                 \cs_generate_from_arg_count:NNnn \l__stex_vars_cs
4868                 \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4869                 \tl_set:Nx \l__stex_vars_args_tl {\_stex_map_args:N \_stex_return_args:nn}
4870                  $\stex_annotate:nn{shtml:returntype={}}{$ 
4871                     \exp_after:wN \l__stex_vars_cs \l__stex_vars_args_tl!}$
4872             }
4873             \tl_if_empty:NF \l_stex_key_argtypes_clist {
4874                 \stex_annotate:nn{shtml:argtypes={}}{
4875                     \_stex_annotate_force_break:n{
4876                         \clist_map_inline:Nn \l_stex_key_argtypes_clist {
4877                              $\stex_annotate:nn{shtml:type={}}{##1}$ 
4878                         }
4879                     }
4880                 }
4881             }
4882         }
4883     }\egroup
4884 }
4885 }

```

(End of definition for `\vardef`. This function is documented on page 95.)

`\_stex_vardecl_notation_macro:`

```

4886 \cs_new_protected:Nn \_stex_vardecl_notation_macro: {
4887     \tl_set_eq:cN {l_stex_notation_
4888         \l_stex_key_name_str _
4889         \l_stex_key_variant_str _cs
4890     }\l_stex_notation_macrocode_cs
4891     \cs_if_exist:cF {l_stex_notation_\l_stex_key_name_str __cs}{
4892         \tl_set_eq:cN{l_stex_notation_\l_stex_key_name_str __cs}
4893         \l_stex_notation_macrocode_cs
4894     }
4895     \tl_if_empty:NF \l_stex_key_op_tl {
4896         \tl_set_eq:cN {l_stex_notation_\l_stex_key_name_str _op_
4897             \l_stex_key_variant_str _cs}\l_stex_key_op_tl
4898         \cs_if_exist:cF {l_stex_notation_\l_stex_key_name_str _op__cs}{
4899             \cs_set_eq:cN{l_stex_notation_\l_stex_key_name_str _op__cs}
4900             \l_stex_key_op_tl
4901         }
4902     }
4903 }

```



(End of definition for `\stex_vardecl_notation_macro`:. This function is documented on page 125.)

`\stex_get_symbol_or_var:n`

`\stex_get_var:n`

```

4904 \cs_new_protected:Nn \__stex_vars_set_vars:nnnnnnN {
4905   \stex_debug:nn{symbols}{Variable~#1~found}
4906   \cs_set:Npn \stex_variable:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {}
4907   \str_clear:N \l_stex_get_symbol_mod_str
4908   \str_set:Nn \l_stex_get_symbol_name_str {#1}
4909   \int_set:Nn \l_stex_get_symbol_arity_int {#2}
4910   \tl_set:Nn \l_stex_get_symbol_args_tl {#3}
4911   \tl_set:Nn \l_stex_get_symbol_def_tl {#4}
4912   \tl_set:Nn \l_stex_get_symbol_type_tl {#5}
4913   \tl_set:Nn \l_stex_get_symbol_return_tl {#6}
4914   \tl_set:Nn \l_stex_get_symbol_invoke_cs {#7}
4915 }
4916
4917 \cs_new_protected:Nn \__stex_vars_get_var:n {
4918   \prop_map_inline:Nn \l_stex_variables_prop {
4919     \__stex_vars_check_var:nnnnnnnnN {#1} ##2
4920   }
4921 }
4922
4923 \cs_new_protected:Nn \__stex_vars_check_var:nnnnnnnnN {
4924   \str_if_eq:nnTF{#1}{#2}{
4925     \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}}
4926   }{
4927     \str_if_eq:nnT{#1}{#3}{
4928       \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}}
4929     }
4930   }
4931 }
4932
4933 \cs_new_protected:Nn \stex_get_var:n {
4934   \str_clear:N \l_stex_get_symbol_name_str
4935   \__stex_vars_get_var:n{#1}
4936   \str_if_empty:NT \l_stex_get_symbol_name_str {
4937     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4938   }
4939 }
4940
4941 \cs_new_protected:Nn \stex_get_symbol_or_var:n {
4942   \str_clear:N \l_stex_get_symbol_name_str
4943   \__stex_vars_get_var:n{#1}
4944   \str_if_empty:NT \l_stex_get_symbol_name_str {
4945     \stex_debug:nn{symbols}{No~variable~#1~found}
4946     \stex_get_symbol:n{#1}
4947   }
4948 }

```

(End of definition for `\stex_get_symbol_or_var:n` and `\stex_get_var:n`. These functions are documented on page 125.)

`\svar`

```

4949 \NewDocumentCommand \svar {0{ } m}{

```

```

4950 \group_begin:
4951   \tl_if_empty:nTF{#1}{
4952     \str_set:Nn \l_stex_current_symbol_str {#2}
4953   }{
4954     \str_set:Nn \l_stex_current_symbol_str {#1}
4955   }
4956   \bool_if:NTF \l_stex_allow_semantic_bool{
4957     \tl_clear:N \l_stex_current_term_tl
4958     \stex_term_omv:nnn{}{}{\_varcomp{#2}}
4959   }{
4960     \msg_error:nxxx{stex}{error/notallowed}{Variable}{\l_stex_current_symbol_str}
4961   }
4962 \group_end:
4963 }

```

(End of definition for `\svar`. This function is documented on page 95.)

### 13.8.4 Sequences

```

4964 <@@=stex_seqs>

```

`\varseq`

```

4965 \stex_new_stylable_cmd:nnnn {varseq}{m 0{} m m} {
4966   \stex_keys_set:nn{symdef}{#2}
4967   \str_set:Nx \l_stex_macroname_str { #1 }
4968   \str_if_empty:NT \l_stex_key_name_str {
4969     \str_set:Nx \l_stex_key_name_str { #1 }
4970   }
4971   \str_if_empty:NT \l_stex_key_args_str {
4972     \str_set:Nn \l_stex_key_args_str {1}
4973   }
4974   \stex_symdecl_do:
4975
4976   \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4977   \clist_set:Nn \l__stex_seqs_range_clist {#3}
4978   \tl_if_empty:NTF \l_stex_key_op_tl {
4979     \stex_notation_parse:n{#4}
4980     \tl_clear:N \l_stex_key_op_tl
4981   }{
4982     \stex_notation_parse:n{#4}
4983   }
4984   \stex_if_do_html:T \__stex_seqs_html:
4985   \stex_if_check_terms:T \__stex_seqs_check_terms:
4986   \__stex_seqs_add:
4987   \__stex_seqs_macro:
4988   \stex_if_check_terms:T \stex_notation_check:
4989   \stex_vardecl_notation_macro:
4990   \group_begin:
4991   \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4992   \tl_clear:N \thisstyle
4993   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4994   \def\thisnotation{
4995     $\let\l_stex_current_symbol_str\thisvarname
4996     \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}}{
4997     \__stex_seqs_make_args:

```

```

4998     }$
4999   }
5000   \stex_style_apply:
5001   \group_end:\ignorespaces
5002 }{}
5003
5004 \cs_new_protected:Nn \__stex_seqs_add: {
5005   \exp_args:NNNo \exp_args:NNnx
5006   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
5007     {\l_stex_macroname_str}
5008     {\l_stex_key_name_str}
5009     {\int_use:N \l_stex_get_symbol_arity_int}
5010     {\l_stex_get_symbol_args_tl}
5011     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
5012     {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
5013     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
5014     \stex_invoke_sequence:
5015   }
5016 }
5017
5018 \cs_new_protected:Nn \__stex_seqs_macro: {
5019   \tl_set:cx{\l_stex_macroname_str}{
5020     \stex_invoke_variable:nnnnnnN
5021     {\l_stex_key_name_str}
5022     {\int_use:N \l_stex_get_symbol_arity_int}
5023     {\l_stex_get_symbol_args_tl}
5024     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
5025     {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
5026     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
5027     \stex_invoke_sequence:
5028   }
5029 }
5030
5031 \cs_new_protected:Nn \__stex_seqs_make_args: { \TODO }
5032 \cs_new_protected:Nn \__stex_seqs_check_terms: { \TODO }
5033
5034 \cs_new_protected:Nn \__stex_seqs_html: {
5035   \exp_args:Ne \stex_annotate_invisible:nn {
5036     shtml:vargseq = {\l_stex_key_name_str},
5037     shtml:args = {\l_stex_key_args_str}
5038     \str_if_empty:NF \l_stex_macroname_str {,
5039       shtml:macroname={\l_stex_macroname_str}
5040     }
5041     \str_if_empty:NF \l_stex_key_assoc_str {,
5042       shtml:assoctype={\l_stex_key_assoc_str}
5043     }
5044     \str_if_empty:NF \l_stex_key_role_str {,
5045       shtml:role={\l_stex_key_role_str}
5046     }
5047     \str_if_empty:NF \l_stex_key_reorder_str {,
5048       shtml:reorderargs={\l_stex_key_reorder_str}
5049     }
5050   }{\hbox\bgroup
5051     \stex_annotate_force_break:n{

```

```

5052     \tl_if_empty:NF \l_stex_key_type_tl {
5053         \stex_annotate:nn{shtml:type={}}{${\l_stex_key_type_tl$}
5054     }
5055     \tl_if_empty:NF \l_stex_key_def_tl {
5056         \stex_annotate:nn{shtml:definien={}}{${\l_stex_key_def_tl$}
5057     }
5058     \tl_if_empty:NF \l_stex_key_return_tl{
5059         \exp_args:Nno \use:n{
5060             \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs
5061             \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
5062         \tl_set:Nx \l__stex_seqs_args_tl {\_stex_map_args:N \_stex_return_args:nn}
5063         \stex_annotate:nn{shtml:returtype={}}{
5064             $\exp_after:wN \l__stex_seqs_cs \l__stex_seqs_args_tl!$}
5065     }
5066     \tl_if_empty:NF \l_stex_key_argtypes_clist {
5067         \stex_annotate:nn{shtml:argtypes={}}{
5068             \_stex_annotate_force_break:n{
5069                 \clist_map_inline:Nn \l_stex_key_argtypes_clist {
5070                     \stex_annotate:nn{shtml:type={}}{${##1$}
5071                 }
5072             }
5073         }
5074     }
5075 }
5076 \egroup}
5077 }

```

(End of definition for `\varseq`. This function is documented on page 95.)

`\stex_invoke_sequence:`

```

5078 \cs_new_protected:Nn \stex_invoke_sequence: {
5079     \peek_charcode_remove:NTF ! {
5080         \peek_charcode:NTF [ \_stex_seqs_do_op:w { \_stex_seqs_do_op:w [] }
5081     } \_stex_seqs_do_first:
5082 }
5083
5084 \cs_new_protected:Npn \_stex_seqs_do_op:w [#1] {
5085     \cs_if_exist:cTF {l_stex_notation\_l_stex_current_symbol_str_op_#1_cs}{
5086         \_stex_maybe_brackets:nn{\neginfprec}{
5087             \_stex_term_oms_or_omv:nn{#1}{}
5088             {\use:c{l_stex_notation\_l_stex_current_symbol_str_op_#1_cs}}
5089         }
5090     } \group_end:
5091 }{
5092     \_stex_seqs_get_index_notation:n{#1}
5093     \peek_charcode:NTF [ \_stex_seqs_doop_range:w { \_stex_seqs_doop_range:w [] }
5094 }
5095 }
5096
5097 \cs_new_protected:Npn \_stex_seqs_doop_range:w [#1] {
5098     \bool_set_true:N \l_stex_allow_semantic_bool
5099     \clist_clear:N \l__stex_seqs_clist
5100     \clist_map_function:NN \l_stex_current_type_tl \_stex_seqs_doop_arg:n
5101     \stex_annotate:nn{

```

```

5102     shtml:term=OMV,
5103     shtml:head={\l_stex_current_symbol_str},
5104     shtml:notationid={ }
5105   }{
5106     \l__stex_seqs_clist
5107   }
5108   \group_end:
5109 }
5110
5111 \cs_new_protected:Nn \__stex_seqs_doop_arg:n {
5112   \tl_if_eq:nnTF{#1}{\ellipses}{
5113     \clist_put_right:Nn \l__stex_seqs_clist {
5114       \ellipses
5115     }
5116   }{
5117     \clist_put_right:Nn \l__stex_seqs_clist {
5118       \exp_args:No \str_if_eq:nnTF \l_stex_current_arity_str {1}{
5119         \group_begin:
5120         \l__stex_seqs_cs \group_end: {#1}
5121       }
5122     }{
5123       \group_begin:
5124       \l__stex_seqs_cs \group_end: #1
5125     }
5126   }
5127 }
5128 }
5129
5130 \cs_new_protected:Nn \__stex_seqs_get_index_notation:n {
5131   \cs_if_exist:cTF {l_stex_notation_\l_stex_current_symbol_str _#1_cs}{
5132     \cs_set_eq:Nc \l__stex_seqs_cs {l_stex_notation_\l_stex_current_symbol_str _#1_cs}
5133   }{
5134     \stex_do_default_notation:
5135     \cs_set_eq:NN \l__stex_seqs_cs \l_stex_default_notation
5136   }
5137 }
5138
5139
5140 \cs_new:Nn \__stex_seqs_do_first_arg:n {{\exp_not:n{## #1}}}
5141
5142 \cs_new_protected:Nn \__stex_seqs_do_first: {
5143   \exp_args:Nnx \use:nn{
5144     \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs \cs_set:Npn
5145     \l_stex_current_arity_str} {{
5146     \tl_set:Nn \exp_not:N \l__stex_seqs_first_args_tl {
5147       \int_step_function:nN \l_stex_current_arity_str \__stex_seqs_do_first_arg:n
5148     }
5149     \exp_not:N \__stex_seqs_do_first_next:
5150   }}
5151   \l__stex_seqs_cs
5152 }
5153
5154 \cs_new_protected:Nn \__stex_seqs_do_first_next: {
5155   \peek_charcode_remove:NTF ! {

```

```

5156     \peek_charcode:NTF [ \__stex_seqs_do_one:w {\__stex_seqs_do_one:w []}
5157   }{
5158     \peek_charcode:NTF [ \__stex_seqs_do_all:w {\__stex_seqs_do_all:w []}
5159   }
5160 }
5161
5162 \cs_new_protected:Npn \__stex_seqs_do_one:w [#1] {
5163   \__stex_seqs_get_index_notation:n{#1}
5164   \stex_debug:nn{HERE~seq~one}{\meaning\l__stex_seqs_cs^^J\meaning\l__stex_seqs_first_args_tl}
5165   \exp_args:Nno\use:nn{\l__stex_seqs_cs\group_end:}\l__stex_seqs_first_args_tl
5166 }
5167
5168 \cs_new_protected:Npn \__stex_seqs_do_all:w [#1] {
5169   \stex_debug:nn{HERE~seq~all}{\meaning\l__stex_seqs_first_args_tl}
5170   \exp_args:Nno\use:nn{\stex_invoke_notation:w [#1]}\l__stex_seqs_first_args_tl
5171 }

```

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

### \seqmap

```

5172 \cs_new_protected:Npn \seqmap #1 #2 {
5173   \symuse{Metatheory?sequence~expression}{\seqmap{#1}{#2}}%\l_tmpa_tl {#2}
5174 }

```

(End of definition for `\seqmap`. This function is documented on page 96.)

## 13.8.5 Expressions

```

5175 <@@=stex_expr>

```

Various variables:

```

5176 \bool_new:N \l_stex_allow_semantic_bool
5177 \bool_set_true:N \l_stex_allow_semantic_bool
5178 \tl_new:N \l_stex_current_term_tl
5179 \tl_set:Nn \l_stex_every_symbol_tl {
5180   \bool_set_false:N \l_stex_allow_semantic_bool
5181 }

```

### `\_stex_next_symbol:n`

```

5182 \tl_new:N \l__stex_expr_reset_tl
5183 \cs_new_protected:Nn \_stex_next_symbol:n {
5184   \tl_set:Nx \l_stex_every_symbol_tl {
5185     \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5186       \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
5187         \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5188       }
5189       \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5190         \exp_args:No \exp_not:n \l__stex_expr_reset_tl
5191       }
5192     }
5193     \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
5194       \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5195     }
5196     \exp_not:n{ \aftergroup \l__stex_expr_reset_tl }
5197     \exp_not:N \l_stex_every_symbol_tl

```

```

5198   \exp_not:n{ #1 }
5199   }
5200 }
5201 \cs_generate_variant:Nn \stex_next_symbol:n {e}

```

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

## `\STEXinvisible`

```

5202 \cs_new_protected:Npn \STEXinvisible #1 {
5203   \stex_annotate_invisible:n { #1 }
5204 }

```

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

## Invoking Semantic Macros

`\stex_invoke_symbol:nnnnnnN`

```

5205 \cs_new_protected:Nn \stex_invoke_symbol:nnnnnnN {
5206   \bool_if:NTF \l_stex_allow_semantic_bool{
5207     \stex_if_html_backend:T{\ifvmode\indent\fi}
5208     \__stex_expr_setup:nnnnnn{\_comp}{#1?#2}{#3}{#4}{#7}{#6}
5209     \cs_set_eq:NN \stex_term_oms_or_omv:nnn \stex_term_oms:nnn
5210     \tl_put_right:Nn \l_stex_current_redo_tl{
5211       \cs_set_eq:NN \stex_term_oms_or_omv:nnn \stex_term_oms:nnn
5212     }
5213     #8
5214   }{
5215     \msg_error:nnxx{stex}{error/notallowed}{#1?#2}{\l_stex_current_symbol_str}
5216   }
5217 }
5218 \cs_generate_variant:Nn \stex_invoke_symbol:nnnnnnN {ooxooooN}
5219
5220 \cs_new_protected:Nn \__stex_expr_setup:nnnnnn {
5221   \group_begin:
5222   \tl_clear:N \l_stex_return_notation_tl
5223   \tl_set:Nn \l_stex_current_redo_tl {
5224     \let \this \stex_current_this:
5225     \def\comp{#1}
5226     \def\maincomp{\comp}
5227     \str_set:Nn \l_stex_current_symbol_str {#2}
5228     \str_set:Nn \l_stex_current_arity_str{ #3 }
5229     \tl_set:Nn \l_stex_current_args_tl{ #4 }
5230     \tl_set:Nn \l_stex_current_return_tl{ #5 }
5231     \tl_set:Nn \l_stex_current_type_tl{ #6 }
5232     \tl_clear:N \l_stex_current_term_tl
5233   }
5234   \tl_put_right:Nx \l_stex_current_redo_tl {
5235     \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5236   }
5237   \l_stex_current_redo_tl
5238 }

```

(End of definition for `\stex_invoke_symbol:nnnnnnN`. This function is documented on page ??.)

`\stex_invoke_variable:nnnnnn`

```
5239 \cs_new_protected:Nn \stex_invoke_variable:nnnnnnN {
5240   \bool_if:NTF \l_stex_allow_semantic_bool{
5241     \stex_if_html_backend:T{\ifvmode\indent\fi}
5242     \__stex_expr_setup:nnnnnn{\_varcomp}{#1}{#2}{#3}{#6}{#5}
5243     \cs_set_eq:NN \stex_term_oms_or_omv:nnn \stex_term_omv:nnn
5244     \tl_put_right:Nn \l_stex_current_redo_tl {
5245       \cs_set_eq:NN \stex_term_oms_or_omv:nnn \stex_term_omv:nnn
5246     }
5247     #7
5248   }{
5249     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
5250   }
5251 }
```

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

`\symuse`

```
5252 \cs_new_protected:Npn \symuse #1 {
5253   \stex_get_symbol:n{#1}
5254   \exp_args:Nno \use:n {\stex_invoke_symbol:ooxooooN
5255     \l_stex_get_symbol_mod_str
5256     \l_stex_get_symbol_name_str
5257     {int_use:N \l_stex_get_symbol_arity_int}
5258     \l_stex_get_symbol_args_tl
5259     \l_stex_get_symbol_def_tl
5260     \l_stex_get_symbol_type_tl
5261     \l_stex_get_symbol_return_tl}
5262   \l_stex_get_symbol_invoke_cs
5263 }
```

(End of definition for `\symuse`. This function is documented on page 89.)

`\stex_invoke_symbol:` Top-Level: Check whether text/math mode or custom notation, whether delimited by !, return code etc.

```
5264 \cs_new_protected:Nn \stex_invoke_symbol: {
5265   \stex_debug:nn{expressions}{Invoking~\l_stex_current_symbol_str}
5266   \mode_if_math:TF \__stex_expr_invoke_math: \__stex_expr_invoke_text:
5267 }
5268
5269 \cs_new_protected:Nn \__stex_expr_invoke_text: {
5270   \stex_debug:nn{expressions}{text~mode}
5271   \peek_charcode_remove:NTF ! \__stex_expr_invoke_op_custom:n \__stex_expr_invoke_custom:n
5272 }
5273
5274 \cs_new_protected:Nn \__stex_expr_invoke_math: {
5275   \stex_debug:nn{expressions}{math~mode}
5276   \peek_charcode_remove:NTF ! {
5277     % operator
5278     \peek_charcode_remove:NTF * \__stex_expr_invoke_op_custom:n {
5279       % op notation
5280       \peek_charcode:NTF [ \__stex_expr_invoke_op_notation:w {
5281         \__stex_expr_invoke_op_notation:w []
5282       }
5283     }
5284   }
```



```

5283   }
5284   }{
5285     \peek_charcode_remove:NTF * \__stex_expr_invoke_custom:n {
5286       % normal
5287       \peek_charcode:NTF [ \_stex_invoke_notation:w {
5288         \_stex_invoke_notation:w []
5289       }
5290     }
5291   }
5292 }

Notations:

5293 \cs_new_protected:Npn \_stex_invoke_notation:w [#1] {
5294   \stex_debug:nn{expressions}{using~notation~#1~for~\l_stex_current_symbol_str}
5295   \cs_if_exist:cTF{l_stex_notation_\l_stex_current_symbol_str_#1_cs}{
5296     \tl_if_empty:NTF \l_stex_current_return_tl {
5297       \stex_debug:nn{expressions}{return~empty}
5298       \use:c{l_stex_notation_\l_stex_current_symbol_str_#1_cs}{\group_end:\_stex_eat_exclam
5299     }{
5300       \stex_debug:nn{expressions}{return?}
5301       \exp_after:wN\exp_after:wN\exp_after:wN
5302       \__stex_expr_invoke_return_maybe:n
5303       \exp_after:wN\exp_after:wN\exp_after:wN
5304       {\cs:w l_stex_notation_\l_stex_current_symbol_str_#1_cs \cs_end: {}}
5305     }
5306   }{
5307     \stex_do_default_notation:
5308     \tl_if_empty:NTF \l_stex_current_return_tl {
5309       \l_stex_default_notation{\group_end:\_stex_eat_exclamation_point:}
5310     }{
5311       \exp_after:wN
5312       \__stex_expr_invoke_return_maybe:n
5313       \exp_after:wN
5314       {\l_stex_default_notation {}}
5315     }
5316   }
5317 }

5318
5319 \cs_new_protected:Npn \__stex_expr_invoke_op_notation:w [#1] {
5320   \stex_debug:nn{expressions}{op~notation~for~\l_stex_current_symbol_str}
5321   \cs_if_exist:cTF{l_stex_notation_\l_stex_current_symbol_str_op_#1_cs}{
5322     \_stex_maybe_brackets:nn{\neginfprec}{
5323       \_stex_term_oms_or_omv:nnn{#1}{}}
5324     {\use:c{l_stex_notation_\l_stex_current_symbol_str_op_#1_cs}}
5325   }
5326   \group_end:
5327 }{
5328   \int_compare:nNnTF \l_stex_current_arity_str = 0 {
5329     \tl_clear:N \l_stex_current_return_tl
5330     \_stex_invoke_notation:w [#1]
5331   }{
5332     \stex_do_default_notation_op:
5333     \_stex_maybe_brackets:nn{\neginfprec}{
5334       \_stex_term_oms_or_omv:nnn{#1}{}}

```

```

5335     {\l_stex_default_notation}
5336   }
5337   \group_end:
5338 }
5339 }
5340 }

Return:

5341 \cs_new_protected:Nn \__stex_expr_invoke_return_maybe:n {
5342   \tl_clear:N \l__stex_expr_return_args_tl
5343   \tl_set:Nn \l__stex_expr_return_this_tl {#1}
5344   \exp_args:Nnx \use:n {
5345     \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
5346     \cs_set:Npn \l_stex_current_arity_str } {
5347     \int_step_function:nN \l_stex_current_arity_str \__stex_expr_return_arg:n
5348     \__stex_expr_invoke_return_next:
5349   }
5350   \__stex_expr_ret_cs
5351 }
5352
5353 \cs_new:Nn \__stex_expr_return_arg:n {
5354   \tl_put_right:Nn \exp_not:N \l__stex_expr_return_args_tl {{#### #1}}
5355 }
5356
5357 \cs_new_protected:Nn \__stex_expr_invoke_return_next: {
5358   \peek_charcode_remove:NTF ! {
5359     \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl \group_end:
5360   }\__stex_expr_invoke_return:
5361 }
5362
5363 \cs_new_protected:Nn \__stex_expr_invoke_return: {
5364   \tl_set:Nx \l__stex_expr_return_this_tl {
5365     \__stex_expr_return_notation:n {
5366       \exp_after:wN \exp_after:wN \exp_after:wN
5367       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
5368         \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl
5369       }
5370     }
5371   }
5372   \stex_debug:nn{return}{Notation:~\meaning\l__stex_expr_return_this_tl}
5373   \tl_put_left:Nx \l__stex_expr_return_this_tl {
5374     \group_begin:\exp_args:No \exp_not:n \l_stex_current_redo_tl
5375   }
5376   \exp_args:Nnx \use:n {
5377     \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
5378     \cs_set:Npn \l_stex_current_arity_str } {
5379     \exp_args:No \exp_not:n \l_stex_current_return_tl
5380   }
5381   \stex_debug:nn{return}{
5382     \meaning\__stex_expr_ret_cs^^J
5383     \meaning\l__stex_expr_return_this_tl^^J
5384     \exp_args:No \exp_not:n \l__stex_expr_return_args_tl^^J
5385   }
5386   \exp_args:Nnx \use:nn {

```

```

5387 \exp_after:wN \group_end: \l__stex_expr_ret_cs
5388 }{
5389 \exp_args:No \exp_not:n \l__stex_expr_return_args_tl
5390 {
5391 \exp_args:No \exp_not:n \l__stex_expr_return_this_tl
5392 \group_end:
5393 }
5394 }
5395 }
5396
5397
5398 \cs_new_protected:Nn \l__stex_expr_return_notation:n {
5399 \tl_if_empty:NTF \l_stex_return_notation_tl { #1 }{
5400 \l_stex_return_notation_tl
5401 }
5402 }
5403

```

#### Custom Notations:

```

5404 \cs_new_protected:Nn \l__stex_expr_invoke_op_custom:n {
5405 \stex_debug:nn{expressions}{custom~op}
5406 \bool_set_true:N \l_stex_allow_semantic_bool
5407 \stex_term_oms_or_omv:nnn{}{}{\maincomp{#1}}
5408 \group_end:
5409 }
5410
5411 \int_new:N \l__stex_expr_arg_counter_int
5412 \cs_new_protected:Nn \l__stex_expr_invoke_custom:n {
5413 \stex_debug:nn{custom}{custom~notation~for~\l_stex_current_symbol_str}
5414 \stex_pseudogroup:nn{
5415 \bool_set_true:N \l_stex_allow_semantic_bool
5416 \prop_gclear:N \l__stex_expr_customs_prop
5417 \seq_gclear:N \l__stex_expr_customs_seq
5418 \int_gzero:N \l__stex_expr_arg_counter_int
5419 \tl_if_empty:NF \l_stex_current_args_tl {
5420 \exp_after:wN \l__stex_expr_add_prop_arg:nnw \l_stex_current_args_tl \_stex_args_end:
5421 \cs_set_eq:NN \arg \l__stex_expr_arg:n
5422 }
5423 \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
5424 \cs_set_eq:NN \l__stex_expr_do_ab_next:nnn \_stex_term_oma:nnn
5425 \stex_map_args:N \l__stex_expr_check_b:nn
5426 \l__stex_expr_do_ab_next:nnn{}{}{#1}
5427 }{
5428 \prop_if_exist:NT \l__stex_expr_customs_prop {
5429 \prop_gset_from_keyval:Nn \exp_not:N \l__stex_expr_customs_prop {
5430 \prop_to_keyval:N \l__stex_expr_customs_prop
5431 }
5432 }
5433 \int_gset:Nn \l__stex_expr_arg_counter_int { \int_use:N \l__stex_expr_arg_counter_int}
5434 \seq_if_exist:NT \l__stex_expr_customs_seq {
5435 \seq_gset_split:Nnn \exp_not:N \l__stex_expr_customs_seq , {
5436 \seq_use:Nn \l__stex_expr_customs_seq ,
5437 }
5438 }

```

```

5439 }
5440 % TODO check that all arguments are present
5441 \group_end:
5442 }
5443
5444 \cs_new_protected:Npn \__stex_expr_add_prop_arg:nnw #1 #2 #3\__stex_args_end: {
5445   \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {}
5446   \seq_gput_right:Nn \l__stex_expr_customs_seq {#2}
5447   \tl_if_empty:nF{#3}{\__stex_expr_add_prop_arg:nnw #3 \__stex_args_end:}
5448 }
5449
5450 \cs_new:Nn \__stex_expr_check_b:nn {
5451   \str_case:nn #2 {
5452     b {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \__stex_term_omb:nnn}
5453     B {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \__stex_term_omb:nnn}
5454   }
5455 }
5456
5457 \NewDocumentCommand \__stex_expr_arg:n {s O{} m} {
5458   \IfBooleanTF #1 {
5459     \stex_annotate_invisible:n{
5460       \__stex_expr_arg_inner:nn{#2}{#3}
5461     }
5462   }{
5463     \__stex_expr_arg_inner:nn{#2}{#3}
5464   }
5465 }
5466
5467 \cs_new_protected:Nn \__stex_expr_arg_inner:nn {
5468   \tl_if_empty:nTF{#1}{
5469     \int_gincr:N \l__stex_expr_arg_counter_int
5470     \exp_args:Ne \__stex_expr_check:nTF{ \int_use:N \l__stex_expr_arg_counter_int }{
5471       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl
5472     }{
5473       \__stex_expr_arg_inner:nn{}
5474     }{ #2 }
5475   }{
5476     \__stex_expr_check:nTF {#1}{
5477       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5478     }{
5479       \exp_args:No \str_case:nnTF \l_tmpb_tl {
5480         {a}{
5481           \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
5482             \l_tmpa_tl X
5483           }
5484           \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5485         }
5486         {B}{
5487           \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
5488             \l_tmpa_tl X
5489           }
5490           \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5491         }
5492       }{

```

```

5493     \_stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5494   }{
5495     \msg_error:nxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5496   }
5497 }
5498 }
5499 }
5500
5501 \prg_new_conditional:Nnn \_stex_expr_check:n {TF} {
5502   \exp_args:NNe \prop_get:NnNTF \l_stex_expr_customs_prop {#1} \l_tmpa_tl {
5503     \tl_set:Nx \l_tmpb_tl {\seq_item:Nn \l_stex_expr_customs_seq {#1}} }
5504   \tl_if_empty:NTF \l_tmpa_tl {
5505     \exp_args:NNe \prop_gput:Nnn \l_stex_expr_customs_prop
5506       { #1 }{X}
5507     \exp_args:No \str_case:nnF \l_tmpb_tl {
5508       {a}{
5509         \tl_set:Nx \l_tmpa_tl{ #1 1 }
5510       }
5511       {B}{
5512         \tl_set:Nx \l_tmpa_tl{ #1 1 }
5513       }
5514     }{
5515       \tl_set:Nx \l_tmpa_tl{ #1 }
5516     }
5517     \prg_return_true:
5518   }{
5519     \prg_return_false:
5520   }
5521 }{
5522   \msg_error:nxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5523   \prg_return_false:
5524 }
5525 }
5526
5527 % #1 argnum #2 argmode #3 code
5528 \cs_new_protected:Nn \_stex_expr_arg_do:nnn {
5529   \stex_debug:nn{custom}{Doing~argument~#1~of~mode~#2:~\tl_to_str:n{#3}}
5530   \group_begin:
5531     \bool_set_true:N \l_stex_allow_semantic_bool
5532     \_stex_term_arg:nnn {#2}{#1}{#3}
5533   \group_end:
5534 }
5535 \cs_generate_variant:Nn \_stex_expr_arg_do:nnn {oon}

```

(End of definition for `\stex_invoke_symbol:`. This function is documented on page 131.)

## Argument Handling and Annotating

```

\_stex_term_arg:nnnnn
\_stex_term_arg:nnn
5536 % 1: argnum 2: argmode 3: precedence 4: argname 5: code
5537 \cs_new_protected:Nn \_stex_term_arg:nnnnn {
5538   \group_begin:
5539     \str_clear:N \l_stex_current_symbol_str
5540     \tl_clear:N \l_stex_current_term_tl

```

```

5541 \int_set:Nn \l_stex_notation_downprec { #3 }
5542 \bool_set_true:N \l_stex_allow_semantic_bool
5543 \stex_term_arg:nnn {#2}{#1}{
5544   \tl_if_empty:nTF{#4}{
5545     #5
5546   }{
5547     \stex_annotate:nn{mml:arg={#4}}{#5}
5548   }
5549 }
5550 \group_end:
5551 }
5552
5553 \cs_new_protected:Nn \stex_term_arg:nnn {
5554   \stex_annotate:nn{ shtml:arg={#2}, shtml:argmode={#1}}{
5555     \stex_annotate_force_break:n{ #3 }
5556   }
5557 }

```

(End of definition for `\stex_term_arg:nnnnn` and `\stex_term_arg:nnn`. These functions are documented on page ??.)

`\stex_term_arg_aB:nnnnn`

```

5558 \tl_set:Nn \__stex_expr_do_aB_clist: {
5559   \seq_use:Nn \l_stex_aB_args_seq {
5560     \mathpunct{\comp{,}}
5561   }
5562 }
5563 \tl_set_eq:NN \stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5564
5565 \int_new:N \l__stex_expr_count_int
5566 \cs_new_protected:Nn \stex_term_arg_aB:nnnnn {
5567   \tl_if_empty:nTF{#5}{
5568     \stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{
5569   }{
5570     \seq_clear:N \l_stex_aB_args_seq
5571     \int_zero:N \l__stex_expr_count_int
5572     \clist_map_inline:nn{#5}{
5573       \__stex_expr_aB_arg:nnnnn{##1}{#1}{#2}{#3}{#4}
5574     }
5575     \stex_term_do_aB_clist:
5576   }
5577 }
5578
5579 % 1: code 2: argnum 3: argmode 4: precedence 5: argname
5580 \cs_new_protected:Npn \__stex_expr_aB_arg:nnnnn #1 {
5581   \int_incr:N \l__stex_expr_count_int
5582   \__stex_expr_is_varseq:nTF{#1}{
5583     \exp_after:wN \exp_after:wN \exp_after:wN
5584     \__stex_expr_assoc_seq:nnnnnnn
5585     \exp_after:wN
5586     \__stex_expr_gobble:nnnnnnn #1 \__stex_expr_end:
5587   }{
5588     \__stex_expr_is_seqmap:nTF{#1}{
5589       \exp_args:NNe \use:nn \__stex_expr_do_seqmap:nnnnnn {\tl_tail:n{#1}}

```

```

5590     }{
5591     \__stex_expr_aB_simple_arg:nnnnn{#1}
5592     }
5593   }
5594 }
5595
5596 \cs_new:Npn \__stex_expr_gobble:nnnnnnn #1 #2 #3 #4 #5 #6 #7 #8 #9 \__stex_expr_end: {
5597   {#2} #3 {#6}
5598 }
5599
5600 \cs_new_protected:Nn \__stex_expr_aB_simple_arg:nnnnn{
5601   \seq_put_right:Nx \l_stex_aB_args_seq {
5602     \stex_term_arg:nnnnn{#2}\int_use:N\l__stex_expr_count_int}{#3}{#4}{#5}{
5603     \exp_not:n{
5604       \tl_set_eq:NN \stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5605       #1
5606     }
5607   }
5608 }
5609 }
5610
5611 \cs_new_protected:Nn \stex_is_sequentialized:n {
5612   \group_begin: #1 \group_end:
5613 }

```

Conditionals: Is the argument a sequence variable or a \seqmap?

```

5614 \prg_new_conditional:Nnn \__stex_expr_is_varseq:n {TF} {
5615   \int_compare:nNnTF {\tl_count:n{#1}} = 1 {
5616     \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_head:n{#1}}
5617     \stex_invoke_variable:nnnnnN {
5618       \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No\tl_item:nn{#1}{8}}
5619       \stex_invoke_sequence:
5620       \prg_return_true:\prg_return_false:
5621     }\prg_return_false:
5622   }\prg_return_false:
5623 }
5624
5625 \prg_new_conditional:Nnn \__stex_expr_is_seqmap:n {TF} {
5626   \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
5627     \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\seqmap}
5628     \prg_return_true:\prg_return_false:
5629   }\prg_return_false:
5630 }

```

Sequence variable:

```

5631 % 1: name 2: arity 3: clist 4: argnum 5: argmode 6: precedence 7: argname
5632 \cs_new_protected:Nn \__stex_expr_assoc_seq:nnnnnnn {
5633   \group_begin:
5634     \seq_clear:N \l_stex_aB_args_seq
5635     \__stex_expr_assoc_make_seq:nnn{#1}{#3}{#2}
5636   \exp_args:NNe \use:nn \group_end: {
5637     \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5638       \stex_is_sequentialized:n{
5639         \stex_term_arg:nnnnn{#4}\int_use:N\l__stex_expr_count_int}{#5}{#6}{#7}{

```

```

5640         \bool_set_true:N \l_stex_allow_semantic_bool
5641         \str_set:Nn \exp_not:N \l_stex_current_symbol_str
5642             {\l_stex_current_symbol_str}
5643         \tl_if_empty:NF \l_stex_current_term_tl {
5644             \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5645                 \exp_args:No \exp_not:n \l_stex_current_term_tl
5646             }
5647         }
5648         \stex_annotate:nn{
5649             shtml:term=OMV,
5650             shtml:head={#1},
5651             shtml:notationid={}}
5652         ){
5653             \_stex_annotate_force_break:n{
5654                 \_stex_term_do_aB_clist:
5655             }
5656         }
5657     }
5658 }
5659 }
5660 }
5661 }
5662
5663 % #1: name, #2: clist, #3:arity
5664 \cs_new_protected:Nn \__stex_expr_assoc_make_seq:nnn {
5665     \cs_if_exist:cTF{l_stex_notation_#1_cs}{
5666         \cs_set_eq:Nc \l__stex_expr_cs {l_stex_notation_#1_cs}
5667     }{
5668         \stex_do_default_notation:
5669         \cs_set_eq:NN \l__stex_expr_cs \l_stex_default_notation
5670     }
5671     \clist_map_inline:nn{#2}{
5672         \tl_if_eq:nnTF{##1}{\ellipses}{
5673             \seq_put_right:Nn \l_stex_aB_args_seq { ##1 }
5674         }{
5675             \int_compare:nNnTF {#3} = 1 {
5676                 \tl_set:Nn \l__stex_expr_iarg_tl { ##1 }
5677             }{
5678                 \tl_set:Nn \l__stex_expr_iarg_tl { ##1 }
5679             }
5680             \seq_put_right:Nx \l_stex_aB_args_seq {
5681                 \group_begin:
5682                 \exp_not:n {
5683                     \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5684                     \def\comp{\_varcomp}
5685                     \str_set:Nn \l_stex_current_symbol_str{#1}
5686                 }
5687                 \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
5688                 \exp_after:wN \exp_after:wN \exp_after:wN {
5689                     \exp_after:wN \l__stex_expr_cs \exp_after:wN \group_end: \l__stex_expr_iarg_tl
5690                 }
5691             }
5692         }
5693     }

```



```

5694 }
      \seqmap:
5695 % 1: fun 2: clist 3: argnum 4: argmode 5: precedence 6: argname
5696 \cs_new_protected:Nn \__stex_expr_do_seqmap:nnnnnn {
5697   \group_begin:
5698     \cs_set:Npn \l_tmpa_cs ##1 {#1}
5699     \seq_clear:N \l_stex_aB_args_seq
5700     \clist_map_inline:nn{#2}{
5701       \__stex_expr_is_varseq:nTF{##1}{
5702         \exp_after:wN
5703         \__stex_expr_varseq_in_map:nnnnnnn ##1
5704       }{
5705         \seq_put_right:Nn \l_stex_aB_args_seq {##1}
5706       }
5707     }
5708     \seq_clear:N \l__stex_expr_old_seq
5709     \seq_map_inline:Nn \l_stex_aB_args_seq {
5710       \tl_if_eq:nnTF{##1}{\ellipses}{
5711         \seq_put_right:Nn \l__stex_expr_old_seq {##1}
5712       }
5713       % TODO \stex_is_sequentialized:n
5714       {
5715         \exp_args:NNo \seq_put_right:Nn \l__stex_expr_old_seq {
5716           \l_tmpa_cs {##1}
5717         }
5718       }
5719     }
5720     \seq_set_eq:NN \l_stex_aB_args_seq \l__stex_expr_old_seq
5721     \exp_args:NNe \use:nn \group_end: {
5722       \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5723         \stex_is_sequentialized:n{
5724           \stex_term_arg:nnnnn{#4\int_use:N\l__stex_expr_count_int}{#4}{#5}{#6}{
5725             \bool_set_true:N \l_stex_allow_semantic_bool
5726             \str_set:Nn \exp_not:N \l_stex_current_symbol_str
5727               {\l_stex_current_symbol_str}
5728             \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5729               \symuse{Metatheory?sequence~map}
5730               {\exp_args:No \exp_not:n \l_tmpa_tl}
5731               {\stex_term_do_aB_clist:}
5732             }
5733             \__stex_expr_do_headterm:nn{}{
5734               \stex_term_do_aB_clist:
5735             }
5736           }
5737         }
5738       }
5739     }
5740 }
5741
5742 \cs_new_protected:Nn \__stex_expr_varseq_in_map:nnnnnnn {
5743   \__stex_expr_assoc_make_seq:nnn{#2}{#6}{#3}
5744 }

```

(End of definition for `\stex_term_arg_aB:nnnnn`. This function is documented on page ??.)

## Term HTML Annotations

```

\stex_term_oms_or_omv:nnn
  \stex_term_oms:nnn
  \stex_term_omv:nnn
5745 \cs_new_protected:Nn \stex_eat_exclamation_point: {
5746   \peek_charcode_remove:NT ! {
5747     \stex_eat_exclamation_point:
5748   }
5749 }
5750
5751 \bool_new:N \stex_in_invisible_html_bool
5752 \bool_set_false:N \stex_in_invisible_html_bool
5753 \stex_if_html_backend:TF {
5754   % 1: variant 2: intent 3: code
5755   \cs_new_protected:Nn \stex_term_oms:nnn {
5756     \tl_if_empty:NTF \l_stex_current_term_tl {
5757       \stex_annotate:nn{
5758         shtml:term=OMID,
5759         shtml:head={\l_stex_current_symbol_str},
5760         shtml:notationid={#1},
5761       }{
5762         \stex_annotate_force_break:n{#3}
5763       }
5764     }{
5765       \__stex_expr_do_headterm:nn{#1}{#3}
5766     }
5767   }
5768   \cs_new_protected:Nn \stex_term_omv:nnn {
5769     \tl_if_empty:NTF \l_stex_current_term_tl {
5770       \stex_annotate:nn{
5771         shtml:term=OMV,
5772         shtml:head={\l_stex_current_symbol_str},
5773         shtml:notationid={#1}
5774       }{
5775         \stex_annotate_force_break:n{#3}
5776       }
5777     }{
5778       \__stex_expr_do_headterm:nn{#1}{#3}
5779     }
5780   }
5781   \cs_new_protected:Nn \__stex_expr_do_headterm:nn {
5782     \bool_if:NTF \stex_in_invisible_html_bool {
5783       {\bool_set_true:N \l_stex_allow_semantic_bool
5784         \ensuremath{\l_stex_current_term_tl}
5785       }
5786     }{
5787       \stex_annotate:nn{
5788         shtml:term=complex,
5789         shtml:head={\l_stex_current_symbol_str},
5790         shtml:notationid={#1}
5791       }{
5792         \stex_annotate_force_break:n{
5793           \stex_annotate_invisible:nn{shtml:headterm={}}{
5794             {\bool_set_true:N \l_stex_allow_semantic_bool
5795               \ensuremath{\l_stex_current_term_tl}

```

```

5796         }
5797     }
5798 }
5799 #2
5800 }
5801 }
5802 }
5803 }{
5804 \cs_new_protected:Nn \_stex_term_oms:nnn {#3}
5805 \cs_new_protected:Nn \_stex_term_omv:nnn {#3}
5806 \cs_new_protected:Nn \_stex_expr_do_headterm:nn { #2 }
5807 }
5808 \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn

```

(End of definition for `\_stex_term_oms_or_omv:nnn`, `\_stex_term_oms:nnn`, and `\_stex_term_omv:nnn`.  
These functions are documented on page ??.)

`\_stex_term_oma:nnn`

```

5809 \stex_if_html_backend:TF {
5810 \cs_new_protected:Nn \_stex_term_oma:nnn {
5811 \tl_if_empty:NTF \l_stex_current_term_tl {
5812 \stex_annotate:nn{
5813 shtml:term=OMA,
5814 shtml:head={\l_stex_current_symbol_str},
5815 shtml:notationid={#1}
5816 }{
5817 \_stex_annotate_force_break:n{#3}
5818 }
5819 }{
5820 \stex_annotate:nn{
5821 shtml:term=OMA,
5822 shtml:head={\l_stex_current_symbol_str},
5823 shtml:notationid={#1}
5824 }{
5825 \_stex_annotate_force_break:n{
5826 \stex_annotate_invisible:nn{shtml:headterm={}}{
5827 \bool_set_true:N \l_stex_allow_semantic_bool
5828 \l_stex_current_term_tl
5829 }}
5830 #3
5831 }
5832 }
5833 }
5834 }
5835 }{
5836 \cs_new_protected:Nn \_stex_term_oma:nnn {#3}
5837 }

```

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

`\_stex_term_omb:nnn`

```

5838 \stex_if_html_backend:TF {
5839 \cs_new_protected:Nn \_stex_term_omb:nnn {
5840 \tl_if_empty:NTF \l_stex_current_term_tl {
5841 \stex_annotate:nn{

```

```

5842     shtml:term=OMBIND,
5843     shtml:head={\l_stex_current_symbol_str},
5844     shtml:notationid={#1}
5845   }{
5846     \stex_annotate_force_break:n{#3}
5847   }
5848 }{
5849   \stex_annotate:nn{
5850     shtml:term=OMBIND,
5851     shtml:head={\l_stex_current_symbol_str},
5852     shtml:notationid={#1}
5853   }{
5854     \stex_annotate_force_break:n{
5855       \stex_annotate_invisible:nn{shtml:headterm={}}{
5856         \bool_set_true:N \l_stex_allow_semantic_bool
5857         \l_stex_current_term_tl
5858       }}
5859     #3
5860   }
5861 }
5862 }
5863 }
5864 }{
5865   \cs_new_protected:Nn \stex_term_omb:nnn { #3 }
5866 }

```

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

## Automated Bracketing

`\infprec`  
`\neginfprec`

```

5867 \tl_const:Nx \infprec {\int_use:N \c_max_int}
5868 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}

```

(End of definition for `\infprec` and `\neginfprec`. These functions are documented on page 91.)

`\dobrackets`

```

5869 \int_new:N \l_stex_notation_downprec
5870 \int_set:Nn \l_stex_notation_downprec \infprec
5871 \tl_set:Nn \l__stex_expr_left_bracket_str (
5872 \tl_set:Nn \l__stex_expr_right_bracket_str )
5873 \bool_new:N \l_stex_brackets_dones_bool
5874
5875 \cs_new_protected:Nn \stex_maybe_brackets:nn {
5876   \bool_if:NTF \l_stex_brackets_dones_bool {
5877     \bool_set_false:N \l_stex_brackets_dones_bool
5878     #2
5879   } {
5880     \stex_debug:nn{brackets}{#1}>\int_eval:n \l_stex_notation_downprec?}
5881     \int_compare:nNnTF { #1 } > \l_stex_notation_downprec {
5882       %\bool_if:NTF \l_stex_inparray_bool { #2 }{
5883         \dobrackets {
5884           #2
5885         }

```

```

5886     %}
5887   }{
5888     #2
5889   }
5890 }
5891 }
5892
5893 %\RequirePackage{scalereel}
5894 \cs_new_protected:Npn \dobrackets #1 {
5895   %\ThisStyle{\if D@m@switch
5896   %   \exp_args:Nnx \use:nn
5897   %   { \exp_after:wN \left\l_1__stex_expr_left_bracket_str #1 }
5898   %   { \exp_not:N\right\l_1__stex_expr_right_bracket_str }
5899   %   \else
5900   \group_begin:
5901   %\stex_pseudogroup_with:nn{\l_1__stex_brackets_dones_bool\l_1__stex_notation_downprec}{
5902     \bool_set_true:N \l_1__stex_brackets_dones_bool
5903     %\int_set:Nn \l_1__stex_notation_downprec \infpref
5904     \mathopen{\cs_if_exist:NT\l_1__stex_current_symbol_str\comp
5905     \l_1__stex_expr_left_bracket_str
5906     }
5907     #1
5908     \group_end:%}
5909     \mathclose{\cs_if_exist:NT\l_1__stex_current_symbol_str\comp
5910     \l_1__stex_expr_right_bracket_str
5911     }
5912     %\fi}
5913 }

```

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

### \withbrackets

```

5914 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
5915   \stex_pseudogroup_with:nn{\l_1__stex_expr_left_bracket_str\l_1__stex_expr_right_bracket_str}{
5916     \tl_set:Nn \l_1__stex_expr_left_bracket_str { #1 }
5917     \tl_set:Nn \l_1__stex_expr_right_bracket_str { #2 }
5918     #3
5919   }
5920 }

```

(End of definition for \withbrackets. This function is documented on page 92.)

### \dowithbrackets

```

5921 \cs_new_protected:Npn \dowithbrackets #1 #2 #3 {
5922   \withbrackets{#1}{#2}{\dobrackets{#3}}
5923 }

```

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

## Symname and Variants

```

\symname
  \sn 5924 \def\maincomp{\comp}
  \sns 5925
\Symname 5926 \stex_keys_define:nnnn{symname}{
  \Sn
  \Sns
\symref
  \sr
\varref
\varname
\Varname

```

```

5927 \tl_clear:N \l_stex_key_pre_tl
5928 \tl_clear:N \l_stex_key_post_tl
5929 %\tl_clear:N \l_stex_key_proot_tl
5930 }{
5931   pre .tl_set:N = \l_stex_key_pre_tl ,
5932   post .tl_set:N = \l_stex_key_post_tl ,
5933   root .code:n = {}% .tl_set:N = \l_stex_key_root_tl
5934 }{}
5935
5936 \NewDocumentCommand \symref { 0{} m m } {
5937   \group_begin:
5938   \stex_keys_set:nn{symname}{#1}
5939   \stex_get_symbol:n{#2}
5940   \__stex_expr_do_ref:nNn{#3}\symrefemph@uri\stex_term_oms:nnn
5941 }
5942 \let\sr\symref
5943
5944 \NewDocumentCommand \symname { 0{} m } {
5945   \group_begin:
5946   \stex_keys_set:nn{symname}{#1}
5947   \stex_get_symbol:n{#2}
5948   \__stex_expr_do_ref:nNn{
5949     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
5950   }\symrefemph@uri\stex_term_oms:nnn
5951 }
5952 \let\sn\symname
5953 \protected\def\sns{\symname[post=s]}
5954
5955 \NewDocumentCommand \Symname { 0{} m } {
5956   \group_begin:
5957   \stex_keys_set:nn{symname}{#1}
5958   \stex_get_symbol:n{#2}
5959   \__stex_expr_do_ref:nNn{
5960     \l_stex_key_pre_tl\exp_after:wN\stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
5961   }\symrefemph@uri\stex_term_oms:nnn
5962 }
5963 \cs_new_protected:Nn \stex_capitalize:n {
5964   \uppercase{#1}
5965 }
5966 \let\Sn\Symname
5967 \protected\def\Sns{\Symname[post=s]}
5968
5969 \cs_new:Npn \stex_split_slash: #1/#2/#3\stex_args_end: {
5970   \tl_if_empty:nTF{#3}{
5971     #2
5972   }{
5973     \stex_split_slash: #2 / #3 \stex_args_end:
5974   }
5975 }
5976
5977 \NewDocumentCommand \varref { 0{} m m } {
5978   \group_begin:
5979   \stex_keys_set:nn{symname}{#1}
5980   \stex_get_var:n{#2}

```

```

5981 \_stex_expr_do_ref:nNn{#3}\varemp@uri{
5982 \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5983 \def\comp{\_varcomp}
5984 \_stex_term_omv:nnn
5985 }
5986 }
5987
5988 \NewDocumentCommand \varname { 0{ } m } {
5989 \group_begin:
5990 \stex_keys_set:nn{symname}{#1}
5991 \stex_get_var:n{#2}
5992 \_stex_expr_do_ref:nNn{
5993 \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
5994 }\varemp@uri{
5995 \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5996 \def\comp{\_varcomp}
5997 \_stex_term_omv:nnn
5998 }
5999 }
6000
6001 \NewDocumentCommand \Varname { 0{ } m } {
6002 \group_begin:
6003 \stex_keys_set:nn{symname}{#1}
6004 \stex_get_var:n{#2}
6005 \_stex_expr_do_ref:nNn{
6006 \l_stex_key_pre_tl\exp_after:wN\stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
6007 }\varemp@uri{
6008 \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
6009 \def\comp{\_varcomp}
6010 \_stex_term_omv:nnn
6011 }
6012 }
6013
6014
6015 \cs_new_protected:Nn \_stex_expr_do_ref:nNn {
6016 \stex_if_html_backend:T{\ifvmode\indent\fi}
6017 \bool_if:NTF \l_stex_allow_semantic_bool{
6018 \str_set:Nx\l_stex_current_symbol_str
6019 {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6020 \str_if_in:NnT \l_stex_get_symbol_name_str / {
6021 \str_set:Nx \l_stex_get_symbol_name_str {
6022 \exp_after:wN \stex_split_slash: \l_stex_get_symbol_name_str
6023 /\_stex_args_end:
6024 }
6025 }
6026 \tl_clear:N \l_stex_current_term_tl
6027 \def\comp{\_comp}
6028 \let\compemph@uri#2
6029 #3{}{\_comp{#1}}
6030 }{
6031 \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
6032 }
6033 \group_end:
6034 }

```

(End of definition for `\symname` and others. These functions are documented on page 89.)

## Highlighting

```
6035 <@@=stex_notationcomps>
      \comp
\compemph@uri 6036 \cs_new_protected:Nn \_do_comp:nNn {
      \compemph 6037 \stex_pseudogroup_with:nn{\comp}{
      \defemph 6038 \def\comp##1{##1}
\defemph@uri 6039 \str_if_empty:NTF \l_stex_current_symbol_str {
      \symrefemph 6040 #3
\symrefemph@uri 6041 }{
      \varemp 6042 \stex_if_html_backend:TF {
\var emph@uri 6043 \stex_annotate:nn { shtml:#1 = \l_stex_current_symbol_str}{ #3 }
      \varemp 6044 }{
      \varemp 6045 \exp_args:Nno #2 { #3 } \l_stex_current_symbol_str
      \varemp 6046 }
      \varemp 6047 }
      \varemp 6048 }
      \varemp 6049 }
6050
6051 \cs_new_protected:Npn \_comp {
6052 \_do_comp:nNn {comp}\compemph@uri
6053 }
6054
6055 \cs_new_protected:Npn \_varcomp {
6056 \_do_comp:nNn {varcomp}\varemp@uri
6057 }
6058
6059 \cs_new_protected:Npn \_defcomp {
6060 \_do_comp:nNn {definiendum}\defemph@uri
6061 }
6062
6063 \cs_set_protected:Npn \comp {}
6064
6065 \cs_new_protected:Npn \compemph@uri #1 #2 {
6066 \compemph{ #1 }
6067 }
6068
6069 \cs_new_protected:Npn \compemph #1 {
6070 #1
6071 }
6072
6073 \cs_new_protected:Npn \defemph@uri #1 #2 {
6074 \defemph{#1}
6075 }
6076
6077 \cs_new_protected:Npn \defemph #1 {
6078 \ifmode\else\expandafter\textbf\fi{#1}
6079 }
6080
6081 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
6082 \symrefemph{#1}
```



```

6083 }
6084
6085 \cs_new_protected:Npn \symrefemph #1 {
6086   \emph{#1}
6087 }
6088
6089 \cs_new_protected:Npn \varemp@uri #1 #2 {
6090   \varemp{#1}
6091 }
6092
6093 \cs_new_protected:Npn \varemp #1 {
6094   #1
6095 }

```

(End of definition for `\comp` and others. These functions are documented on page 90.)

## 13.9 Mathematical Structures

```

6096 <@@=stex_structures>
\this
6097 \cs_new_protected:Npn \stex_current_this: {
6098   { \bool_set_true:N \l_stex_allow_semantic_bool
6099     \tl_if_empty:NTF \l_stex_current_this_tl {} }{
6100     \str_set:Nx \l_stex_current_symbol_str {
6101       \l_stex_current_module_str ? \l__stex_structures_name_str
6102     }
6103     \maincomp{\l_stex_current_this_tl}
6104   }
6105 }
6106 }
6107 \let \this \stex_current_this:

```

(End of definition for `\this`. This function is documented on page 96.)

`mathstructure` (*env.*)

```

6108 \stex_new_stylable_env:nnnnnnn {mathstructure}{m 0}{ }{
6109   \__stex_structures_begin:nn{#1}{#2}
6110   \stex_smsmode_do:
6111 }{
6112   \stex_structural_feature_module_end:
6113   \__stex_structures_do externals:
6114 }{ }{ }{ }
6115
6116 \stex_keys_define:nnnn{mathstructure}{
6117   \tl_clear:N \l_stex_current_this_tl
6118   \str_clear:N \l__stex_structures_name_str
6119 }{
6120   this .tl_set:N = \l_stex_current_this_tl ,
6121   unknown .code:n = {
6122     \str_if_empty:NTF \l_keys_key_str {
6123       \str_set:Nx \l__stex_structures_name_str {\l_keys_key_tl}
6124     }{
6125       \str_set_eq:NN \l__stex_structures_name_str \l_keys_key_str

```

```

6126     }
6127   }
6128 }{}
6129
6130 \cs_new_protected:Nn \__stex_structures_begin:nn {
6131   \stex_keys_set:nn {mathstructure}{#2}
6132   \str_if_empty:NT \l__stex_structures_name_str {
6133     \str_set:Nn \l__stex_structures_name_str {#1}
6134   }
6135   \def\comp{\_comp}
6136
6137   \exp_args:Nne \use:nn { \stex_module_add_symbol:nnnnnnN }
6138     { #1}{\l__stex_structures_name_str}{0}{}{defed}{
6139     \l_stex_current_module_str / \l__stex_structures_name_str-module
6140   }
6141   {} \stex_invoke_structure:
6142   \str_set:Nx \l_stex_macroname_str {#1}
6143   \stex_execute_in_module:x{
6144     \seq_clear:c{l_stex_structure_macros_\l_stex_current_module_str / \l__stex_structures_name_str}
6145     \seq_put_right:cn{l_stex_structure_macros_\l_stex_current_module_str / \l__stex_structures_name_str}{#1}
6146   }
6147   \exp_args:No \stex_structural_feature_module:nn
6148     {\l__stex_structures_name_str}{structure}
6149 }
6150
6151 \stex_sms_allow_env:n{mathstructure}
6152 \stex_deactivate_macro:Nn \mathstructure {module~environments}
6153 \stex_every_module:n {\stex_reactivate_macro:N \mathstructure}
6154
6155 \cs_new_protected:Nn \__stex_structures_do externals: {
6156   \tl_set:Nn \l__stex_structures_replace_this_tl {####1}
6157   \exp_args:No \stex_iterate_symbols:nn{\g_stex_last_feature_str}{
6158     \__stex_structures_external_decl:nnnn{##5}{##4}{##3}{##8}
6159   }
6160 }
6161
6162 \cs_new_protected:Nn \__stex_structures_external_decl:nnnn {
6163   %\stex_debug:nn{structure}{
6164   % Generating~external~declaration~\l__stex_structures_name_str/#3~in~
6165   % \l_stex_current_module_str^^J
6166   % \tl_to_str:n{#1}^^J\tl_to_str:n{#2}^^J\tl_to_str:n{#4}
6167   %}
6168   %\tl_set:Nn \l_stex_get_symbol_args_tl {#1}
6169   %\exp_args:Nnx \use:nn { \stex_module_add_symbol:nnnnnnN } {
6170   % {#1}{\l__stex_structures_name_str/#3}{\int_eval:n{#2 + 1}}
6171   % {1i\tl_if_empty:nF{#1}{\stex_map_args:N \__stex_structures_shift_args:nn}}
6172   % {defed}{typed}
6173   %}{#4}\stex_invoke_outer_field:
6174 }
6175
6176 \cs_new:Nn \__stex_structures_shift_args:nn {
6177   \int_eval:n{#1+1}#2
6178 }

```

`\stex_get_mathstructure:n`

```
6179 \cs_new_protected:Nn \stex_get_mathstructure:n {
6180   \_stex_get_mathstructure:n{#1}
6181   \str_if_empty:NT \l_stex_get_structure_module_str {
6182     \msg_error:nnn{stex}{error/unknownstructure}{#1}
6183   }
6184 }
6185 \cs_new_protected:Nn \_stex_get_mathstructure:n {
6186   \str_clear:N \l_stex_get_structure_module_str
6187   \_stex_get_symbol:n{#1}
6188   \str_if_empty:NF \l_stex_get_symbol_name_str {
6189     \exp_args:No \tl_if_eq:NNT \l_stex_get_symbol_invoke_cs \stex_invoke_structure: {
6190       \str_set_eq:NN \l_stex_get_structure_module_str \l_stex_get_symbol_type_tl
6191     }
6192   }
6193 }
```

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

`extstructure (env.)`

```
6194 \stex_new_stylable_env:nnnnnnn {extstructure}{m 0{} m}{
6195   \seq_clear:N \l__stex_structures_imports_seq
6196   \clist_map_inline:nn{#3}{
6197     \stex_get_mathstructure:n{##1}
6198     \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6199       \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{####1}}{
6200         \seq_put_right:Nn \l__stex_structures_imports_seq{####1}
6201       }
6202     }
6203     \stex_execute_in_module:x{
6204       \seq_put_right:cn{l_stex_structure_macros_\l_stex_get_structure_module_str _seq}{#1}
6205     }
6206   }
6207   \__stex_structures_begin:nn{#1}{#2}
6208   \seq_map_inline:Nn \l__stex_structures_imports_seq{
6209     \stex_if_do_html:T {
6210       \hbox{\stex_annotate_invisible:nn
6211         {shtml:import={##1}} {}}
6212     }
6213     \stex_module_add_morphism:nonn
6214     {#1}{import}{}
6215     \stex_execute_in_module:x{
6216       \stex_activate_module:n{##1}
6217     }
6218   }
6219   \stex_smsmode_do:
6220 }{
6221   \stex_structural_feature_module_end:
6222   \__stex_structures_do externals:
6223 }{}{}{}
6224
6225 \stex_sms_allow_env:n{extstructure}
6226 \stex_deactivate_macro:Nn \extstructure {module-environments}
6227 \stex_every_module:n {
```

```

6228 \stex_reactivate_macro:N \extstructure
6229 }
6230
6231 \cs_new:Nn \__stex_structures_extend_structure_i:NnnnnnnN {
6232   \exp_not:n{#1{#2}{#3}{#4}{#5}{defed}}{\l__stex_structures_extmod_str,#7}\exp_not:n{#8}{#9}
6233 }
6234 \cs_new_protected:Nn \__stex_structures_extend_structure:nn {
6235   \stex_debug:nn{ext}{Extending~#1~by~#2}
6236   \str_set:Nn \l__stex_structures_extmod_str{#2}
6237   \tl_set:cx{#1}{
6238     \exp_after:wN \exp_after:wN \exp_after:wN
6239     \__stex_structures_extend_structure_i:NnnnnnnN \cs:w #1 \cs_end:
6240   }
6241 }
6242
6243 \stex_new_stylable_env:nnnnnnn {extstructure*}{m}{
6244   \__stex_structures_new_extstruct_name:
6245   \seq_clear:N \l__stex_structures_imports_seq
6246   \stex_get_mathstructure:n{#1}
6247
6248   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6249     \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6250       \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6251     }
6252   }
6253
6254   \stex_execute_in_module:x{
6255     \seq_map_inline:cn{l_stex_structure_macros_\l_stex_get_structure_module_str _seq}{
6256       \exp_not:N \tl_if_exist:cT{####1}{
6257         \__stex_structures_extend_structure:nn{####1}{\l_stex_current_module_str/\l__stex_st
6258       }
6259     }
6260   }
6261
6262   \exp_args:No \__stex_structures_begin:nn\l__stex_structures_exstruct_name_str{}
6263
6264   \seq_map_inline:Nn \l__stex_structures_imports_seq {
6265     \stex_if_do_html:T {
6266       \stex_annotate_invisible:nn
6267         {shtml:import= {##1}} {}
6268     }
6269     \stex_module_add_morphism:nonn
6270       {}{##1}{import}{}
6271     \stex_execute_in_module:x{
6272       \stex_activate_module:n{##1}
6273     }
6274   }
6275
6276   \stex_smsmode_do:
6277 }{
6278   \prop_map_inline:cn{
6279     c_stex_module_ \l_stex_current_module_str _symbols_prop
6280   }{
6281     \__stex_structures_check_def:nnnnnnn ##2

```

```

6282 }
6283 \stex_structural_feature_module_end:
6284 \__stex_structures_do externals:
6285 }{}{}{}
6286
6287 \stex_sms_allow_env:n{extstructure*}
6288 \exp_after:wN \stex_deactivate_macro:Nn
6289 \cs:w extstructure*\cs_end: {module-environments}
6290 \stex_every_module:n {
6291 \exp_after:wN \stex_reactivate_macro:N \cs:w extstructure*\cs_end:
6292 }
6293
6294 \cs_new_protected:Nn \__stex_structures_check_def:nnnnnnn {
6295 \tl_if_empty:nT{#5}{
6296 \msg_error:nnnn{stex}{error/needsdefiniens}{#2}{extstructure*}
6297 }
6298 }
6299
6300 \stex_every_module:n{ \str_set:Nn \l__stex_structures_extname_count 0}
6301
6302 \cs_new_protected:Nn \__stex_structures_new_extstruct_name: {
6303 \stex_do_up_to_module:n {
6304 \str_set:Nx \l__stex_structures_extname_count {\int_eval:n{\l__stex_structures_extname_c
6305 }
6306 \str_set:Nx \l__stex_structures_exstruct_name_str {EXTSTRUCT\l__stex_structures_extname_c
6307 }
6308

```

Invoking structures:

```

6309 \cs_new_protected:Nn \stex_invoke_structure: {
6310 \tl_set:Nn \l__stex_structures_set_comp_tl {\__stex_structures_set_thiscomp:}
6311 \__stex_structures_invoke_top:n {}
6312 }
6313
6314 \cs_new_protected:Nn \__stex_structures_invoke_top:n {
6315 \stex_debug:nn{structure}{
6316 invoking-structure~{\l_stex_current_type_tl}<\tl_to_str:n{#1}>
6317 }
6318 \peek_charcode:NTF [ {
6319 \__stex_structures_merge:nw{#1}
6320 }{
6321 \__stex_structures_invokation_type:n {#1}
6322 \tl_set:Nn \l__stex_structures_this_tl {}
6323 \peek_charcode_remove:NTF ! {
6324 \peek_charcode:NTF [ {
6325 \__stex_structures_maybe_notation:w
6326 }{
6327 \__stex_structures_maybe_notation:w []
6328 }
6329 }{
6330 \__stex_structures_invoke_this:n
6331 }
6332 }
6333 }

```

```

6334
6335 \cs_new_protected:Npn \__stex_structures_merge:nw #1 [ #2 ] {
6336   \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{comp=}{
6337     \__stex_structures_set_customcomp: #2 \__stex_structures_end:
6338     \__stex_structures_invoke_top:n{#1}
6339   }{
6340     \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{this=}{
6341       \__stex_structures_set_thisnotation: #2 \__stex_structures_end:
6342       \__stex_structures_invoke_top:n{#1}
6343     }{
6344       \tl_if_empty:nTF{#1}{
6345         \__stex_structures_invoke_top:n{#2}
6346       }{
6347         \tl_if_empty:nTF{#2}{
6348           \__stex_structures_invoke_top:n{#1}
6349         }{
6350           \__stex_structures_invoke_top:n{#1,#2}
6351         }
6352       }
6353     }
6354   }
6355 }
6356
6357 \cs_new_protected:Npn \__stex_structures_set_thisnotation: this= #1 \__stex_structures_end: {
6358   \tl_set:Nn \l_stex_return_notation_tl { \comp{#1} }
6359   \tl_set:Nn \l__stex_structures_set_comp_tl {}
6360 }
6361
6362 \cs_new_protected:Npn \__stex_structures_set_customcomp: comp= #1 \__stex_structures_end: {
6363   \tl_set:Nn \l__stex_structures_set_comp_tl {
6364     \__stex_structures_set_custom_comp:n{#1}
6365   }
6366   \tl_set:Nn \l_stex_return_notation_tl { \comp{} }
6367 }

```

The structure type:

```

6368 \cs_new_protected:Nn \__stex_structures_invokation_type:n {
6369   \__stex_structures_do_assign_list:n{#1}
6370   \clist_if_empty:NTF \l__stex_structures_fields_clist {
6371     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6372       = 1 {
6373         \tl_set:Nx \l__stex_structures_current_type_tl {
6374           \exp_args:No \exp_not:n \l_stex_current_redo_tl
6375           \stex_term_oms_or_omv:nnn{}{}{}
6376         }
6377       }{
6378         \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6379       }
6380   }{
6381     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6382       = 1 {
6383         \__stex_structures_make_type:n {}
6384       }{
6385         \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6386       }

```

```

6387 }
6388 }
6389
6390 \cs_new_protected:Nn \__stex_structures_do_assign_list:n {
6391   \clist_clear:N \l__stex_structures_fields_clist
6392   \tl_if_empty:nF {#1} {
6393     \keyval_parse:NNn\TODO\__stex_structures_do_assign:nn{#1}
6394   }
6395 }
6396
6397 \cs_new_protected:Nn \__stex_structures_do_assign:nn {
6398   \clist_put_right:Nn \l__stex_structures_fields_clist {{#1}{#2}}
6399 }
6400
6401 \cs_new_protected:Nn \__stex_structures_make_type:n {
6402   \tl_if_empty:nTF{#1}{
6403     \seq_clear:N \l_tmpa_seq
6404   }{
6405     \seq_set_split:Nnn \l_tmpa_seq ,{#1}
6406     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
6407     \seq_reverse:N \l_tmpa_seq
6408   }
6409   \tl_set:Nx \l__stex_structures_current_type_tl {
6410     \symuse{Metatheory?module~type~merge}{
6411       {
6412         \exp_args:No \exp_not:n \l_stex_current_redo_tl
6413         \stex_term_oms_or_omv:nnn{}{}{}
6414       }
6415       \seq_map_function:NN \l_tmpa_seq \__stex_structures_make_mod:n
6416       \clist_if_empty:NF \l__stex_structures_fields_clist {
6417         ,\symuse{Metatheory?anonymous~record}{
6418           \exp_args:Ne \tl_tail:n{
6419             \clist_map_function:NN \l__stex_structures_fields_clist \__stex_structures_make_
6420           }
6421         }
6422       }
6423     }
6424   }
6425 }
6426
6427 \cs_new:Nn \__stex_structures_make_mod:n {
6428   ,\symuse{Metatheory?module~type}{
6429     \stex_annotate:nn{shtml:term=OMMOD,shtml:head={#1}}{}
6430   }
6431 }
6432
6433 \cs_new:Nn \__stex_structures_make_oml:n {
6434   \__stex_structures_make_oml:nn #1
6435 }
6436 \cs_new:Nn \__stex_structures_make_oml:nn {
6437   ,\stex_annotate:nn{
6438     shtml:term=OML,
6439     shtml:head={#1}
6440   }{

```

```

6441     \stex_annotate_force_break:n{
6442         \stex_annotate:nn{shtml:definien={}}{\exp_not:n{#2!}}
6443     }
6444 }
6445 }

```

Insert the structure type as a term:

```

6446 \cs_new:Nn \__stex_structures_current_type: {
6447     %\exp_args:No \exp_not:n \l_stex_current_redo_tl
6448     \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6449         \exp_args:No\exp_not:n\l__stex_structures_current_type_tl
6450     }
6451     \stex_term_oms_or_omv:nnn{}{}{}
6452 }

```

The structure type itself:

```

6453 \cs_new_protected:Npn \__stex_structures_maybe_notation:w [ #1 ] {
6454     \tl_set_eq:NN \l_stex_current_term_tl \l_stex_structures_current_type_tl
6455     \cs_if_exist:cTF{l_stex_notation_\l_stex_current_symbol_str_#1_cs}{
6456         \use:c{l_stex_notation_\l_stex_current_symbol_str_#1_cs}\group_end:
6457     }{
6458         \__stex_structures_make_prop:
6459         \__stex_structures_make_prop_assign:
6460         \__stex_structures_present_i:w [ #1 ]
6461     }
6462 }
6463
6464 \cs_new_protected:Nn \__stex_structures_present: {
6465     \peek_charcode:NTF [ {
6466         \__stex_structures_present_i:w
6467     }{
6468         \__stex_structures_present:nn{}{}
6469     }
6470 }
6471
6472 \cs_new_protected:Npn \__stex_structures_present_i:w [ #1 ] {
6473     \int_compare:nNnTF{\clist_count:n{#1}} = 1 {
6474         \__stex_structures_present:nn{}{#1}
6475     }{
6476         \peek_charcode:NTF [ {
6477             \__stex_structures_present_ii:nw{#1}
6478         }{
6479             \__stex_structures_present:nn{#1}{}
6480         }
6481     }
6482 }
6483
6484 %First: clist, second:notation-id
6485 \cs_new_protected:Npn \__stex_structures_present_ii:nw #1 [ #2 ] {
6486     \__stex_structures_present:nn{#1}{#2}
6487 }
6488
6489 \cs_new_protected:Nn \__stex_structures_present:nn {
6490     \clist_clear:N \l__stex_structures_clist
6491     \tl_if_empty:nTF{#1}{

```



```

6492     \cs_set:Npn \l__stex_structures_cs ##1 ##2 ##3 {
6493         \tl_if_empty:nF{##2}{
6494             \__stex_structures_present_entry:nn {##1}{##3}
6495         }
6496     }
6497 }{
6498     \cs_set:Npn \l__stex_structures_cs ##1 ##2 ##3 {
6499         \exp_args:Ne \clist_if_in:nnT{\tl_to_str:n{##1}}{##1}{
6500             \__stex_structures_present_entry:nn {##1}{##3}
6501         }
6502     }
6503 }
6504 \prop_map_inline:Nn \l__stex_structures_prop {
6505     \l__stex_structures_cs {##1} ##2
6506 }
6507 \__stex_term_oms_or_omv:nnn{}{}{
6508     \exp_args:Nno \use:n{
6509         \bool_set_true:N \l_stex_allow_semantic_bool
6510         \symuse{Metatheory?mathematical~structure}[#2]
6511     }{\l__stex_structures_clist}
6512 }\group_end:
6513 }
6514
6515 \cs_new_protected:Nn \__stex_structures_present_entry:nn {
6516     \seq_if_in:NnTF \l__stex_structures_assigned_seq {##1}{
6517         \clist_put_right:Nn \l__stex_structures_clist {##2!}
6518     }{
6519         \exp_args:Nne \clist_put_right:Nn \l__stex_structures_clist {
6520             \__stex_next_symbol:n {
6521                 \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
6522                 \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
6523                     \exp_args:No \exp_not:n \l__stex_structures_this_tl
6524                 }
6525                 \exp_not:n {
6526                     \tl_set_eq:NN \this \l__stex_structures_this_tl
6527                 }
6528                 \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6529                     \exp_args:No \exp_not:n \l_stex_return_notation_tl
6530                 }
6531             }
6532             \exp_not:n{##2!}
6533         }
6534     }
6535 }
6536
6537 \cs_new_protected:Npn \_thiscomp #1 #2 {
6538     {\tl_set:cn{this}{{}}#1{##2}\c_math_subscript_token{
6539         \group_begin:
6540         \bool_set_true:N \l_stex_allow_semantic_bool
6541         \l__stex_structures_this_tl
6542         \group_end:
6543     }
6544 }
6545 }

```

```

6546
6547 \cs_new_protected:Nn \__stex_structures_set_thiscomp: {
6548   \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_thiscomp {
6549     \edef\maincomp {\_thiscomp{\comp}}
6550   }
6551 }
6552
6553 \cs_new_protected:Nn \__stex_structures_set_custom_comp:n {
6554   \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
6555     \cs_set_protected:Npx \_customthiscomp ##1 {
6556       \group_begin:
6557         \bool_set_true:N \l_stex_allow_semantic_bool
6558         \exp_not:n{
6559           \cs_set:Npn \l__stex_structures_comp_cs ##1 {
6560             #1
6561           }
6562           \def\maincomp
6563             }{\comp}
6564           \exp_not:N\l__stex_structures_comp_cs{\comp{##1}}
6565         \group_end:
6566       }
6567     \def\maincomp {\_customthiscomp}
6568   }
6569 }
6570

```

this (of type structure):

```

6571
6572 \cs_new_protected:Nn \__stex_structures_invoke_this:n {
6573   \peek_charcode_remove:NTF ! {
6574     \exp_args:Nne\use:nn{
6575       \group_end:\symuse{Metatheory?of~type}[invisible]{#1}
6576     }{
6577       {\__stex_structures_current_type:}
6578     }
6579   }{
6580     \__stex_structures_invoke_maybe_field:nn{#1}
6581   }
6582 }
6583
6584 \cs_new_protected:Nn \__stex_structures_invoke_maybe_field:nn {
6585   \__stex_structures_make_prop:
6586   \__stex_structures_set_this:n{#1}
6587   \tl_if_empty:nTF{#2}{
6588     \__stex_structures_make_prop_assign:
6589     \__stex_structures_present:
6590   }{
6591     \__stex_structures_invoke_field:n{#2}
6592   }
6593 }
6594
6595 \cs_new_protected:Nn \__stex_structures_set_this:n {
6596   \tl_if_empty:nTF{#1}{
6597     %\tl_put_right:Nn \l_stex_current_redo_tl {
6598     % \tl_clear:N \l__stex_structures_this_tl

```

```

6599   %}
6600   }{
6601     \tl_set:Nx \l__stex_structures_this_tl {{
6602       \bool_set_true:N \l_stex_allow_semantic_bool
6603       \tl_set:Nn \exp_not:N \this {
6604         \exp_args:No \exp_not:n \this
6605       }
6606       \exp_not:n{#1}
6607     }}
6608     \tl_set_eq:NN \this \l__stex_structures_this_tl
6609     % \l_stex_return_notation_tl
6610   }
6611 }
6612
6613 \cs_new_protected:Nn \__stex_structures_get_field_name:n {
6614   \str_set:Nx \l__stex_structures_field_name_str {
6615     \exp_args:Nne \use:n {\exp_after:wN \use_i:nn \use:n}
6616     {\prop_item:Nn \l__stex_structures_prop {#1}}
6617   }
6618   \str_if_empty:NT \l__stex_structures_field_name_str {
6619     \str_set:Nn \l__stex_structures_field_name_str {#1}
6620   }
6621 }
6622
6623 \cs_new_protected:Nn \__stex_structures_invoke_field:n {
6624   \prop_if_in:NnTF \l__stex_structures_prop {#1}{
6625     \__stex_structures_get_field_name:n{#1}
6626     \tl_clear:N \l__stex_structures_more_nextsymbol_tl
6627     %\exp_args:Nne \seq_if_in:NnF \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}{
6628       \tl_set:Nx \l__stex_structures_more_nextsymbol_tl {
6629         \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
6630           \exp_args:No \exp_not:n \l__stex_structures_this_tl
6631         }
6632         \exp_not:n {
6633           \tl_set_eq:NN \this \l__stex_structures_this_tl
6634         }
6635         \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6636           \exp_args:No \exp_not:n \l_stex_return_notation_tl
6637         }
6638         \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
6639       }
6640     %}
6641     \exp_args:NNx \use:nn \group_end: {
6642       \__stex_next_symbol:n {
6643         \exp_args:No \exp_not:n \l__stex_structures_redo_tl
6644         \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6645           \symuse{Metatheory?record~field}{
6646             \symuse{Metatheory?of~type}{
6647               \exp_args:No \exp_not:n \l__stex_structures_this_tl
6648             }{ \__stex_structures_current_type: }
6649           }{
6650             \stex_annotate:nn{shtml:term=OML,shtml:head={\l__stex_structures_field_name_str}
6651           }
6652         }

```

```

6653     \exp_args:No \exp_not:n \l__stex_structures_more_nextsymbol_tl
6654   }
6655   \exp_not:N \use_ii:nn
6656   \prop_item:Nn \l__stex_structures_prop {#1}
6657 }
6658 }{
6659   \msg_error:nnn{stex}{error/unknownfield}{#1}
6660 }
6661 }
6662
6663 \cs_new_protected:Nn \__stex_structures_make_prop: {
6664   \prop_clear:N \l__stex_structures_prop
6665   \seq_clear:N \l__stex_structures_seq
6666   \seq_clear:N \l__stex_structures_assigned_seq
6667   \tl_clear:N \l__stex_structures_redo_tl
6668   \__stex_structures_prop_do_decls:
6669   \__stex_structures_prop_do_notations:
6670 }
6671
6672 \cs_new_protected:Nn \__stex_structures_make_prop_assign: {
6673   \clist_if_empty:NF \l__stex_structures_fields_clist {
6674     \clist_map_inline:Nn \l__stex_structures_fields_clist {
6675       \__stex_structures_make_prop_assign:nn ##1
6676     }
6677   }
6678 }
6679
6680 \cs_new_protected:Nn \__stex_structures_make_prop_assign:nn {
6681   \prop_if_in:NnTF \l__stex_structures_prop {#1}{
6682     \exp_args:Nne \seq_put_right:Nn \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}
6683     \exp_args:Nne \use:nn {\__stex_structures_make_prop_assign_replace:nnnn {#1}{#2}}
6684     {\prop_item:Nn \l__stex_structures_prop {#1}}
6685   }{
6686     \msg_error:nnn{stex}{error/unknownfieldass}{#1}
6687   }
6688 }
6689
6690 \cs_new_protected:Nn \__stex_structures_make_prop_assign_replace:nnnn {
6691   \prop_put:Nnn \l__stex_structures_prop {#1}{#3}{#2}
6692   \tl_if_empty:nF{#3}{
6693     \tl_set:cn{#1}{ #2 }
6694     \tl_put_right:Nn \l__stex_structures_redo_tl {
6695       \tl_set:cn{#1}{ #2 }
6696     }
6697   }
6698 }
6699
6700 \cs_new_protected:Nn \__stex_structures_prop_do_decls: {
6701   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_type_tl {
6702     \tl_if_empty:nTF{##2}{
6703       \__stex_structures_do_decl_nomacro:nnnnnnnn{##3}
6704     }{
6705       \__stex_structures_do_decl:nnnnnnnn{##2}
6706     }
6707     {##1}{##3}{##4}{##5}{##6}{##7}{##8}{##9}

```

```

6707 }
6708 }
6709
6710 \cs_new_protected:Nn \__stex_structures_do_decl_nomacro:nnnnnnnn {
6711   \prop_if_in:NnF \l__stex_structures_prop {#1} {
6712     \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
6713     \prop_put:Nnn \l__stex_structures_prop {#1}{
6714       }{
6715         \stex_invoke_symbol:nnnnnnN
6716         {#2}
6717         {#3}
6718         {#4}{#5}{#6}{#7}{#8}#9
6719       }
6720     }
6721   }
6722 }
6723
6724 \cs_new_protected:Nn \__stex_structures_do_decl:nnnnnnnn {
6725   \prop_if_in:NnF \l__stex_structures_prop {#1} {
6726     \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
6727     \prop_put:Nnn \l__stex_structures_prop {#1}{
6728       {#3}{
6729         \stex_invoke_symbol:nnnnnnN
6730         {#2}
6731         {#3}
6732         {#4}{#5}{#6}{#7}{#8}#9
6733       }
6734     }
6735   }
6736   %\tl_set:cn{#1}{
6737   % \stex_invoke_symbol:nnnnnnN
6738   % {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
6739   %}
6740   %\tl_put_right:Nn \l__stex_structures_redo_tl {
6741   % \tl_set:cn{#1}{
6742   % \stex_invoke_symbol:nnnnnnN
6743   % {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
6744   % }
6745   %}
6746 }
6747
6748 \cs_new_protected:Nn \__stex_structures_prop_do_notations: {
6749   \exp_args:No \stex_iterate_notations:nn\l_stex_current_type_tl{
6750     \exp_args:NNe \seq_if_in:NnT \l__stex_structures_seq {\tl_to_str:n{##1}}{
6751       \tl_put_right:Nn \l__stex_structures_redo_tl {
6752         \cs_if_exist:cF{l_stex_notation_##1_##2_cs}{
6753           \tl_set:cn{l_stex_notation_##1_##2_cs}{##4}
6754         }
6755         \cs_if_exist:cF{l_stex_notation_##1_##2_cs}{
6756           \tl_set:cn{l_stex_notation_##1_##2_cs}{##4}
6757         }
6758       }
6759     \cs_if_exist:cF{l_stex_notation_##1_##2_cs}{
6760       \tl_set:cn{l_stex_notation_##1_##2_cs}{##4}

```

```

6761     }
6762     \cs_if_exist:cF{l_stex_notation_##1 __cs}{
6763       \tl_set:cn{l_stex_notation_##1 __cs}{##4}
6764     }
6765     \tl_if_empty:nF{##5}{
6766       \tl_put_right:Nn \l__stex_structures_redo_tl {
6767         \cs_if_exist:cF{l_stex_notation_##1 _op_##2_cs}{
6768           \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
6769         }
6770         \cs_if_exist:cF{l_stex_notation_##1 _op__cs}{
6771           \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
6772         }
6773       }
6774       \cs_if_exist:cF{l_stex_notation_##1 _op_##2_cs}{
6775         \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
6776       }
6777       \cs_if_exist:cF{l_stex_notation_##1 _op__cs}{
6778         \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
6779       }
6780     }
6781   }
6782 }
6783 }

```

### `\usestructure`

```

6784 \cs_new_protected:Npn \usestructure #1 {
6785   \stex_get_mathstructure:n{ #1 }
6786   \seq_clear:N \l__stex_structures_imports_seq
6787   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6788     \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6789       \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6790     }
6791   }
6792   \seq_map_inline:Nn \l__stex_structures_imports_seq {
6793     \stex_if_do_html:T {
6794       \hbox{\stex_annotate_invisible:nn
6795         {shtml:usemodule=##1} {}}
6796     }
6797     \stex_activate_module:n {##1}
6798   }
6799 }

```

(End of definition for `\usestructure`. This function is documented on page 96.)

## 13.10 Statements

```

6800 <@@=stex_statements>
6801
6802 \stex_keys_define:nnnn{statement}{
6803   \str_clear:N \l_stex_key_name_str
6804   \str_clear:N \l_stex_key_macroname_str
6805   \clist_clear:N \l_stex_key_for_clist
6806   \str_clear:N \l_stex_key_args_str

```

```

6807 \tl_clear:N \l_stex_key_type_tl
6808 \tl_clear:N \l_stex_key_def_tl
6809 \tl_clear:N \l_stex_key_return_tl
6810 \clist_clear:N \l_stex_key_argtypes_clist
6811 }{
6812   name .str_set:N = \l_stex_key_name_str ,
6813   for .clist_set:N = \l_stex_key_for_clist ,
6814   macro .str_set:N = \l_stex_key_macroname_str ,
6815   % start .str_set:N = \l_stex_key_title_str , % TODO remove
6816   type .tl_set:N = \l_stex_key_type_tl ,
6817   judgment .code:n = {},
6818   from .code:n= {}, % TODO remove
6819   to .code:n={} % TODO remove
6820 }{id,title,style,symargs}

```

\stex\_new\_statement:nn

```

6821 \cs_new_protected:Npn \_stex_do_for_list: {
6822   \seq_clear:N \l_stex_fors_seq
6823   \clist_map_inline:Nn \l_stex_key_for_clist {
6824     \exp_args:Ne \stex_get_symbol:n{\tl_to_str:n{##1}}
6825     \seq_put_right:Nx \l_stex_fors_seq
6826       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6827   }
6828 }
6829
6830 \cs_new_protected:Nn \__stex_statements_setup:nn {
6831   \str_if_empty:NF \l_stex_key_macroname_str {
6832     \str_if_empty:NT \l_stex_key_name_str {
6833       \str_set_eq:NN \l_stex_key_name_str \l_stex_key_macroname_str
6834     }
6835   }
6836   \_stex_do_for_list:
6837   \str_if_empty:NF \l_stex_key_name_str {
6838     \__stex_statements_force_id:
6839     \seq_put_right:Nx \l_stex_fors_seq {
6840       \l_stex_current_module_str ? \l_stex_key_name_str
6841     }
6842     \str_set_eq:NN \l_stex_macroname_str \l_stex_key_macroname_str
6843     \str_set:Nn \l_stex_key_role_str {#2}
6844     \stex_symdecl_do:
6845     \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnN}{
6846       {\l_stex_key_macroname_str}{\l_stex_key_name_str}
6847       {\int_use:N \l_stex_get_symbol_arity_int}
6848       {\l_stex_get_symbol_args_tl}
6849       {#1}{\stex_invoke_symbol:
6850     }
6851     \stex_if_do_html:T \stex_symdecl_html:
6852   }
6853   \str_clear:N \l__stex_statements_uri_str
6854   \str_if_empty:NTF \l_stex_key_name_str {
6855     \stex_debug:nn{statement}{no~name}
6856     \int_compare:nNnTF {\seq_count:N \l_stex_fors_seq} = 1 {
6857       \str_set:Nx \l__stex_statements_uri_str {\seq_item:Nn \l_stex_fors_seq 1}
6858       \stex_debug:nn{statement}{for:~\l__stex_statements_uri_str}

```

```

6859     }{
6860     \stex_debug:nn{statement}{no~for}
6861     }
6862   }{
6863     \str_set:Nx \l__stex_statements_uri_str {\l_stex_current_module_str ? \l_stex_key_name_s
6864     \stex_debug:nn{statement}{name:~\l__stex_statements_uri_str}
6865     }
6866   }
6867
6868   \cs_new:Nn \__stex_statements_html_keyvals:nn {
6869     shtml:#1={},
6870     shtml:inline={#2},
6871     \seq_if_empty:NF \l_stex_fors_seq {,
6872     shtml:fors={\seq_use:Nn \l_stex_fors_seq ,}
6873     }
6874     \str_if_empty:NF \l_stex_key_id_str {,
6875     shtml:id={\stex_uri_use:N \l_stex_current_doc_uri ? \l_stex_key_id_str}
6876     }
6877     \clist_if_empty:NF \l_stex_key_style_clist {,
6878     shtml:styles={\l_stex_key_style_clist}
6879     }
6880   }
6881
6882   \cs_new_protected:Nn \stex_new_statement:nnn {
6883     \stex_new_stylable_env:nnnnnn {#1}{0{}}{
6884     \stex_keys_set:nn{statement}{##1}
6885     #3
6886
6887     \stex_if_smsmode:F {
6888     \exp_args:Nne \begin{stex_annotate_env}{
6889     \__stex_statements_html_keyvals:nn{#1}{false}
6890     }
6891     \tl_set_eq:NN \thistitle \l_stex_key_title_tl
6892     \str_set_eq:NN \thisname \l_stex_key_name_str
6893     \clist_set_eq:NN \thisfor \l_stex_key_for_str
6894     \stex_if_html_backend:TF {
6895     \noindent
6896     \stex_annotate:nn{shtml:statementtitle={}}{\_stex_annotate_force_break:n\l_stex_key
6897     }
6898     \stex_style_apply:
6899     }
6900     \_stex_do_id:
6901     \stex_smsmode_do:
6902   }{
6903     \stex_if_smsmode:F {
6904     \stex_if_html_backend:F \stex_style_apply:
6905     \end{stex_annotate_env}
6906     }
6907   }{}{s}
6908   \stex_sms_allow_env:n{s#1}
6909
6910   \tl_if_empty:nF{#2}{
6911     \exp_after:wN \NewDocumentCommand \cs:w inline#2\cs_end: { 0{ } m}{
6912     \group_begin:

```



```

6913     \stex_keys_set:nn{statement}{##1}
6914     #3
6915     \_stex_do_id:
6916     \stex_if_smsmode:F{
6917         \exp_args:Ne \stex_annotate:nn{\_stex_statements_html_keyvals:nn{#1}{true}}{
6918             \_stex_annotate_force_break:n{##2}
6919         }
6920     }
6921     \group_end:
6922     \stex_smsmode_do:
6923 }
6924 \exp_after:wN \stex_sms_allow_escape:N\cs:w inline#2\cs_end:
6925 }
6926 }
6927
6928 \cs_new_protected:Nn \_stex_statements_setup_def: {
6929     \stex_if_smsmode:F{
6930         \seq_map_inline:Nn \l_stex_fors_seq {
6931             \stex_ref_new_sym_target:n{##1}
6932         }
6933     }
6934     \stex_reactivate_macro:N \definiendum
6935     \stex_reactivate_macro:N \defnotation
6936     \stex_reactivate_macro:N \definame
6937     \stex_reactivate_macro:N \Definame
6938     \stex_reactivate_macro:N \varbind
6939 }
6940
6941 \cs_new_protected:Nn \_stex_statements_force_id: {
6942     \str_if_empty:NT \l_stex_key_id_str {
6943         \_stex_ref_new_id:n{
6944             \str_set_eq:NN \l_stex_key_id_str \l__stex_refs_str
6945         }
6946     }
6947 }
6948
6949 \stex_new_statement:nnn{definition}{def}{
6950     \_stex_statements_force_id:
6951     \_stex_statements_setup:nn{}{}
6952     \_stex_statements_setup_def:
6953     \stex_reactivate_macro:N \definiens
6954 }
6955
6956 \stex_new_statement:nnn{assertion}{ass}{
6957     \_stex_statements_setup:nn{}{assertion}
6958     \stex_if_smsmode:F{
6959         \seq_map_inline:Nn \l_stex_fors_seq {
6960             \stex_ref_new_sym_target:n{##1}
6961         }
6962     }
6963     \stex_reactivate_macro:N \varbind
6964     \stex_reactivate_macro:N \conclusion
6965     \stex_reactivate_macro:N \premise
6966     \stex_reactivate_macro:N \definiendum
6967     \stex_reactivate_macro:N \defnotation
6968     \stex_reactivate_macro:N \definame

```

```

6967 \stex_reactivate_macro:N \Definame
6968 }
6969 \stex_new_statement:nnn{example}{ex}{\__stex_statements_setup:nn{}{example}}
6970 \stex_new_statement:nnn{paragraph}{}{}
6971 \clist_if_in:NnTF \l_stex_key_style_clist {syndoc}{
6972   \__stex_statements_force_id:
6973   \__stex_statements_setup:nn{}{}
6974   \__stex_statements_setup_def:
6975 }{
6976   \__stex_statements_setup:nn{}{}
6977 }
6978 }

```

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

### definiendum

```

6979 \cs_new_protected:Nn \__stex_statements_do_defref:nn {
6980   \stex_if_html_backend:T{\ifvmode\indent\fi}
6981   \group_begin:
6982   \stex_get_symbol:n{#1}
6983   \bool_if:NTF \l_stex_allow_semantic_bool{
6984     \str_set:Nx\l_stex_current_symbol_str
6985       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6986     \str_if_in:NnT \l_stex_get_symbol_name_str / {
6987       \str_set:Nx \l_stex_get_symbol_name_str {
6988         \exp_after:wN \stex_split_slash: \l_stex_get_symbol_name_str
6989         /\_stex_args_end:
6990       }
6991     }
6992     \exp_args:No \stex_ref_new_sym_target:n \l_stex_current_symbol_str
6993     \def\comp{\_defcomp}
6994     \stex_annotate:nn{shtml:definiendum=\l_stex_current_symbol_str}{\comp{#2}}
6995   }{
6996     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
6997   }
6998   \group_end:
6999 }
7000
7001 \NewDocumentCommand \defnotation{ m } {
7002   \_stex_next_symbol:n { \def\comp{\_defcomp}}#1
7003 }
7004 \stex_deactivate_macro:Nn \defnotation {definition~environments}
7005
7006 \NewDocumentCommand \definiendum { O{} m m } {
7007   \stex_keys_set:nn{symname}{ #1 }
7008   \__stex_statements_do_defref:nn{#2}{#3}
7009 }
7010 \stex_deactivate_macro:Nn \definiendum {definition~environments}
7011
7012 \NewDocumentCommand \definame { O{} m } {
7013   \stex_keys_set:nn{symname}{#1}
7014   \__stex_statements_do_defref:nn{#2}{
7015     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
7016   }

```

```

7017 }
7018 \stex_deactivate_macro:Nn \definame {definition-environments}
7019
7020 \NewDocumentCommand \Definame { 0{} m } {
7021   \stex_keys_set:nn{symname}{#1}
7022   \__stex_statements_do_defref:nn{#2}{
7023     \l_stex_key_pre_tl\exp_after:wN\l_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
7024   }
7025 }
7026 \stex_deactivate_macro:Nn \Definame {definition-environments}
7027
7028
7029 \NewDocumentCommand \definiens { 0{} m }{
7030   \group_begin:
7031   \str_clear:N \l_stex_get_symbol_name_str
7032   \tl_if_empty:nF {#1} {
7033     \stex_get_symbol:n { #1 }
7034     \str_set:Nx \l__stex_statements_uri_str
7035       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7036   }
7037   \str_if_empty:NT \l__stex_statements_uri_str {
7038     \msg_error:nn{stex}{error/definiensfor}
7039   }
7040   \stex_debug:nn{definiens}{Checking~\l__stex_statements_uri_str}
7041
7042   \exp_args:No \stex_add_definiens:nn \l__stex_statements_uri_str{#2}
7043
7044   \group_end:
7045   \stex_smsmode_do:
7046 }
7047 \stex_deactivate_macro:Nn \definiens {definition-environments}
7048 \stex_sms_allow_escape:N \definiens
7049
7050 \cs_new_protected:Nn \stex_add_definiens:nn {
7051   \exp_args:Nno \stex_str_if_starts_with:nnT{#1} \l_stex_current_module_str {
7052     \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _symbols_prop}{
7053       \stex_debug:nn{definiens}{#1 == \l_stex_current_module_str?##1}
7054       \str_if_eq:noT {#1} {\l_stex_current_module_str?##1}{
7055         \prop_map_break:n{\stex_add_definiens_inner:nnnnnnn ##2}
7056       }
7057     }
7058   }
7059   \stex_if_smsmode:F{
7060     \stex_annotate:nn{ shtml:definiens={#1}}{
7061       #2 %\stex_annotate_force_break:n{ #2 }
7062     }
7063   }
7064 }
7065
7066 \cs_new_protected:Nn \stex_add_definiens_inner:nnnnnnn {
7067   \stex_debug:nn{definiens}{Adding~definiens~to~\l_stex_current_module_str?#2}
7068   \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
7069     {#2}{{#1}{#2}{#3}{#4}{defed}{#6}{#7}{#8}}
7070 }

```

```

7071
7072 \NewDocumentCommand \varbind {m} {
7073   \clist_map_inline:nn {#1} {
7074     \stex_get_var:n {##1}
7075     \stex_if_do_html:T {
7076       \stex_annotate_invisible:nn {shtml:bind=\l_stex_get_symbol_name_str}{}
7077     }
7078   }
7079 }
7080 \stex_deactivate_macro:Nn \varbind {definition~or~assertion~environments}
7081
7082 \NewDocumentCommand \conclusion { 0{} m} {
7083   \group_begin:
7084   \str_clear:N \l_stex_get_symbol_name_str
7085   \tl_if_empty:nF {#1} {
7086     \stex_get_symbol:n { #1 }
7087     \str_set:Nx \l__stex_statements_uri_str
7088       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7089   }
7090   \str_if_empty:NT \l__stex_statements_uri_str {
7091     \msg_error:nn{stex}{error/conclusionfor}
7092   }
7093   \stex_annotate:nn{ shtml:conclusion=\l__stex_statements_uri_str}{
7094     #2 %\stex_annotate_force_break:n{ #2 }
7095   }
7096   \group_end:
7097 }
7098 \stex_deactivate_macro:Nn \conclusion {assertion~environments}
7099
7100 \NewDocumentCommand \premise {0{} m} {
7101   \tl_if_empty:nF {#1} {
7102     \stex_debug:nn{Here:}{Variable~#1}
7103     \exp_args:Nne\use:nn{\vardef}{v#1}[name=#1]{#1}}
7104   }
7105   \stex_annotate:nn{shtml:premise={#1}}{#2}
7106 }
7107 \stex_deactivate_macro:Nn \premise {assertion~environments}

```

(End of definition for definiendum. This function is documented on page 100.)

## 13.11 Proofs

We first define some keys for the sproof environment.

```

7108 <@@=stex_proof>
7109 \stex_keys_define:nnnn{ spf }{
7110   \tl_clear:N \l_stex_key_for_clist
7111   \tl_clear:N \l_stex_key_from_tl
7112   \tl_set_eq:NN \l_stex_key_proofend_tl \l__stex_proof_proof_box_tl
7113   \tl_clear:N \l_stex_key_continues_tl
7114   \tl_clear:N \l_stex_key_term_tl
7115   \tl_clear:N \l_stex_key_functions_tl
7116   \tl_clear:N \l_stex_key_method_tl
7117   \bool_set_false:N \l_stex_key_hide_bool

```

```

7118 }{
7119   for      .clist_set:N = \l_stex_key_for_clist ,
7120   from     .tl_set:N    = \l_stex_key_from_tl ,
7121   proofend .tl_set:N    = \l_stex_key_proofend_tl,
7122   continues.tl_set:N    = \l_stex_key_continues_tl,
7123   functions.tl_set:N    = \l_stex_key_functions_tl,
7124   term     .tl_set:N    = \l_stex_key_term_tl,
7125   method  .tl_set:N    = \l_stex_key_method_tl,
7126   hide    .bool_set:N   = \l_stex_key_hide_bool
7127 }{id,style,title}
7128
7129 \bool_set_true:N \l__stex_proof_inc_counter_bool

```

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.

```

7130 \intarray_new:Nn\l__stex_proof_counter_intarray{50}
7131 \cs_new_protected:Npn \__stex_proof_insert_number: {
7132   \int_set:Nn \l_tmpa_int {1}
7133   \bool_while_do:mn {
7134     \int_compare_p:nNn {
7135       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7136     } > 0
7137   }{
7138     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .
7139     \int_incr:N \l_tmpa_int
7140   }
7141 }
7142 \cs_new_protected:Nn \__stex_proof_number_as_string:N {
7143   \str_clear:N #1
7144   \int_set:Nn \l_tmpa_int {1}
7145   \bool_while_do:mn {
7146     \int_compare_p:nNn {
7147       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7148     } > 0
7149   }{
7150     \str_put_right:Nx #1 {\intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .}
7151     \int_incr:N \l_tmpa_int
7152   }
7153 }
7154
7155 \cs_new_protected:Npn \__stex_proof_inc_counter: {
7156   \int_set:Nn \l_tmpa_int {1}
7157   \bool_while_do:mn {
7158     \int_compare_p:nNn {
7159       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7160     } > 0
7161   }{
7162     \int_incr:N \l_tmpa_int

```

```

7163 }
7164 \int_compare:nNnF \l_tmpa_int = 1 {
7165   \int_decr:N \l_tmpa_int
7166 }
7167 \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int {
7168   \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int + 1
7169 }
7170 }
7171
7172 \cs_new_protected:Npn \__stex_proof_add_counter: {
7173   \int_set:Nn \l_tmpa_int {1}
7174   \bool_while_do:nn {
7175     \int_compare_p:nNn {
7176       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7177     } > 0
7178   }{
7179     \int_incr:N \l_tmpa_int
7180   }
7181   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 1 }
7182 }
7183
7184 \cs_new_protected:Npn \__stex_proof_remove_counter: {
7185   \int_set:Nn \l_tmpa_int {1}
7186   \bool_while_do:nn {
7187     \int_compare_p:nNn {
7188       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7189     } > 0
7190   }{
7191     \int_incr:N \l_tmpa_int
7192   }
7193   \int_decr:N \l_tmpa_int
7194   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 0 }
7195 }

```

### spfsketch

```

7196 \newenvironment{spfsketchenv}{}{}
7197 \stex_new_stylable_cmd:nnnn{spfsketch}{0}{ m}{\par
7198   \begin{spfsketchenv}
7199   \stex_keys_set:nn{spf}{#1}
7200   \_stex_do_for_list:
7201   \_stex_do_id:
7202   \exp_args:Ne \stex_annotate:nn{
7203     shtml:proofsketch={
7204       \seq_if_empty:NF \l_stex_fors_seq {
7205         \seq_use:Nn \l_stex_fors_seq ,
7206       }
7207     }
7208   }{
7209     \stex_style_apply:
7210     #2
7211   }
7212   \end{spfsketchenv}
7213 }{
7214   \noindent\emph{\spfsketchenvautorefname :}~

```

```
7215 }
```

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

`\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

```
7216 \tl_set:Nn \__stex_proof_proof_box_tl {
7217   \ltx@ifpackageloaded{amssymb}{\square$}{
7218     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
7219   }
7220 }
7221
7222 \tl_set:Nn \sproofend {
7223   \tl_if_empty:NF \l_stex_key_proofend_tl {
7224     \hfil\null\nobreak\hfill\l_stex_key_proofend_tl\par\smallskip
7225   }
7226 }
```

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

`\stexcommentfont`

```
7227 \cs_new_protected:Npn \stexcommentfont {
7228   \small\itshape
7229 }
```

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

`sproof (env.)`

```
7230 \cs_new_protected:Nn \__stex_proof_start_list:n {
7231   \begin{list}{}{
7232     \setlength\topsep{0pt}
7233     \setlength\parsep{0pt}
7234     \setlength\rightmargin{0pt}
7235   }\item[#1]
7236 }
7237 \cs_new_protected:Nn \__stex_proof_end_list: {
7238   \end{list}
7239 }
7240
7241 \cs_new_protected:Nn \__stex_proof_html: {
7242   \stex_annotate_invisible:n{\hbox{
7243     \tl_if_empty:NF \l_stex_key_term_tl {
7244       $\stex_annotate:nn{shtml:proofterm={}}{\l_stex_key_term_tl}$
7245     }
7246     \tl_if_empty:NF \l_stex_key_method_tl {
7247       \stex_annotate:nn{shtml:proofmethod={}}{\l_stex_key_method_tl}
7248     }
7249   }}
7250 }
7251
7252 \cs_new_protected:Nn \__stex_proof_html_env:n {
7253   \exp_args:Nne \begin{stex_annotate_env}{
7254     shtml:#1={
7255       \seq_if_empty:NF \l_stex_fors_seq {
7256         \seq_use:Nn \l_stex_fors_seq ,
```

```

7257     }
7258   }
7259   \bool_if:NT \l_stex_key_hide_bool {,
7260     shtml:proofhide=true
7261   }
7262 }
7263 \__stex_proof_html:
7264 }
7265
7266 \bool_set_false:N \l__stex_proof_in_spfblock_bool
7267 \cs_new_protected:Nn \__stex_proof_begin_proof:nn {\par
7268   \intarray_gzero:N \l__stex_proof_counter_intarray
7269   \intarray_gset:Nnn \l__stex_proof_counter_intarray 1 1
7270   \stex_keys_set:nn{spfsteps}{#1}
7271   \stex_do_for_list:
7272   \stex_if_do_html:T {
7273     \__stex_proof_html_env:n{proof}
7274   }
7275   \seq_map_inline:Nn \l_stex_fors_seq {
7276     \stex_debug:nn{definiens}{Adding~definiens~to~##1}
7277     \stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7278   }
7279   \stex_style_apply:
7280   \__stex_do_id:
7281   \stex_reactivate_macro:N \subproof
7282   \stex_reactivate_macro:N \spfstep
7283   \stex_reactivate_macro:N \conclude
7284   \stex_reactivate_macro:N \assumption
7285   \stex_reactivate_macro:N \eqstep
7286   \stex_reactivate_macro:N \yield
7287   \stex_reactivate_macro:N \spfblock
7288   \stex_reactivate_macro:N \spfjust
7289   \stex_annotate:nn{shtml:prooftitle={}}{#2}
7290   \stex_if_do_html:T{
7291     \begin{stex_annotate_env}{shtml:proofbody={}}
7292   }
7293 }
7294 \stex_new_stylable_env:nnnnnn{proof}{0{} m}{
7295   \__stex_proof_begin_proof:nn{#1}{#2}
7296   \bool_set_true:N\l__stex_proof_in_spfblock_bool\__stex_proof_start_list:n{
7297   \group_begin:\stexcommentfont
7298   }{
7299     \stex_style_apply:
7300     \stex_if_do_html:T{\end{stex_annotate_env}\end{stex_annotate_env}}
7301   }{
7302     \emph{\sproofautorefname :}~
7303   }{
7304     \sproofend
7305   }{s}
7306 \AddToHook{env/sproof/end}{
7307   \bool_if:NT\l__stex_proof_in_spfblock_bool {
7308     \group_end:\__stex_proof_end_list:
7309   }
7310 }

```



```

7311
7312 \stex_new_stylable_env:nnnnnn{proof*}{0{}}{
7313   \__stex_proof_begin_proof:nn{#1}{}
7314   \bool_set_false:N\l__stex_proof_in_spfblock_bool
7315 }{
7316   \stex_style_apply:
7317   \stex_if_do_html:T{\end{stex_annotate_env}\end{stex_annotate_env}}
7318 }{
7319   \emph{Proof:}~
7320 }{
7321   \sproofend
7322 }{s}

```

subproof (*env.*)

```

7323 \str_set_eq:NN \subproofautorefname \spfstepautorefname
7324 \stex_new_stylable_env:nnnnnn{subproof}{s 0{ } m}{\par
7325   \stex_keys_set:nn{spf}{#2}
7326   \__stex_do_for_list:
7327   \stex_if_do_html:T {
7328     \__stex_proof_html_env:n{subproof}
7329   }
7330   \seq_map_inline:Nn \l_stex_fors_seq {
7331     \stex_debug:nn{definiens}{Adding~definiens~to~##1}
7332     \stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7333   }
7334
7335   \IfBooleanTF #1 {
7336     \stex_style_apply:
7337     \str_if_empty:NF \l_stex_key_id_str {
7338       \__stex_proof_number_as_string:N \@currentlabel
7339       \str_set:Nx \@currentHref{subproof.\@currentlabel}
7340       \stex_do_id:
7341     }
7342     \bool_set_false:N \l__stex_proof_in_spfblock_bool
7343     \stex_annotate:nn{shtml:prooftitle={}}{#3}
7344   }{
7345     \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7346       \str_if_empty:NF \l_stex_key_id_str {
7347         \__stex_proof_number_as_string:N \@currentlabel
7348         \str_set:Nx \@currentHref{subproof.\@currentlabel}
7349         \stex_do_id:
7350       }
7351       \__stex_proof_start_list:n\__stex_proof_insert_number:
7352       \stex_annotate:nn{shtml:prooftitle={}}{#3}
7353       \__stex_proof_add_counter:
7354       \stex_style_apply:
7355     }{
7356       \stex_annotate:nn{shtml:prooftitle={}}{#3}
7357       \stex_style_apply:
7358       \stex_do_id:
7359     }
7360   }
7361   \stex_if_do_html:T{
7362     \begin{stex_annotate_env}{shtml:proofbody={}}

```

```

7363 }
7364 \bool_if:NT \l__stex_proof_in_spfblock_bool {\group_begin:\stexcommentfont}
7365 }{
7366   \stex_style_apply:
7367   \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_inc_counter:
7368   \stex_if_do_html:T{\end{stex_annotate_env}}
7369   \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
7370   \stex_if_do_html:T{\end{stex_annotate_env}}
7371   \aftergroup \__stex_proof_inblock_restore:
7372 }{}{}{}
7373 \AddToHook{env/subproof/before}{
7374   \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7375 }
7376 \AddToHook{env/subproof/end}{
7377   \bool_if:NT \l__stex_proof_in_spfblock_bool {
7378     \group_end:\__stex_proof_remove_counter:
7379     %\__stex_proof_end_list:
7380   }
7381 }
7382 \stex_deactivate_macro:Nn \subproof {sproof~environments}
7383
7384 \cs_new_protected:Nn \__stex_proof_inblock_restore: {
7385   \bool_if:NT \l__stex_proof_in_spfblock_bool {
7386     \group_begin:\stexcommentfont
7387   }
7388 }

\spfstep
\conclude
\assumption
  \have
  \eqstep
7389 \stex_keys_define:nnnn { spfsteps } {
7390   \clist_clear:N \l_stex_key_for_clist
7391   \str_clear:N \l_stex_key_name_str
7392   \tl_clear:N \l_stex_key_method_tl
7393   \tl_clear:N \l_stex_key_term_tl
7394 }{
7395   for      .clist_set:N = \l_stex_key_for_clist ,
7396   method   .tl_set:N    = \l_stex_key_method_tl,
7397   term     .tl_set:N    = \l_stex_key_term_tl,
7398   name     .str_set_x:N = \l_stex_key_name_str
7399   % todo: style=inline
7400 }{id,style,title}
7401
7402
7403 \newenvironment{spfstepenv}{
7404   \str_set_eq:NN \spfstepenvautorefname \spfstepautorefname
7405 }{}
7406
7407 \cs_new_protected:Nn \__stex_proof_step_html:nn {
7408   \stex_if_do_html:TF{
7409     \exp_args:Ne \stex_annotate:nn{
7410       shtml:spf#1={
7411         \seq_if_empty:NF \l_stex_fors_seq {
7412           \seq_use:Nn \l_stex_fors_seq ,
7413         }
7414       }
7415     }

```

```

7415     \str_if_empty:NF \l_stex_key_name_str {,
7416       shtml:stepname={\l_stex_key_name_str}
7417     }
7418   }{
7419     \__stex_proof_html:
7420     #2
7421   }
7422 }{ #2 }
7423 }
7424
7425 \cs_new_protected:Nn \__stex_proof_make_step_macro:Nnnnn {
7426   \NewDocumentCommand #1 {s O{} +m} {
7427     \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7428     \stex_keys_set:nn{spfsteps}{##2}
7429     \str_if_empty:NF \l_stex_key_name_str {
7430       \stex_debug:nn{Here:}{\Variable~\l_stex_key_name_str}
7431       \exp_args:Nne\use:nn{\vardef}{\v\l_stex_key_name_str}[name=\l_stex_key_name_str]{\l_st
7432     }
7433
7434     \begin{spfstepenv}
7435       \str_if_empty:NF \l_stex_key_id_str {
7436         \__stex_proof_number_as_string:N \@currentlabel
7437         \str_set:Nx \@currentHref{spfstep.\@currentlabel}
7438         \stex_do_id:
7439       }
7440
7441       \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7442         \IfBooleanTF ##1 {
7443           \__stex_proof_step_html:nn{#2}{##3}
7444         }{
7445           \__stex_proof_step_html:nn{#2}{\__stex_proof_start_list:n{#3} ##3 \__stex_proof_end
7446             #5
7447         }
7448         \end{spfstepenv}
7449         \group_begin:\stexcommentfont
7450       }{
7451         \__stex_proof_step_html:nn{#2}{##3}
7452         \end{spfstepenv}
7453       }
7454     }
7455     \stex_deactivate_macro:Nn #1 {sproof~environments}
7456   }
7457
7458   \__stex_proof_make_step_macro:Nnnnn \assumption {assumption} \__stex_proof_insert_number: {}
7459   \__stex_proof_make_step_macro:Nnnnn \conclude {conclusion} {\$\Rightarrow$} {} {}
7460   \__stex_proof_make_step_macro:Nnnnn \spfstep {step} \__stex_proof_insert_number: {} \__stex_
7461
7462   \NewDocumentCommand \eqstep {s m}{
7463     \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7464       \group_end:
7465       \IfBooleanTF #1 {
7466         \__stex_proof_step_html:nn{eqstep}{\$= #2\$}
7467       }{
7468         \__stex_proof_step_html:nn{eqstep}{\__stex_proof_start_list:n{\$=\$} \$#2\$ \__stex_proof_

```

```

7469     }
7470     \group_begin:\stexcommentfont
7471   }{
7472     \__stex_proof_step_html:nn{eqstep}{$= #2$}
7473   }
7474 }
7475 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
7476
7477 \NewDocumentCommand \yield {+m}{
7478   \stex_annotate:nn{shtml:proofterm={}}{ #1 }
7479 }
7480 \stex_deactivate_macro:Nn \yield {sproof~environments}
7481
7482 \NewDocumentEnvironment{spfblock}{}{
7483   \bool_set_false:N \l__stex_proof_in_spfblock_bool
7484 }{
7485   \aftergroup\__stex_proof_inblock_restore:
7486 }
7487 \stex_deactivate_macro:Nn \spfblock {sproof~environments}
7488 \AddToHook{env/spfblock/before}{
7489   \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7490 }
7491
7492
7493 \newcommand\spfjust[1]{
7494   \stex_annotate:nn{spfjust={}}{ #1 }
7495 }
7496 \stex_deactivate_macro:Nn \spfjust {sproof~environments}

```

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

## 13.12 Metatheory

```

7497 <@@=stex_meta>
7498 \group_begin:
7499   \cs_set:Npn \__stex_modules_persist_module: {}
7500   \cs_set:Npn \stex_check_term:n #1 {}
7501   \cs_set:Npn \stex_sref_do_aux:n #1 { #1 }
7502   \bool_set_false:N \stex_html_do_output_bool
7503   \bool_set_false:N \c_stex_check_terms_bool
7504   \stex_uri_resolve:Nn \l_stex_current_ns_uri {http://mathhub.info/sTeX/meta}
7505   \stex_module_setup:n{Metatheory}
7506
7507   \symdef{of~type}[args=ii,invisible]{#1}
7508   \notation{of~type}[colon]{#1 \mathbin{\comp{:}} #2}
7509
7510   \symdef{apply}[args=ia,prec=0;\infpref x\infpref]{#1\mathopen{\comp{}} #2 \mathclose{\comp{}}}
7511   \notation{apply}[lambda]{#1\; \argsep{#2}{\;}}
7512   \notation{apply}[infixop]{\argsep{#2}{\mathbin{#1}}}
7513   \notation{apply}[infixrel]{\argsep{#2}{\mathrel{#1}}}
7514
7515   % structures
7516   \symdef{module~type}[args=i,op=\mathtt{MOD}]
7517   {\mathopen{\comp{\mathtt{MOD}}{}}#1\mathclose{\comp{}}}

```

```

7518 \symdef{module~type~merge}[args=a,op=\oplus]
7519   {\argsep{#1}{\mathbin{\comp{\oplus}}}}
7520 \symdef{anonymous-record}[args=a]
7521   {\mathopen{\comp{[[]]}#1\mathclose{\comp{[]}}}}
7522 \symdef{record-field}[args=2]{#1\comp{.}#2}
7523 \symdecl*{record-type}
7524
7525 \symdecl{mathstruct}[name=mathematical~structure,args=a] % TODO
7526 \notation{mathstruct}[angle,prec=nobrackets]
7527   {\mathopen{\comp{\langle} } #1 \mathclose{\comp{\rangle}}}
7528 \notation{mathstruct}[parens,prec=nobrackets]
7529   {\mathopen{\comp{() } } #1 \mathclose{\comp{}}}}
7530
7531 % sequences
7532 \symdef{ellipses}[ldots]{\ldots}
7533 \symdef{sequence-expression}[comma,args=a]{#1}
7534 \symdef{sequence-type}[args=1]{#1^{\comp{\ast}}}
7535 \symdef{sequence-map}[args=ia]{
7536   \comp{\mathrm{map}}\mathopen{\comp{() } }#1\mathpunct{\comp{,}}
7537   #2\mathclose{\comp{()}}
7538 }
7539 \iffalse
7540 % binder (\forall, \Pi, \lambda etc.)
7541 \symdef{pibind}[name=dependent function type,prec=nobrackets,
7542   op=(\cdot)\;\to\;\cdot,args=Bi,assoc=pre]
7543   {\argmap{#1}{
7544     \mathopen{\comp{() } }##1 \mathclose{\comp{()}}
7545     }{\mathbin{\comp{\to}}}\ \mathbin{\comp{\to}} #2}
7546 \notation{pibind}[forall]{\comp{\forall} #1\mathpunct{\comp{.}} #2}
7547 \notation{pibind}[Pi]{\mathop{\comp{\prod}}\c_math_subscript_token{#1}#2}
7548
7549 \symdef{mapbind}[name=lambda,mapsto,prec=nobrackets,op=\mapsto,args=Bi,assoc=pre]
7550   { #1 \mathrel{\comp{\mapsto}} #2}
7551 \notation{mapbind}[lambda,prec=nobrackets,op=\lambda]
7552   {\comp{\lambda} #1 \mathpunct{\comp{.}} #2}
7553 \fi
7554 \symdecl{bind}[args=Bi,assoc=pre]
7555 \notation{bind}[depfun,prec=nobrackets,op=(\cdot)\;\to\;\cdot]
7556   {\mathopen{\comp{() } } #1 \mathclose{\comp{()}}\mathbin{\comp{\to}} #2}
7557 \notation{bind}[forall]{\comp{\forall} #1.\;#2}
7558 \notation{bind}[Pi]{\mathop{\comp{\prod}}\c_math_subscript_token{#1}#2}
7559
7560 \symdef{implicit-bind}[args=Bi,assoc=pre]{\mathopen{\comp{\{} } } #1 \mathclose{\comp{\}}\c_math
7561
7562 \symdecl*{integer~literal}
7563 \notation{integer~literal}{\mathbb Z}
7564
7565 \symdecl*{ordinal}
7566 \notation{ordinal}{\mathtt{Ord}}
7567
7568 % propositions
7569 \symdef{prop}[name=proposition]{\mathtt{Prop}}
7570 \symdef{judgment-holds}[args=i,role=judgment]{\comp{\vdash};#1}
7571

```

```

7572 % any object
7573 \symdef{object}{\mathtt{Obj}}
7574
7575 % TODO DELETE
7576 \symdef{aseqdots}[args=a,prec=nobrackets]
7577   {#1\comp{,\ldots}}%{##1\comp,##2}
7578 \symdef{aseqfromto}[args=ai,prec=nobrackets]
7579   {#1\comp{,\ldots,}#2}%{##1\comp,##2}
7580 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]
7581   {#1\comp{,\ldots,}#2\comp{,\ldots,}#3}%{##1\comp,##2}
7582
7583
7584 \stex_close_module:
7585 \stex_uri_add_module:Nnn \l_stex_metatheory_uri \l_stex_current_ns_uri {Metatheory}
7586 \global \let \l_stex_metatheory_uri \l_stex_metatheory_uri
7587 \global \let \c_stex_default_metatheory \l_stex_metatheory_uri
7588 \group_end:

```

### 13.13 MMT Interfaces

```

7589 <@=@todo>
7590 \cs_new_protected:Npn \MSC #1 {}

```

\MMTinclude

```

7591 \stex_new_stylable_cmd:nnnn{MMTinclude}{m}{
7592   \stex_annotate_invisible:nn{shtml:import={#1}}{-}
7593 }{-}
7594 \stex_deactivate_macro:Nn \MMTinclude {module~environments}
7595 \stex_every_module:n {\stex_reactivate_macro:N \MMTinclude}

```

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

\MMTrule

```

7596 \NewDocumentCommand \MMTrule {m m}{
7597   \tl_if_empty:nTF{#2}{\seq_clear:N \l_tmpa_seq}{
7598     \seq_set_split:Nnn \l_tmpa_seq , {#2}
7599   }
7600   \int_zero:N \l_tmpa_int
7601   \stex_annotate_invisible:n{
7602     $
7603     \stex_annotate:nn{shtml:rule={scala://#1}}{
7604       \stex_annotate_force_break:n{
7605         \seq_if_empty:NF \l_tmpa_seq {
7606           \seq_map_inline:Nn \l_tmpa_seq {
7607             \int_incr:N \l_tmpa_int
7608             \stex_annotate:nn{
7609               shtml:argmode=i,
7610               shtml:arg={\int_use:N \l_tmpa_int}
7611             }{ ##1 }
7612           }
7613         }
7614       }
7615     }$
7616   }
7617 }

```

```

7618 \stex_deactivate_macro:Nn \MMTrule {module-environments}
7619 \stex_every_module:n{\stex_reactivate_macro:N \MMTrule}

```

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

mmtinterface (env.)

```

7620 \NewDocumentEnvironment { mmtinterface } { 0{} m m } {
7621   \_stex_module_setup_top_nosig:n { #3 }
7622   \str_set_eq:NN \l__todo_mmt_module_str \l_stex_current_module_str
7623   \str_clear:N \l_stex_current_module_str
7624   \stex_keys_set:nn { smodule }{ #1 }
7625   \stex_module_setup:n{ #2 }
7626   \str_set_eq:NN \l__todo_stex_module_str \l_stex_current_module_str
7627   \stex_debug:nn{mmt}{Interface~\l__todo_stex_module_str^^Jfor~\l__todo_mmt_module_str}
7628
7629   \stex_if_do_html:T {
7630     \exp_args:Nne \begin{stex_annotate_env} {
7631       shtml:theory={\l_stex_current_module_str},
7632       shtml:language={ \l_stex_current_language_str},
7633       shtml:signature={ }
7634       \tl_if_empty:NF \l_stex_metatheory_uri {,
7635         shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
7636       }
7637     }
7638     \stex_annotate_invisible:n{ }
7639     \stex_annotate_invisible:nn
7640     {shtml:import=\l__todo_mmt_module_str} { }
7641   }
7642   \stex_module_add_code:x{
7643     \stex_activate_module:n{ \l__todo_mmt_module_str }
7644   }
7645   \stex_module_add_morphism:nonn
7646   {}{\l__todo_mmt_module_str}{import}{ }
7647   \stex_reactivate_macro:N \mmtdef
7648   \stex_smsmode_do:
7649 }{
7650   \str_set_eq:NN \l_stex_current_module_str \l__todo_mmt_module_str
7651   \stex_close_module:
7652   \str_set_eq:NN \l_stex_current_module_str \l__todo_stex_module_str
7653   \stex_close_module:
7654   \stex_if_do_html:T { \end{stex_annotate_env} }
7655 }
7656 \stex_sms_allow_env:n{mmtinterface}

```

\mmtdef

```

7657 \NewDocumentCommand \mmtdef {m 0{} m} {
7658   \stex_keys_set:nn{symdef}{#2}
7659   \str_set:Nx \l_stex_macroname_str { #1 }
7660   \str_if_empty:NT \l_stex_key_name_str {
7661     \str_set:Nx \l_stex_key_name_str { #1 }
7662   }
7663   \stex_symdecl_do:
7664
7665   \str_set_eq:NN \l_stex_current_module_str \l__todo_mmt_module_str

```

```

7666 \cs_set_eq:NN \l__todo_old_metagroup_cd \stex_metagroup_do_in:nn
7667 \cs_set_protected:Npn \stex_metagroup_do_in:nn ##1 ##2 {##2}
7668 \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnN}{
7669   {\l_stex_macroname_str}
7670   {\l_stex_key_name_str}
7671   {\int_use:N \l_stex_get_symbol_arity_int}
7672   {\l_stex_get_symbol_args_tl}
7673   {}
7674   {}
7675   {}
7676   \stex_invoke_symbol:
7677 }
7678 \cs_set_eq:NN \stex_metagroup_do_in:nn \l__todo_old_metagroup_cd
7679 \str_set_eq:NN \l_stex_current_module_str \l__todo_stex_module_str
7680
7681 \str_set_eq:NN \l_stex_get_symbol_mod_str \l__todo_mmt_module_str
7682 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
7683 \stex_notation_parse:n{#3}
7684 \_stex_notation_check:
7685 \_stex_notation_add:
7686 \stex_if_do_html:T{
7687   \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7688 }
7689 \stex_smsmode_do:
7690 }
7691 \stex_deactivate_macro:Nn \mmtdef {mmtinterface-environments}
7692 \stex_sms_allow_escape:N \mmtdef

```

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

#### VoLL-KI Annotations

```

7693 \newcommand\precondition[2]{
7694   \str_clear:N \l_stex_get_symbol_name_str
7695   \stex_get_symbol:n{#2}
7696   \str_if_empty:NTF \l_stex_get_symbol_name_str{
7697     \errmessage{Unknown~symbol~#2}
7698   }{
7699     \str_case:nnTF {#1}{
7700       {remember}{}
7701       {understand}{}
7702       {analyze}{}
7703       {evaluate}{}
7704       {apply}{}
7705       {create}{}
7706     }{
7707       \stex_annotate_invisible:nn{
7708         shtml:preconditionsymbol={\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str},
7709         shtml:preconditiondimension={#1}
7710       }{}
7711     }{\errmessage{Unknown~cognitive~dimension~#1}}
7712   }
7713 }
7714 \newcommand\objective[2]{
7715   \str_clear:N \l_stex_get_symbol_name_str
7716   \stex_get_symbol:n{#2}

```



```

7717     \str_if_empty:NTF \l_stex_get_symbol_name_str{
7718         \errmessage{Unknown-symbol-#2}
7719     }{
7720     \str_case:nnTF {#1}{
7721 {remember}{}
7722 {understand}{}
7723 {analyze}{}
7724 {evaluate}{}
7725 {apply}{}
7726 {create}{}
7727     }{
7728     \stex_annotate_invisible:nn{
7729     shtml:objectivesymbol={\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str},
7730     shtml:objectivedimension={#1}
7731     }{}
7732 }{\errmessage{Unknown-cognitive-dimension-#1}}
7733     }
7734 }

7735 \seq_if_empty:NT \g_stex_current_file {
7736     \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
7737 }
7738 \stex_persist_read_now:
7739 \stex_every_file:
7740 \cs_new_protected:Nn \__todo_newlabel:n {
7741     \exp_args:Ne\__todo_old_newlabel:{\tl_to_str:n{#1}}
7742 }
7743 \AtBeginDocument{
7744     \iow_now:Nn \@auxout {
7745         \ExplSyntaxOn
7746         \let\__todo_old_newlabel:\newlabel
7747         \let\newlabel\__todo_newlabel:n
7748         \ExplSyntaxOff
7749     }
7750 }
7751 </package>

```

# Chapter 14

## Additional Packages

### 14.1 Implementation: The notesslides Package

#### 14.1.1 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.

```
7752 <*cls>
7753 <@@=notesslides>
7754 \ProvidesExplClass{notesslides}{2023/10/13}{3.4.0}{notesslides Class}
7755 \RequirePackage{l3keys2e}
7756
7757 \str_const:Nn \c__notesslides_class_str {article}
7758
7759 \keys_define:nn{notesslides / cls}{
7760   class .str_set_x:N = \c__notesslides_class_str,
7761   notes .bool_set:N = \c_notesslides_notes_bool ,
7762   slides .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
7763   %docopt .str_set_x:N = \c__notesslides_docopt_str,
7764   unknown .code:n     = {
7765     \PassOptionsToClass{\CurrentOption}{beamer}
7766     \PassOptionsToClass{\CurrentOption}{\c__notesslides_class_str}
7767     \PassOptionsToPackage{\CurrentOption}{notesslides}
7768     \PassOptionsToPackage{\CurrentOption}{stex}
7769   }
7770 }
7771 \ProcessKeysOptions{ notesslides / cls }
7772
7773 \RequirePackage{stex}
7774 \stex_if_html_backend:T {
7775   \bool_set_true:N\c_notesslides_notes_bool
7776 }
7777
7778 \bool_if:NTF \c_notesslides_notes_bool {
7779   \PassOptionsToPackage{notes=true}{notesslides}
7780   \message{notesslides.cls:~Formatting~document~in~notes~mode}
```

```

7781 }{
7782   \PassOptionsToPackage{notes=false}{notesslides}
7783   \message{notesslides.cls:~Formatting-document-in~slides-mode}
7784 }
7785
7786 \bool_if:NTF \c_notesslides_notes_bool {
7787   \LoadClass{\c_notesslides_class_str}
7788 }{
7789   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7790   %\newcounter{Item}
7791   %\newcounter{paragraph}
7792   %\newcounter{subparagraph}
7793   %\newcounter{Hfootnote}
7794 }
7795 \RequirePackage{notesslides}
7796 </cls>

```

now we do the same for the notesslides package.

```

7797 <*package>
7798 \ProvidesExplPackage{notesslides}{2023/10/13}{3.4.0}{notesslides Package}
7799 \RequirePackage{!3keys2e}
7800
7801 \keys_define:nn{notesslides / pkg}{
7802   notes           .bool_set:N = \c_notesslides_notes_bool ,
7803   slides          .code:n     = { \bool_set_false:N \c_notesslides_notes_bool },
7804   sectocframes   .bool_set:N = \c_notesslides_sectocframes_bool ,
7805   topsect        .str_set_x:N = \c_notesslides_topsect_str,
7806   unknown        .code:n     = {
7807     \PassOptionsToPackage{\CurrentOption}{stex}
7808     \PassOptionsToPackage{\CurrentOption}{tikzinput}
7809   }
7810 }
7811 \ProcessKeysOptions{ notesslides / pkg }
7812
7813 \RequirePackage{stex}
7814 \stex_if_html_backend:T {
7815   \bool_set_true:N \c_notesslides_notes_bool
7816 }
7817
7818 \cs_set:Npn \sectiontitleemph #1 {
7819   \textbf{\Large #1}
7820 }
7821
7822 \newif\ifnotes
7823 \bool_if:NTF \c_notesslides_notes_bool {
7824   \notesttrue
7825   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7826   \RequirePackage[noamsthm,hyperref]{beamerarticle}
7827   \RequirePackage{mdframed}
7828   \str_if_empty:NTF \c_notesslides_topsect_str{
7829     %\setsectionlevel{section}
7830   } {
7831     \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7832   }

```

```

7833 }{
7834 \notesfalse
7835
7836 \cs_new_protected:Nn \__notesslides_do_sectocframes: {
7837 \cs_set_protected:Nn \__notesslides_do_label:n {
7838 \str_case:nnF{##1}{
7839 {part} {
7840 \tl_set:Nx\l__notesslides_num{\thepart}
7841 \tl_set:cx{@ @ label}{
7842 \cs_if_exist:NTF\parttitlename{\exp_not:N\parttitlename}{\exp_not:N\partname}{~}
7843 }
7844 {chapter} {
7845 \tl_set:Nx\l__notesslides_num{\thechapter}
7846 \tl_set:cx{@ @ label}{
7847 \cs_if_exist:NTF\chaptertitlename{\exp_not:N\chaptertitlename}{\exp_not:N\chaptername}{~}
7848 }
7849 {section} {
7850 \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection}
7851 \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7852 }
7853 {subsection} {
7854 \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.}
7855 \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7856 }
7857 {subsubsection} {
7858 \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.}
7859 \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7860 }
7861 {paragraph} {
7862 \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.}
7863 \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7864 }
7865 }{
7866 \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.\thechapter}
7867 \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7868 }
7869 }
7870 \cs_set_protected:Nn \_sfragment_do_level:nn {
7871 \tl_if_exist:cT{c@##1}{\stepcounter{##1}}
7872 \addcontentsline{toc}{##1}{\protect\numberline{\use:c{the##1}}##2}
7873 \__notesslides_do_label:n{##1}
7874 \pdfbookmark[\int_use:N \l_stex_doheader_sect]{\l__notesslides_num\ ##2}{##1.\l__notesslides_num}
7875 \begin{frame}[noframenumbering]
7876 \vfill\centering
7877 \sectiontitleemph{
7878 \use:c{@ @ label} ##2
7879 }
7880 \end{frame}
7881 \int_incr:N \l_stex_doheader_sect
7882 \tl_set:Nn \stex_current_section_level{##1}
7883 }
7884 }
7885
7886 \AtBeginDocument{

```

```

7887 \str_if_empty:NTF \c_notesslides_topsect_str {
7888   \setsectionlevel{section}
7889 } {
7890   \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7891   \exp_args:No \str_if_eq:nnTF \c_notesslides_topsect_str {chapter} {
7892     \__notesslides_define_chapter:
7893   }{
7894     \exp_args:No \str_if_eq:nnT \c_notesslides_topsect_str {part} {
7895       \__notesslides_define_chapter:
7896       \__notesslides_define_part:
7897     }
7898   }
7899 }
7900 }
7901
7902 \bool_if:NT \c_notesslides_sectocframes_bool {
7903   \__notesslides_do_sectocframes:
7904 }
7905 }
7906
7907 \cs_new_protected:Nn \__notesslides_define_chapter: {
7908   \cs_if_exist:NF \chaptername {
7909     \cs_set_protected:Npn \chaptername {Chapter}
7910   }
7911   \cs_if_exist:NF \chapter {
7912     \cs_set_protected:Npn \chapter {INVALID}
7913   }
7914   \cs_if_exist:NF \c@chapter {
7915     \newcounter{chapter}\counterwithin*{section}{chapter}
7916   }
7917 }
7918
7919 \cs_new_protected:Nn \__notesslides_define_part: {
7920   \cs_if_exist:NF \partname {
7921     \cs_set_protected:Npn \partname {Part}
7922   }
7923   \cs_if_exist:NF \part {
7924     \cs_set_protected:Npn \part {INVALID}
7925   }
7926   \cs_if_exist:NF \c@part {
7927     \newcounter{part}\counterwithin*{chapter}{part}
7928   }
7929 }

```

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

```

7930 \def \c__notesslides_document_str{document}
7931 \newcommand\afterprematuarestop{}
7932 \def\prematuarestop@endsfragment{
7933   \unless\ifx\@currenvir\c__notesslides_document_str
7934     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7935     \expandafter\prematuarestop@endsfragment
7936   \fi
7937 }

```

```

7938 \providecommand\prematuarestop{
7939   \stex_if_html_backend:F{
7940     \message{Stopping~sTeX~processing~prematurely}
7941     \prematuarestop@endsfragment
7942     \afterprematuarestop
7943     \end{document}
7944   }
7945 }

```

(End of definition for `\prematuarestop`. This function is documented on page 108.)

## 14.1.2 Notes and Slides

For the notes case, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class.

```

7946 \bool_if:NT \c_notesslides_notes_bool {
7947   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamertheme#2}}
7948 }
7949 \NewDocumentCommand \libusetheme {0{} m} {
7950   \libusepackage[#1]{beamertheme#2}
7951 }

```

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

```

7952 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7953 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

We first set up the slide boxes in notes mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```

7954 \ifnotes
7955
7956 \newlength{\slideframewidth}
7957 \setlength{\slideframewidth}{1.5pt}

```

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

```

7958 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7959   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7960     \bool_set_true:N #1
7961   }{
7962     \bool_set_false:N #1
7963   }
7964 }
7965
7966 \stex_keys_define:nmmm{notesslides / frame}{
7967   \str_clear:N \l__notesslides_frame_label_str
7968   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7969   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7970   \bool_set_true:N \l__notesslides_frame_fragile_bool
7971   \bool_set_true:N \l__notesslides_frame_shrink_bool
7972   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7973   \bool_set_true:N \l__notesslides_frame_t_bool
7974 }{
7975   label .str_set_x:N = \l__notesslides_frame_label_str,
7976   allowframebreaks .code:n = {

```

```

7977   \_notesslides_do_yes_param:Nn \l_notesslides_frame_allowframebreaks_bool { #1 }
7978 },
7979 allowdisplaybreaks .code:n      = {
7980   \_notesslides_do_yes_param:Nn \l_notesslides_frame_allowdisplaybreaks_bool { #1 }
7981 },
7982 fragile .code:n      = {
7983   \_notesslides_do_yes_param:Nn \l_notesslides_frame_fragile_bool { #1 }
7984 },
7985 shrink .code:n      = {
7986   \_notesslides_do_yes_param:Nn \l_notesslides_frame_shrink_bool { #1 }
7987 },
7988 squeeze .code:n     = {
7989   \_notesslides_do_yes_param:Nn \l_notesslides_frame_squeeze_bool { #1 }
7990 },
7991 t .code:n           = {
7992   \_notesslides_do_yes_param:Nn \l_notesslides_frame_t_bool { #1 }
7993 },
7994 unknown .code:n    = {}
7995 }{}

```

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

```

7996 \cs_new_protected:Nn \_notesslides_setup_itemize: {
7997   \def\itemize@level{outer}
7998   \def\itemize@outer{outer}
7999   \def\itemize@inner{inner}
8000   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
8001   \renewenvironment{itemize}{
8002     \ifx\itemize@level\itemize@outer
8003       \def\itemize@label{\$ \rhd \$}
8004     \fi
8005     \ifx\itemize@level\itemize@inner
8006       \def\itemize@label{\$ \scriptstyle \rhd \$}
8007     \fi
8008     \begin{list}
8009       {\itemize@label}
8010       {\setlength{\labelsep}{.3em}
8011        \setlength{\labelwidth}{.5em}
8012        \setlength{\leftmargin}{1.5em}
8013       }
8014     \edef\itemize@level{\itemize@inner}
8015   }{
8016     \end{list}
8017   }
8018 }

```

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

```

8019 \stex_if_html_backend:TF {
8020   \cs_new_protected:Nn \_notesslides_frame_box_begin: {
8021     \vbox\bgroup
8022     \begin{stex_annotate_env}{shtml:frame={}}
8023       \mdf@patchamsthm\notesslidesfont
8024     }
8025   \cs_new_protected:Nn \_notesslides_frame_box_end: {
8026     %^A \notesslides@slidelabel

```

```

8027     \medskip\par\noindent\tiny\notesslidesfooter
8028     \end{stex_annotate_env}\egroup
8029   }
8030 }{
8031   \cs_new_protected:Nn \__notesslides_frame_box_begin: {
8032     \begin{mdframed}[
8033       linewidth=\slideframewidth,
8034       skipabove=1ex,
8035       skipbelow=1ex,
8036       userdefinedwidth=\slidewidth,
8037       align=center
8038     ]\notesslidesfont
8039   }
8040   \cs_new_protected:Nn \__notesslides_frame_box_end: {
8041     \medskip\par\noindent\tiny\notesslidesfooter%^A\notesslides@slidelabel
8042     \end{mdframed}
8043   }
8044 }

```

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

```

8045 \renewenvironment{frame}[1][]{
8046   \stex_keys_set:nn{notesslides / frame}{#1}
8047   \stepcounter{framenum}
8048   \renewcommand\newpage{\addtocounter{framenum}{1}}
8049   \def@currentlabel{\theframenum}
8050   \str_if_empty:NF \l__notesslides_frame_label_str {
8051     \label{\l__notesslides_frame_label_str}
8052   }
8053   \__notesslides_setup_itemize:
8054   \__notesslides_frame_box_begin:
8055 }{
8056   \__notesslides_frame_box_end:
8057 }

```

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

`\frametitle`

```

8058 \renewcommand{\frametitle}[1]{
8059   \stexdoctitle { #1 }
8060   \notesslidestitleemph{#1}\medskip
8061 }

```

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

`\pause`

```

8062 \newcommand\pause{}

```

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

We redefine the columns and column environments:

```

8063 \renewenvironment{columns}[1][]{
8064   \par\noindent
8065   \begin{minipage}
8066   \slidewidth\centering\leavevmode
8067 % \stex_if_html_backend:T{

```



```

8068 % \cs_if_exist:NT \rustex_if:T {
8069 %   \rustex_if:T {\par
8070 %     \rustex_direct_HTML:n{<table><tr><td>}
8071 %   }
8072 % }
8073 % }
8074 }{
8075 % \stex_if_html_backend:T{
8076 %   \cs_if_exist:NT \rustex_if:T {
8077 %     \rustex_if:T {\par
8078 %       \rustex_direct_HTML:n{</td></tr></table>}
8079 %     }
8080 %   }
8081 % }
8082 \end{minipage}\par\noindent
8083 }
8084 \newsavebox\columnbox
8085 \renewenvironment<>{column}[2][ ]{
8086 \begin{lrbox}{\columnbox}
8087 % \stex_if_html_backend:T{
8088 %   \cs_if_exist:NT \rustex_if:T {
8089 %     \rustex_if:T {\par
8090 %       \rustex_direct_HTML:n{</td><td>}
8091 %     }
8092 %   }
8093 % }
8094 \begin{minipage}{#2}
8095 }{
8096 \end{minipage}
8097 % \stex_if_html_backend:T{
8098 %   \cs_if_exist:NT \rustex_if:T {
8099 %     \rustex_if:T {\par
8100 %       \rustex_direct_HTML:n{</td><td>}
8101 %     }
8102 %   }
8103 % }
8104 \end{lrbox}\usebox\columnbox
8105 }
8106 \fi

```

### 14.1.3 Environment and Macro Patches

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 to produce no output.

```

8107 \bool_if:NTF \c_notesslides_notes_bool {
8108 \renewenvironment{note}{\ignorespaces}{}
8109 }{
8110 \renewenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup}{\egroup}
8111 }

```

For other environments we introduce variants prefixed with `n`, which are excluded in `slides` mode.

```

8112 \cs_new_protected:Nn \__notesslides_notes_env:nmmn {
8113   \bool_if:NTF \c_notesslides_notes_bool {
8114     \newenvironment{#1}#2{#3}{#4}
8115   }{
8116     \newenvironment{#1}#2{
8117       \cs_set:Npn \__notesslides_eat: #####1 \end #####2 {
8118         \str_if_eq:nnTF{#1}{#####2}{
8119           \end{#1}
8120         }{
8121           \__notesslides_eat:
8122         }
8123       }
8124       \__notesslides_eat:
8125       %\setbox\l_tmpa_box\vbox\bgroup#3
8126     }{
8127       %#4\egroup
8128     }
8129   }
8130 }
8131
8132 \__notesslides_notes_env:nmmn{nparagraph}{[1][ ]}{\begin{sparagraph}[#1]}{\end{sparagraph}}
8133 \__notesslides_notes_env:nmmn{nfragment}{[2][ ]}{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
8134 \__notesslides_notes_env:nmmn{ndefinition}{[1][ ]}{\begin{sdefinition}[#1]}{\end{sdefinition}}
8135 \__notesslides_notes_env:nmmn{nassertion}{[1][ ]}{\begin{sassertion}[#1]}{\end{sassertion}}
8136 \__notesslides_notes_env:nmmn{nproof}{[2][ ]}{\begin{sproof}[#1]{#2}}{\end{sproof}}
8137 \__notesslides_notes_env:nmmn{nexample}{[1][ ]}{\begin{sexample}[#1]}{\end{sexample}}
8138
8139 \RequirePackage{graphicx}
8140
8141 \NewDocumentCommand\frameimage{s O{} m}{
8142   \IfBooleanTF #1 {
8143     \begin{frame}[plain]
8144   }{
8145     \begin{frame}
8146   }
8147   \bool_if:NTF \c_notesslides_notes_bool {
8148     \slidewidth=\dimexpr\slidewidth-(2\slideframewidth)\relax
8149   }{
8150     \slidewidth=\textwidth\relax
8151   }
8152   \def\Gin@ewidth{}\setkeys{Gin}{#2}
8153   \tl_if_empty:NTF \Gin@ewidth {
8154     \mhgraphics[width=\slidewidth,#2]{#3}
8155   }{
8156     \mhgraphics[#2]{#3}
8157   }
8158   \end{frame}
8159 }

```

hacking inputref:

```

\inputref*
8160 \cs_set_eq:NN\__notesslides_inputref:\inputref
8161 \cs_set_protected:Npn\inputref{\@ifstar\ninputref\__notesslides_inputref:}

```

```

8162 \bool_if:NTF \c_notesslides_notes_bool {
8163   \newcommand\inputref [2] [] {
8164     \_notesslides_inputref: [#1] {#2}
8165   }
8166 }{
8167   \newcommand\inputref [2] [] {}
8168 }

```

(End of definition for `\inputref*`. This function is documented on page 107.)

#### 14.1.4 Styling Across Notes/Slides

```

8169 \def\notesslidestitleemph#1{
8170   {\Large\bf\sf#1}
8171   \vskip0.1\baselineskip
8172   \leaders\vrule width \textwidth
8173   \vskip0.4pt%
8174   \nointerlineskip
8175 }
8176
8177 \def\notesslidesfooter{}
8178
8179 \let\notesslidesfont\sffamily

```

#### 14.1.5 Beamer Compatibility

All of this should be removed and made part of a template

```

8180
8181 \bool_if:NT \c_notesslides_notes_bool {
8182   \def\author{\@dblarg\ns@author}
8183   \long\def\ns@author [#1] #2{%
8184     \tl_if_empty:nTF{#1}{
8185       \def\beamer@shortauthor{#2}
8186     }{
8187       \def\beamer@shortauthor{#1}
8188     }
8189     \def\@author{#2}
8190   }
8191   \def\title{\@dblarg\ns@title}
8192   \long\def\ns@title [#1] #2{%
8193     \tl_if_empty:nTF{#1}{
8194       \def\beamer@shorttitle{#2}
8195     }{
8196       \def\beamer@shorttitle{#1}
8197     }
8198     \def\@title{#2}
8199     \stexdoctitle{#2}
8200   }
8201   \def\insertshortauthor{
8202     \hbox\bgroup\def\{\}\cs_if_exist:NT\beamer@shortauthor\beamer@shortauthor\egroup
8203   }
8204   \def\insertshorttitle{
8205     \hbox\bgroup\def\{\}\cs_if_exist:NT\beamer@shorttitle\beamer@shorttitle\egroup
8206   }
8207   \stex_if_html_backend:TF{

```

```

8208     \def\insertframenumbers{\stex_annotate:nn{shtml:framenumbers={}}{}}
8209   }{
8210     \def\insertframenumbers{\@arabic\c@framenumbers}
8211   }
8212   \def\insertshortdate{\today}
8213 }

```

### 14.1.6 TODO 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.

```

8214 \gdef\printexcursions{}
8215 \newcommand\excursionref[2]{% label, text
8216   \bool_if:NT \c_notesslides_notes_bool {
8217     \begin{sparagraph}[title=Excursion]
8218       #2 \sref[fallback=the appendix]{#1}.
8219     \end{sparagraph}
8220   }
8221 }
8222 \newcommand\activate@excursion[2][]{
8223   \tl_gput_right:Nn\printexcursions{\inputref[#1]{#2}}
8224 }
8225 \newcommand\excursion[4][]{% repos, label, path, text
8226   \bool_if:NT \c_notesslides_notes_bool {
8227     \activate@excursion[#1]{#3}
8228     \excursionref{#2}{#4}
8229   }
8230 }

```

(End of definition for `\excursion`. This function is documented on page 109.)

### **\excursiongroup**

```

8231 \keys_define:nn{notesslides / excursiongroup }{
8232   id      .str_set_x:N = \l__notesslides_excursion_id_str,
8233   intro   .tl_set:N    = \l__notesslides_excursion_intro_tl,
8234   archive .str_set_x:N = \l__notesslides_excursion_mhrepos_str
8235 }
8236 \cs_new_protected:Nn \__notesslides_excursion_args:n {
8237   \tl_clear:N \l__notesslides_excursion_intro_tl
8238   \str_clear:N \l__notesslides_excursion_id_str
8239   \str_clear:N \l__notesslides_excursion_mhrepos_str
8240   \keys_set:nn {notesslides / excursiongroup }{ #1 }
8241 }
8242 \newcommand\excursiongroup[1][]{
8243   \__notesslides_excursion_args:n{ #1 }
8244   \tl_if_empty:NF\printexcursions
8245   {\IfInputref}{\begin{note}
8246     \begin{sfragment}{Excursions}% TODO pass on id
8247     \ifdefempty\l__notesslides_excursion_intro_tl{
8248       \exp_args:NNe \use:nn \inputref{[\l__notesslides_excursion_mhrepos_str]{
8249         \l__notesslides_excursion_intro_tl
8250       }}
8251     }

```

```

8252     \printexcursions%
8253     \end{sfragment}
8254     \end{note}}}}
8255 }
8256 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi

(End of definition for \excursiongroup. This function is documented on page 109.)

8257 \prop_new:N \g__notesslides_variables_prop
8258 \cs_set_protected:Npn \setSGvar #1 #2 {
8259   \prop_gput:Nnn \g__notesslides_variables_prop {#1}{#2}
8260 }
8261 \cs_set_protected:Npn \useSGvar #1 {
8262   \prop_item:Nn \g__notesslides_variables_prop {#1}
8263 }
8264 \cs_set_protected:Npn \ifSGvar #1 #2 #3 {
8265   \prop_get:NnNF \g__notesslides_variables_prop {#1} \l__notesslides_tmp {
8266     \PackageError{document-structure}
8267     {The sTeX Global variable #1 is undefined}
8268     {set it with \protect\setSGvar}\TODO better error
8269   }
8270   \tl_if_eq:NnT \l__notesslides_tmp {#2}{ #3 }
8271 }
8272
8273
8274 \end{package}

```

## 14.2 Implementation: The problem Package

### 14.2.1 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.

```

8275 <*package>
8276 <@@=problems>
8277 \ProvidesExplPackage{problem}{2023/10/13}{3.4.0}{Semantic Markup for Problems}
8278 \RequirePackage{l3keys2e}
8279
8280 \keys_define:nn { problem / pkg }{
8281   notes      .default:n = { true },
8282   notes      .bool_set:N = \c__problems_notes_bool,
8283   gnotes     .default:n = { true },
8284   gnotes     .bool_set:N = \c__problems_gnotes_bool,
8285   hints      .default:n = { true },
8286   hints      .bool_set:N = \c__problems_hints_bool,
8287   solutions  .default:n = { true },
8288   solutions  .bool_set:N = \c__problems_solutions_bool,
8289   pts        .default:n = { true },
8290   pts        .bool_set:N = \c__problems_pts_bool,
8291   min        .default:n = { true },
8292   min        .bool_set:N = \c__problems_min_bool,
8293   %boxed     .default:n = { true },
8294   %boxed     .bool_set:N = \c__problems_boxed_bool,

```

```

8295 test .default:n = { true },
8296 test .bool_set:N = \c_problems_test_bool,
8297 unknown .code:n = {
8298   \PassOptionsToPackage{\CurrentOption}{stex}
8299 }
8300 }
8301 \newif\ifsolutions
8302
8303 \ProcessKeysOptions{ problem / pkg }
8304 \bool_if:NTF \c_problems_solutions_bool {
8305   \solutionstrue
8306 }{
8307   \solutionsfalse
8308 }
8309 \newif\ifintest
8310 \bool_if:NTF \c_problems_test_bool {
8311   \intesttrue
8312 }{
8313   \intestfalse
8314 }
8315
8316 \RequirePackage{stex}

```

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

```

8317 \AddToHook{begindocument}{
8318   \ExplSyntaxOn\makeatletter
8319   \input{problem-english.ldf}
8320   \ltx@ifpackageloaded{babel}{
8321     \clist_set:Nx \l_tmpa_clist { \exp_args:No \tl_to_str:n \bbl@loaded }
8322     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8323       \input{problem-ngerman.ldf}
8324     }
8325     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8326       \input{problem-finnish.ldf}
8327     }
8328     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8329       \input{problem-french.ldf}
8330     }
8331     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8332       \input{problem-russian.ldf}
8333     }
8334   }{ }
8335   \makeatother\ExplSyntaxOff
8336 }

```

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

## 14.2.2 Problems and Solutions

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

```

8337 \bool_new:N \l_stex_key_autogradable_bool

```

```

8338 \stex_keys_define:nnnn{ problem }{
8339   \tl_set:Nn \l_stex_key_pts_tl 0
8340   \tl_set:Nn \l_stex_key_min_tl 0
8341   \str_clear:N \l_stex_key_name_str
8342   \str_clear:N \l_stex_key_mhrepos_str
8343   \bool_set_false:N \l_stex_key_autogradable_bool
8344 }{
8345   pts      .tl_set:N      = \l_stex_key_pts_tl,
8346   min      .tl_set:N      = \l_stex_key_min_tl,
8347   name     .str_set:N     = \l_stex_key_name_str,
8348   autogradable .bool_set:N = \l_stex_key_autogradable_bool,
8349   archive  .code:n = {},
8350   %archive .str_set:N     = \l_stex_key_mhrepos_str,
8351   creators .code:n = {}
8352   %imports .tl_set:N     = \l__problems_prob_imports_tl,
8353   %refnum  .int_set:N    = \l__problems_prob_refnum_int,
8354 }{id,title,style,uses}

```

Then we set up a counter for problems.

`\numberproblemsin`

```

8355 \newcounter{sproblem}[section]
8356 \newcommand\numberproblemsin[1]{
8357   \@addtoreset{sproblem}{#1}
8358   \def\thesproblem{\arabic{#1}.\arabic{sproblem}}
8359 }
8360 \numberproblemsin{section}
8361 %\def\theplainsproblem{\arabic{sproblem}}
8362 %\def\thesproblem{\thesection.\theplainsproblem}

```

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

`sproblem (env.)`

```

8363 \cs_new:Nn \__problems_activate_macros: {
8364   \stex_reactivate_macro:N \solution
8365   \stex_reactivate_macro:N \mcb
8366   \stex_reactivate_macro:N \scb
8367   \stex_reactivate_macro:N \fillinsol
8368   \stex_reactivate_macro:N \hint
8369   \stex_reactivate_macro:N \exnote
8370   \stex_reactivate_macro:N \gnote
8371 }
8372
8373 \newcounter{pts}
8374 \newcounter{min}
8375 \bool_new:N \l__problems_in_problem_bool
8376 \bool_new:N \l__problems_has_pts_bool
8377 \bool_new:N \l__problems_has_min_bool
8378 \bool_set_false:N \l__problems_in_problem_bool
8379 \stex_new_stylable_env:nnnnnn {problem} {0}{}{
8380   \bool_if:NT \l__problems_in_problem_bool {
8381     \msg_error:nn{stex}{error/nestedproblem}
8382   }
8383   \cs_if_exist:NTF \l_problem_inputproblem_keys_tl {
8384     \tl_put_left:Nn \l_problem_inputproblem_keys_tl {#1,}

```

```

8385     \exp_args:Nno \stex_keys_set:nn{problem}{
8386       \l_problem_inputproblem_keys_tl
8387     }
8388   }{
8389     \stex_keys_set:nn{problem}{#1}
8390   }
8391   \refstepcounter{sproblem}
8392
8393   \stex_if_do_html:T {
8394     \str_if_empty:NT \l_stex_key_name_str {
8395       \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8396       \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8397     }
8398     \exp_args:Nne \begin{stex_annotate_env} {
8399       shtml:problem={\l_stex_key_name_str},
8400       shtml:language={ \l_stex_current_language_str},
8401       shtml:autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
8402     }
8403     \tl_if_empty:NF \l_stex_key_title_tl {
8404       \exp_args:No \stexdoctitle \l_stex_key_title_tl
8405     }
8406     \stex_annotate_invisible:nn{
8407       shtml:problempoints={\l_stex_key_pts_tl}
8408     }{ \l_stex_key_pts_tl }
8409   }
8410
8411   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
8412
8413   \bool_set_true:N \l__problems_in_problem_bool
8414   \tl_set_eq:NN \l__problems_pts_tl \l_stex_key_pts_tl
8415   \tl_set_eq:NN \l__problems_min_tl \l_stex_key_min_tl
8416   \tl_if_eq:NnTF \l__problems_pts_tl {0}
8417     {\bool_set_false:N \l__problems_has_pts_bool}
8418     {\bool_set_true:N \l__problems_has_pts_bool}
8419   \tl_if_eq:NnTF \l__problems_min_tl {0}
8420     {\bool_set_false:N \l__problems_has_min_bool}
8421     {\bool_set_true:N \l__problems_has_min_bool}
8422   \int_gzero:N \g__problems_subproblem_int
8423
8424   \stex_style_apply:
8425   \_stex_do_id:
8426   \__problems_activate_macros:
8427   }{
8428     \addtocounter{pts}{\l__problems_pts_tl}
8429     \addtocounter{min}{\l__problems_min_tl}
8430     \__problems_record_problem:
8431     \stex_style_apply:
8432     \stex_if_do_html:T{ \end{stex_annotate_env} }
8433   }{
8434     \par\noindent\problemheader
8435     \bool_if:NT \c__problems_pts_bool {
8436       \tl_if_eq:NnF \l__problems_pts_tl {0}{
8437         \marginpar{\l__problems_pts_tl}{~\problem@kw@pts\smallskip}
8438       }

```



```

8439 }
8440 \bool_if:NT \c__problems_min_bool {
8441   \tl_if_eq:NnF \l__problems_min_tl {0} {
8442     \marginpar{\l__problems_min_tl}{~\problem@kw@minutes\smallskip}
8443   }
8444 }
8445 \par
8446 \stex_ignore_spaces_and_pars:
8447 }{
8448   \par\bigskip
8449   % \bool_if:NT \c__problems_test_bool \pagebreak
8450 }{s}
8451
8452 \tl_set:Nn \problemheader {
8453   \textbf{\sproblemautorefname{~}\thesproblem
8454     \tl_if_empty:NF \thistitle {
8455       {~}(\thistitle)
8456     }
8457   }
8458 }
8459
8460 \cs_new_protected:Nn \__problems_record_problem: {
8461   \exp_args:NNe \iow_now:Nn \@auxout {
8462     \problem@restore {\thesproblem}{\l__problems_pts_tl}{\l__problems_min_tl}
8463   }
8464 }
8465
8466 \cs_new_protected:Npn \problem@restore #1 #2 #3 {}

```

subproblem (*env.*)

```

8467 \int_new:N \g__problems_subproblem_int
8468
8469 \stex_new_stylable_env:nnnnnn {subproblem} {0}{0}{
8470   \stex_keys_set:nn{problem}{#1}
8471   \bool_if:NF \l__problems_in_problem_bool{
8472     \ifstexhtml\else
8473       \par{\bfseries WARNING~subproblem~to~be~used~in~some~problem}\par
8474     \fi
8475     \__problems_activate_macros:
8476     \bool_set_true:N \l__problems_in_problem_bool
8477     \tl_set:Nn \l__problems_pts_tl{0}
8478     \tl_set:Nn \l__problems_min_tl{0}
8479   }
8480   \str_if_empty:NT \l_stex_key_name_str {
8481     \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8482     \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8483   }
8484   \stex_if_do_html:T {
8485     \str_if_empty:NT \l_stex_key_name_str {
8486       \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8487       \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8488     }
8489     \exp_args:Nne \begin{stex_annotate_env} {
8490       shtml:subproblem={\l_stex_key_name_str},

```

```

8491     shtml:language={ \l_stex_current_language_str},
8492     shtml:autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
8493   }
8494   \stex_annotate_invisible:n{}
8495   \tl_if_empty:NF \l_stex_key_title_tl {
8496     \exp_args:No \stexdoctitle \l_stex_key_title_tl
8497   }
8498 }
8499 \int_gincr:N \g__problems_subproblem_int
8500 \bool_if:NF \l__problems_has_pts_bool {
8501   \tl_gset:Nx \l__problems_pts_tl {\int_eval:n {\l__problems_pts_tl + \l_stex_key_pts_tl}}
8502 }
8503 \bool_if:NF \l__problems_has_min_bool {
8504   \tl_gset:Nx \l__problems_min_tl {\int_eval:n {\l__problems_min_tl + \l_stex_key_min_tl}}
8505 }
8506 \stex_if_smsmode:F \stex_style_apply:
8507 }{
8508   \stex_if_smsmode:F \stex_style_apply:
8509   \stex_if_do_html:T{ \end{stex_annotate_env} }
8510 }{
8511   \begin{list}{}{
8512     \setlength\topsep{0pt}
8513     \setlength\parsep{0pt}
8514     \setlength\rightmargin{0pt}
8515   } \item[\int_use:N \g__problems_subproblem_int .]
8516   \bool_if:NT \c__problems_pts_bool {
8517     \marginpar{\smallskip\l_stex_key_pts_tl}~\problem@kw@pts}
8518   }
8519 }
8520 }
8521 \bool_if:NT \c__problems_min_bool {
8522   \bool_if:NF \l__problems_has_min_bool{
8523     \marginpar{\smallskip\l_stex_key_min_tl}~\problem@kw@minutes}
8524   }
8525 }
8526 }{
8527   \end{list}
8528 }{}

```

### **\includeproblem**

```

8529 \stex_keys_define:nnnn{ includeproblem }{
8530   \str_clear:N \l_stex_key_mhrepos_str
8531 }{
8532   archive .str_set:N      = \l_stex_key_mhrepos_str,
8533   unknown .code:n = {}
8534 }{}
8535
8536 \NewDocumentCommand\includeproblem{0{} m}{
8537   \group_begin:
8538   \tl_set:Nn \l_problem_inputproblem_keys_tl {#1}
8539   \stex_keys_set:nn{includeproblem}{#1}
8540   \exp_args:Nno \use:nn{\inputref[]\l_stex_key_mhrepos_str}{#2}
8541   \group_end:
8542 }

```

8543

(End of definition for `\includeproblem`. This function is documented on page 114.)

`solution (env.)`

```
8544 \int_new:N \g_problem_id_counter
8545 \dim_new:N \l_stex_key_testspace_dim
8546 \stex_keys_define:nnnn{ solution }{
8547   \str_clear:N \l_stex_key_answerclass_str
8548   \dim_zero:N \l_stex_key_testspace_dim
8549 }{
8550   testspace .dim_set:N = \l_stex_key_testspace_dim,
8551   answerclass .str_set:N = \l_stex_key_answerclass_str
8552 }{id,title,style}
8553
8554 \cs_new_protected:Nn \__problems_solution_start:n {
8555   \stex_keys_set:nn{ solution }{#1}
8556   \str_if_empty:NT \l_stex_key_id_str {
8557     \int_gincr:N \g_problem_id_counter
8558     \str_set:Nx \l_stex_key_id_str {
8559       SOLUTION_\int_use:N \g_problem_id_counter
8560     }
8561   }
8562   \stex_if_do_html:T{
8563     \begin{stex_annotate_env}{
8564       shtml:solution=\l_stex_key_id_str,
8565       shtml:answerclass={\l_stex_key_answerclass_str}
8566     }
8567   }
8568   \stex_style_apply:
8569 }
8570
8571 \stex_new_stylable_env:nnnnnnn { solution }{ 0{} }{
8572   \stex_if_do_html:TF{
8573     \__problems_solution_start:n{#1}
8574   }{
8575     \ifsolutions
8576       \__problems_solution_start:n{#1}
8577     \else
8578       \stex_keys_set:nn{ solution }{#1}
8579       \testspace{\l_stex_key_testspace_dim}
8580       \setbox\l_tmpa_box\vbox\bgroup
8581       \fi
8582     }
8583   }{
8584     \stex_if_do_html:TF{
8585       \stex_style_apply:
8586       \end{stex_annotate_env}
8587     }{
8588       \ifsolutions
8589         \stex_style_apply:
8590         \stex_if_do_html:T{
8591           \end{stex_annotate_env}
8592         }

```

```

8593     \else
8594     \egroup
8595     \fi
8596   }
8597 }{
8598   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8599   \noindent\emph{\problem@kw@solution\tl_if_empty:NF \l_stex_key_title_tl{
8600     {~}\l_stex_key_title_tl \str_if_empty:NF \l_stex_key_answerclass_str {
8601       (Answer Class: \l_stex_key_answerclass_str)
8602     }
8603   } :~}
8604 }{
8605   \par\rule[.3em]{\linewidth}{0.4pt}\newline
8606 }{}
8607
8608 \stex_deactivate_macro:Nn \solution {problem~environments}

```

`\startsolutions`

`\stopsolutions`

```

8609 \cs_new_protected:Npn \startsolutions{
8610   \global\solutionstrue
8611 }
8612 \cs_new_protected:Npn \stopsolutions{
8613   \global\solutionsfalse
8614 }

```

*(End of definition for \startsolutions and \stopsolutions. These functions are documented on page 111.)*

`hint (env.)`

```

8615
8616 \stex_keys_define:nmmn{ problemenv }-{}{id,title,style}
8617
8618 \cs_new_protected:Nn \__problems_hint_start:n {
8619   \stex_keys_set:nn{ problemenv }{#1}
8620   \str_if_empty:NT \l_stex_key_id_str {
8621     \int_gincr:N \g_problem_id_counter
8622     \str_set:Nx \l_stex_key_id_str {
8623       HINT_\int_use:N \g_problem_id_counter
8624     }
8625   }
8626   \stex_if_do_html:T{
8627     \begin{stex_annotate_env}{
8628       shtml:problemhint=\l_stex_key_id_str
8629     }
8630   }
8631   \stex_style_apply:
8632 }
8633
8634 \stex_new_stylable_env:nmmmmn { hint }{ 0{} }{
8635   \stex_if_do_html:TF{
8636     \__problems_hint_start:n{#1}
8637   }{
8638     \bool_if:NTF \c__problems_hints_bool {
8639       \__problems_hint_start:n{#1}

```

```

8640     }{
8641     \setbox\l_tmpa_box\vbox\bgroup
8642     }
8643   }
8644 }{
8645 \stex_if_do_html:TF{
8646   \stex_style_apply:
8647   \end{stex_annotate_env}
8648 }{
8649   \bool_if:NTF \c__problems_hints_bool {
8650     \stex_style_apply:
8651     \stex_if_do_html:T{
8652       \end{stex_annotate_env}
8653     }
8654   }{
8655     \egroup
8656   }
8657 }
8658 }{
8659 \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8660 \noindent\emph{\problem@kw@hint\tl_if_empty:NF \l_stex_key_title_tl{
8661   {~}}\l_stex_key_title_tl
8662   } :~}
8663 }{
8664 \par\rule[.3em]{\linewidth}{0.4pt}\newline
8665 }{}
8666 \stex_deactivate_macro:Nn \hint {sproblem~environments}

```

exnote (*env.*)

```

8667 \cs_new_protected:Nn \__problems_exnote_start:n {
8668   \stex_keys_set:nn{ problemenv }{#1}
8669   \str_if_empty:NT \l_stex_key_id_str {
8670     \int_gincr:N \g_problem_id_counter
8671     \str_set:Nx \l_stex_key_id_str {
8672       EXNOTE_\int_use:N \g_problem_id_counter
8673     }
8674   }
8675   \stex_if_do_html:T{
8676     \begin{stex_annotate_env}{
8677       shtml:problemnote=\l_stex_key_id_str
8678     }
8679   }
8680   \stex_style_apply:
8681 }
8682
8683 \stex_new_stylable_env:nnnnnnn { exnote }{ 0{} }{
8684   \stex_if_do_html:TF{
8685     \__problems_exnote_start:n{#1}
8686   }{
8687     \bool_if:NTF \c__problems_notes_bool {
8688       \__problems_exnote_start:n{#1}
8689     }{
8690       \setbox\l_tmpa_box\vbox\bgroup
8691     }

```

```

8692 }
8693 }{
8694   \stex_if_do_html:TF{
8695     \stex_style_apply:
8696     \end{stex_annotate_env}
8697   }{
8698     \bool_if:NTF \c__problems_notes_bool {
8699       \stex_style_apply:
8700       \stex_if_do_html:T{
8701         \end{stex_annotate_env}
8702       }
8703     }{
8704       \egroup
8705     }
8706   }
8707 }{
8708   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8709   \noindent\emph{\problem@kw@note\tl_if_empty:NF \l_stex_key_title_tl{
8710     {~}\l_stex_key_title_tl
8711   } :~}
8712 }{
8713   \par\rule[.3em]{\linewidth}{0.4pt}\newline
8714 }{}
8715 \stex_deactivate_macro:Nn \exnote {sproblem~environments}

```

*gnote (env.)*

```

8716 \int_new:N \l__problems_anscls_int
8717
8718 \cs_new_protected:Nn \__problems_gnote_start:n {
8719   \stex_keys_set:nn{ problemenv }{#1}
8720   \str_if_empty:NT \l_stex_key_id_str {
8721     \int_gincr:N \g_problem_id_counter
8722     \str_set:Nx \l_stex_key_id_str {
8723       GNOTE_\int_use:N \g_problem_id_counter
8724     }
8725   }
8726 }
8727
8728 \stex_if_do_html:T{
8729   \begin{stex_annotate_env}{
8730     shtml:problemgnote=\l_stex_key_id_str
8731   }
8732 }
8733 \stex_style_apply:
8734 }
8735
8736 \stex_new_stylable_env:nnnnnn { gnote }{ 0{} }{
8737   \stex_if_do_html:TF{
8738     \__problems_gnote_start:n{#1}
8739   }{
8740     \bool_if:NTF \c__problems_gnotes_bool {
8741       \__problems_gnote_start:n{#1}
8742     }{
8743       \setbox\l_tmpa_box\vbox\bgroup

```

```

8744 }
8745 \stex_reactivate_macro:N \anscls
8746 }{
8747   \stex_if_do_html:TF{
8748     \stex_style_apply:
8749     \end{stex_annotate_env}
8750   }{
8751     \bool_if:NTF \c__problems_gnotes_bool {
8752       \stex_style_apply:
8753       \stex_if_do_html:T{
8754         \end{stex_annotate_env}
8755       }
8756     }{
8757       \egroup
8758     }
8759   }
8760 }{
8761   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8762   \noindent\emph{\problem@kw@grading\str_if_empty:NF \l_stex_key_title_tl{
8763     {~}\l_stex_key_title_tl
8764   } :~}
8765 }{
8766   \par\rule[.3em]{\linewidth}{0.4pt}\newline
8767 }{}
8768 \stex_deactivate_macro:Nn \gnote {sproblem-environments}
8769
8770
8771 \stex_keys_define:nnnn{ anscls }{
8772   \str_clear:N \l_stex_key_pts_str
8773   \tl_clear:N \l_stex_key_feedback_tl
8774 }{
8775   pts      .str_set:N = \l_stex_key_pts_str,
8776   feedback .tl_set:N = \l_stex_key_feedback_str,
8777   update   .code:n    = {}
8778 }{id}
8779 \newcommand \anscls [2] [] {
8780   \stex_keys_set:nn{ anscls }{#1}
8781   \str_if_empty:NT \l_stex_key_id_str {
8782     \int_incr:N \l__problems_anscls_int
8783     \str_set:Nx \l_stex_key_id_str {
8784       AC\int_use:N \l__problems_anscls_int
8785     }
8786   }
8787   \begin{list}{}{
8788     \setlength\topsep{0pt}
8789     \setlength\parsep{0pt}
8790     \setlength\rightmargin{0pt}
8791   }\item[\l_stex_key_id_str]
8792     \stex_if_do_html:TF{
8793       \stex_annotate:nn{
8794         shtml:answerclass={\l_stex_key_id_str}
8795         \str_if_empty:NF \l_stex_key_pts_str{
8796           ,shtml:answerclass-pts={\l_stex_key_pts_str}
8797         }

```

```

8798     \str_if_empty:NF \l_stex_key_feedback_str{
8799       ,shtml:answerclass-feedback={\l_stex_key_feedback_str}
8800     }
8801   }{
8802     #2
8803   }
8804 }{#2}
8805 \str_if_empty:NF \l_stex_key_pts_str {\par
8806 ~ \problem@kw@points:~\l_stex_key_pts_str
8807 }
8808 \str_if_empty:NF \l_stex_key_feedback_str {\par
8809 ~ \problem@kw@feedback :~\l_stex_key_feedback_str
8810 }
8811 \end{list}
8812 }
8813 \stex_deactivate_macro:Nn \anscls {gnote~environments}

```

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

```

8814 \def\pts#1{
8815   \bool_if:NT \c__problems_pts_bool {
8816     \stex_annotate:nn{shtml:problempoints={#1}}{\marginpar{#1~\problem@kw@pts}}
8817   }\hbox_unpack:N\c_empty_box
8818 }
8819 \def\min#1{
8820   \bool_if:NT \c__problems_min_bool {
8821     \stex_annotate:nn{shtml:problemminutes={}}{\marginpar{#1~\problem@kw@minutes}}
8822   }\hbox_unpack:N\c_empty_box
8823 }

```

*mcb (env.)*

```

8824 \stex_new_stylable_env:nnnnnn{mcb}{0}{}{
8825   \stex_keys_set:nn{style}{#1}
8826   \stex_if_do_html:T{
8827     \tl_set:Nn\problem_mcc_box_tl{}
8828     \exp_args:Nne \begin{stex_annotate_env}{
8829       shtml:multiple-choice-block={}
8830       \clist_if_empty:NF \l_stex_key_style_clist {,
8831         shtml:styles={\l_stex_key_style_clist}
8832       }
8833     }
8834   }
8835   \stex_deactivate_macro:Nn \mcb {sproblem~environments}
8836   \stex_deactivate_macro:Nn \scb {sproblem~environments}
8837   \stex_deactivate_macro:Nn \solution {sproblem~environments}
8838   \stex_deactivate_macro:Nn \hint {sproblem~environments}
8839   \stex_deactivate_macro:Nn \exnote {sproblem~environments}
8840   \stex_deactivate_macro:Nn \gnote {sproblem~environments}
8841   \stex_reactivate_macro:N \mcc
8842   \cs_set:Nn \__problems_mccline:n{
8843     \begin{list}{}{
8844       \setlength\topsep{0pt}
8845       \setlength\parsep{0pt}
8846       \setlength\rightmargin{0pt}
8847     }\item[\problem_mcc_box_tl] ##1 \end{list}

```



```

8848 }
8849 \stex_style_apply:
8850 ){
8851   \stex_style_apply:
8852   \stex_if_do_html:T{
8853     \end{stex_annotate_env}
8854   }
8855 }{\par}{}{}
8856 \stexstylemcb[inline]{
8857   \cs_set:Nn \__problems_mccline:n{\problem_mcc_box_tl{~} #1}
8858 }{}
8859
8860 \stex_deactivate_macro:Nn \mcb {sproblem-environments}
      we define the keys for the mcc macro
8861 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8862   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8863     \bool_set_true:N #1
8864   }{
8865     \bool_set_false:N #1
8866   }
8867 }
8868 \stex_keys_define:nnnn{mcc}{
8869   \tl_clear:N \l_stex_key_feedback_tl
8870   \bool_set_false:N \l_stex_key_T_bool
8871   \tl_clear:N \l_stex_key_Ttext_tl
8872   \tl_clear:N \l_stex_key_Ftext_tl
8873 }{
8874   feedback .tl_set:N      = \l_stex_key_feedback_tl ,
8875   T        .code:n       = {\bool_set_true:N \l_stex_key_T_bool} ,
8876   F        .code:n       = {\bool_set_false:N \l_stex_key_T_bool} ,
8877   Ttext    .tl_set:N      = \l_stex_key_Ttext_tl ,
8878   Ftext    .tl_set:N      = \l_stex_key_Ftext_tl ,
8879 }{id}
8880
8881 \tl_set:Nn \problem_mcc_box_tl {
8882   \ltx@ifpackageloaded{amssymb}{$\square$}{
8883     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
8884   }
8885 }
8886 \newcommand\mcc[2][]{
8887   \stex_keys_set:nn{mcc}{#1}
8888   \tl_set:Nn \l_tmpa_tl{
8889     \bool_if:NTF \l_stex_key_T_bool {
8890       \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
8891     }{
8892       \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
8893     }
8894     \tl_if_empty:NF \l_stex_key_feedback_tl {
8895       \\emph{\l_stex_key_feedback_tl}
8896     }
8897   }

```

```

8898
8899 \__problems_mccline:n{
8900   \stex_if_do_html:TF{
8901     \stex_annotate:nn{shtml:mcc={
8902       \bool_if:NTF \l_stex_key_T_bool {true}{false}
8903     }}{
8904       #2\stex_annotate:nn{shtml:mcc-solution={}}{\l_tmpa_tl}
8905     }
8906   }{
8907     #2\ifsolutions\footnote{\l_tmpa_tl}\fi
8908   }
8909 }
8910 }
8911 \stex_deactivate_macro:Nn \mcc {mcb~environments}

```

(End of definition for \mcc. This function is documented on page 112.)

**scb** (*env.*)

```

8912
8913 \tl_set:Nn\problem_scc_box_tl{${\bigcirc}}
8914
8915 \stex_new_stylable_env:nnnnnn{scb}{0}{}{
8916   \stex_keys_set:nn{style}{#1}
8917   \stex_if_do_html:T{
8918     \exp_args:Nne\begin{stex_annotate_env}{
8919       shtml:single-choice-block={
8920         \clist_if_empty:NF \l_stex_key_style_clist {,
8921           shtml:styles={\l_stex_key_style_clist}
8922         }
8923       }
8924     \tl_set:Nn\problem_scc_box_tl{}
8925   }
8926   \stex_deactivate_macro:Nn \mcb {sproblem~environments}
8927   \stex_deactivate_macro:Nn \scb {sproblem~environments}
8928   \stex_deactivate_macro:Nn \solution {sproblem~environments}
8929   \stex_deactivate_macro:Nn \hint {sproblem~environments}
8930   \stex_deactivate_macro:Nn \exnote {sproblem~environments}
8931   \stex_deactivate_macro:Nn \gnote {sproblem~environments}
8932   \stex_reactivate_macro:N \scc
8933   \cs_set:Nn \__problems_sccline:n{
8934     \begin{list}{}{
8935       \setlength\topsep{0pt}
8936       \setlength\parsep{0pt}
8937       \setlength\rightmargin{0pt}
8938     }\item[\problem_scc_box_tl] ##1 \end{list}
8939   }
8940   \stex_style_apply:
8941 }{
8942   \stex_style_apply:
8943   \stex_if_do_html:T{
8944     \end{stex_annotate_env}
8945   }
8946 }{\par}{}{
8947 \stexstylescb[inline]{

```

```

8948 \cs_set:Nn \__problems_sccline:n{\problem_scc_box_tl{~} #1}
8949 }{}
8950
8951 \stex_deactivate_macro:Nn \scb {sproblem~environments}

```

\scc

```

8952
8953 \newcommand\scc[2] []{
8954   \stex_keys_set:nn{mcc}{#1}
8955   \tl_set:Nn \l_tmpa_tl{
8956     \bool_if:NTF \l_stex_key_T_bool {
8957       \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
8958     }{
8959       \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
8960     }
8961     \tl_if_empty:NF \l_stex_key_feedback_tl {
8962       \\emph{\l_stex_key_feedback_tl}
8963     }
8964   }
8965
8966   \__problems_sccline:n{
8967     \stex_if_do_html:TF{
8968       \stex_annotate:nn{shtml:scc={
8969         \bool_if:NTF \l_stex_key_T_bool {true}{false}
8970       }}{
8971         #2\stex_annotate:nn{shtml:scc-solution={}}{\l_tmpa_tl}
8972       }
8973     }{
8974       #2\ifsolutions\footnote{\l_tmpa_tl}\fi
8975     }
8976   }
8977 }
8978 \stex_deactivate_macro:Nn \scc {scb~environments}
8979
8980
8981 \newcommand\yesTnoF{
8982   \begin{scb}[style=inline]
8983     \scc[T]{yes}~\scc[F]{no}
8984   \end{scb}
8985 }
8986 \newcommand\yesFnoT{
8987   \begin{scb}[style=inline]
8988     \scc[F]{yes}~\scc[T]{no}
8989   \end{scb}
8990 }
8991 \newcommand>trueTfalseF{
8992   \begin{scb}[style=inline]
8993     \scc[T]{true}~\scc[F]{false}
8994   \end{scb}
8995 }
8996 \newcommand>trueFfalseT{
8997   \begin{scb}[style=inline]
8998     \scc[F]{true}~\scc[T]{false}
8999   \end{scb}

```

```
9000 }
```

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

### `\fillinsol`

```
9001 \stex_keys_define:nmmm{fillinsol}{
9002   \tl_clear:N \l__problems_fillin_solution_tl
9003   \dim_zero:N \l_stex_key_testspace_dim
9004 }{
9005   testspace .dim_set:N = \l_stex_key_testspace_dim,
9006   exact .code:n = {\__problems_parse_fillin_arg:nmmn{exact}#1 },
9007   numrange .code:n = {\__problems_parse_fillin_arg:nmmn{numrange}#1 },
9008   regex .code:n = {\__problems_parse_fillin_arg:nmmn{regex}#1 }
9009 }{}
9010
9011 \cs_new:Nn \__problems_parse_fillin_arg:nmmn {
9012   \stex_if_do_html:TF{
9013     \tl_set:Nn \l_tmpa_tl{#3}
9014     \tl_set:Nn \l_tmpb_tl{T}
9015     \stex_annotate_invisible:nn{
9016       shtml:fillin-case={#1},
9017       shtml:fillin-case-value={#2},
9018       shtml:fillin-case-verdict={
9019         \tl_if_eq:NNTF\l_tmpa_tl\l_tmpb_tl{true}{false}
9020       },
9021     }{#4}
9022   }{
9023     \tl_put_right:Nn \l__problems_fillin_solution_tl {
9024       #1{~} & #2{~} & #3{~} & #4{~} \\
9025     }
9026   }
9027 }
9028
9029 \newcommand\fillinsol[2] []{
9030   \quad
9031   \mode_if_math:TF{
9032     \hbox{\__problems_fillinsol:nn{#1}{#2$}}
9033   }{
9034     \__problems_fillinsol:nn{#1}{#2}
9035   }
9036   \quad
9037 }
9038 \cs_new_protected:Nn \__problems_fillinsol:nn {
9039   \stex_keys_set:nn{fillinsol}{#1}
9040   \stex_if_do_html:TF{
9041     \stex_annotate:nn{shtml:fillinsol={}}{ \stex_annotate_force_break:n{
9042       #2
9043       \l__problems_fillin_solution_tl
9044     } }
9045   }{
9046     \ifsolutions
9047     \textcolor{red}{\fbox{#2}}
9048     \tl_if_empty:NF \l__problems_fillin_solution_tl {
9049       \footnote{
```

```

9050         \halign{ ##\hfil & ##\hfil &##\hfil&##\hfil \cr
9051             \textbf{type }&\textbf{case }&\textbf{verdict }&\textbf{feedback } \cr
9052             \l__problems_fillin_solution_tl
9053         }
9054     }
9055 }
9056 \else
9057     \fbox{\dim_compare:nNnTF\l_stex_key_testspace_dim={0pt}{
9058         \phantom{\huge{#2}}
9059     }{
9060         \hspace{\l_stex_key_testspace_dim}
9061     }}
9062 \fi
9063 }
9064 }
9065 \stex_deactivate_macro:Nn \fillinsol {sproblem~environments}

```

(End of definition for `\fillinsol`. This function is documented on page 114.)

### `\testemptypage`

```

9066 \newcommand\testemptypage[1] [] {%
9067 \bool_if:NT \c__problems_test_bool {\ \vfill\begin{center}\hwexam@kw@testemptypage\end{center}
9068 }

```

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

### `\testspace`

```

9069 \newcommand\testspace[1]{\bool_if:NT \c__problems_test_bool {\vspace*{#1}}}
9070 \newcommand\testsmallspace{\testspace{1cm}}
9071 \newcommand\testmedspace{\testspace{2cm}}
9072 \newcommand\testbigspace{\testspace{3cm}}

```

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

### `\testnewpage`

```

9073 \newcommand\testnewpage{\bool_if:NT \c__problems_test_bool {\newpage}}

```

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

```

9074 \</package>

```

## 14.3 Implementation: The hwexam Package

### 14.3.1 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.

```

9075 <*package>
9076 \ProvidesExplPackage{hwexam}{2023/10/13}{3.4.0}{homework assignments and exams}
9077 \RequirePackage{l3keys2e}
9078
9079 \keys_define:nn {hwexam / pkg}{
9080 multiple .default:n = { false },

```

```

9081 multiple .bool_set:N = \c_hwexam_multiple_bool,
9082 unknown .code:n = {
9083 \PassOptionsToPackage{\CurrentOption}{problem}
9084 }
9085 }
9086 \ProcessKeysOptions{ hwexam /pkg }
9087 \RequirePackage{problem}

```

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

```

9088 \AddToHook{begindocument}{
9089 \ExplSyntaxOn\makeatletter
9090 \input{hwexam-english.ldf}
9091 \ltx@ifpackageloaded{babel}{
9092 \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
9093 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
9094 \input{hwexam-ngerman.ldf}
9095 }
9096 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
9097 \input{hwexam-finnish.ldf}
9098 }
9099 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
9100 \input{hwexam-french.ldf}
9101 }
9102 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
9103 \input{hwexam-russian.ldf}
9104 }
9105 }{}
9106 \makeatother\ExplSyntaxOff
9107 }

```

*(End of definition for `\hwexam_kw_*`. This function is documented on page ??.)*

### 14.3.2 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.

`assignment (env.)`

```

9108 \stex_keys_define:nnnn{ assignment }{
9109 \tl_clear:N \l_stex_key_number_tl
9110 \tl_clear:N \l_stex_key_given_tl
9111 \tl_clear:N \l_stex_key_due_tl
9112 }{
9113 number .tl_set:N = \l_stex_key_number_tl,
9114 given .tl_set:N = \l_stex_key_given_tl,
9115 due .tl_set:N = \l_stex_key_due_tl,
9116 unknown .code:n = {}
9117 }{id,title,style}
9118
9119 \newcounter{assignment}
9120 \stex_new_stylable_env:nnnnnn {assignment}{0{} }{
9121 \cs_if_exist:NTF \l_hwexam_includeassignment_keys_tl {

```

```

9122     \tl_put_left:Nn \l_hwexam_includeassignment_keys_tl {#1,}
9123     \exp_args:Nno \stex_keys_set:nn{assignment}{
9124       \l_hwexam_includeassignment_keys_tl
9125     }
9126   }{
9127     \stex_keys_set:nn{assignment}{#1}
9128   }
9129 \tl_if_empty:NF \l_stex_key_number_tl {
9130 \global\setcounter{assignment}{\int_eval:n{\l_stex_key_number_tl-1}}
9131 }
9132 \global\refstepcounter{assignment}
9133 \setcounter{sproblem}{0}
9134 \def\thesproblem{\theassignment.\arabic{sproblem}}
9135 \stex_style_apply:
9136 \_stex_do_id:
9137 }{
9138 \stex_style_apply:
9139 }{
9140 \par\begin{center}
9141 \textbf{\Large\assignmentautorefname~\theassignment}
9142 \tl_if_empty:NF \l_stex_key_title_tl {
9143 {~}---\l_stex_key_title_tl
9144 }
9145 }\par\smallskip
9146 \textbf{
9147 \tl_if_empty:NF \l_stex_key_given_tl {
9148 \hwexam@kw@given :~\l_stex_key_given_tl\quad
9149 }
9150 \tl_if_empty:NF \l_stex_key_due_tl {
9151 \hwexam@kw@due :~\l_stex_key_due_tl\quad
9152 }
9153 }
9154 \end{center}
9155 \par\bigskip
9156 }{
9157 \par\pagebreak
9158 }{}

```

`\includeassignment`

```

9159 \NewDocumentCommand\includeassignment{0{} m}{
9160 \group_begin:
9161 \tl_set:Nn \l_hwexam_includeassignment_keys_tl {#1}
9162 \stex_keys_set:nn{includeproblem}{#1}
9163 \exp_args:Nno \use:nn{\inputref[]\l_stex_key_mhrepos_str}{#2}
9164 \group_end:
9165 }

```

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

Restoring information about problems:

```

9166 \prop_new:N \c_@@_problems_prop
9167 \tl_set:Nn \c_@@_total_mins_tl {0}
9168 \tl_set:Nn \c_@@_total_pts_tl {0}
9169 \int_new:N \c_@@_total_problems_int
9170 \cs_set_protected:Npn \problem@restore #1 #2 #3 {

```

```

9171 \int_gincr:N \c_@@_total_problems_int
9172 \prop_gput:Nnn \c_@@_problems_prop {#1}{{#2}{{#3}}
9173 \tl_gset:Nx \c_@@_total_pts_tl { \int_eval:n { \c_@@_total_pts_tl + #2 }}
9174 \tl_gset:Nx \c_@@_total_mins_tl { \int_eval:n { \c_@@_total_mins_tl + #2 }}
9175 }

```

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

```

9176 \newcommand\correction@table{
9177 \int_compare:nNnT \c_@@_total_problems_int = 0 {
9178 \int_incr:N \c_@@_total_problems_int
9179 \prop_put:Nnn \c_@@_problems_prop {~}{{~}{{~}}
9180 }
9181 \tl_clear:N \l_tmpa_tl
9182 \tl_clear:N \l_tmpb_tl
9183 \tl_clear:N \l_tmpc_tl
9184 \prop_map_inline:Nn \c_@@_problems_prop {
9185 \tl_put_right:Nn \l_tmpa_tl { ##1 & }
9186 \tl_put_right:Nx \l_tmpb_tl { \use_i:nn ##2 & }
9187 \tl_put_right:Nn \l_tmpc_tl { & }
9188 }
9189 \resizebox{\textwidth}{!}{%
9190 \exp_args:Nne \begin{tabular}{|l|*{\int_use:N \c_@@_total_problems_int}{c|}c|l|}\hline
9191 &\exp_args:Ne \multicolumn{\int_eval:n{ \c_@@_total_problems_int + 1}}{c|}
9192 {\footnotesize\hwexam@kw@forgrading} &\\\hline
9193 \hwexam@kw@probs & \l_tmpa_tl \hwexam@kw@sum & \hwexam@kw@grade\\\hline
9194 \hwexam@kw@pts & \l_tmpb_tl \c_@@_total_pts_tl & \hwexam@kw@reqpts\\\hline
9195 \hwexam@kw@reached & \l_tmpc_tl & \hwexam@kw@tools\\\hline
9196 \end{tabular}}

```

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

`\testheading`

```

9197 \def\hwexamheader{\input{hwexam-default.header}}
9198
9199 \def\hwexamminutes{
9200 \tl_if_empty:NTF \hwexam@duration {
9201 {\hwexam@min}~\hwexam@minutes@kw
9202 }{
9203 \hwexam@duration
9204 }
9205 }
9206
9207 \stex_keys_define:nnnn{ hwexam / testheading }{
9208 \tl_clear:N \hwexam@min
9209 \tl_clear:N \hwexam@duration
9210 \tl_clear:N \hwexam@reqpts
9211 \tl_clear:N \hwexam@tools
9212 }{
9213 min .tl_set:N = \hwexam@min,
9214 duration .tl_set:N = \hwexam@duration,
9215 reqpts .tl_set:N = \hwexam@reqpts,
9216 tools .tl_set:N = \hwexam@tools
9217 }{}
9218

```



```

9219 \newenvironment{testheading}[1][]{
9220 \stex_keys_set:nn { hwexam / testheading}{#1}
9221
9222 \tl_set_eq:NN \hwexam@totalpts \c_@@_total_pts_tl
9223 \tl_set_eq:NN \hwexam@totalmin \c_@@_total_mins_tl
9224 \tl_set:Nx \hwexam@checktime {\int_eval:n { \hwexam@min - \hwexam@totalmin }}
9225
9226 \newif\if@bonuspoints
9227 \tl_if_empty:NTF \hwexam@reqpts {
9228 \@bonuspointsfalse
9229 }{
9230 \tl_set:Nx \hwexam@bonuspts {
9231 \int_eval:n{\hwexam@totalpts - \hwexam@reqpts}
9232 }
9233 \@bonuspointstrue
9234 }
9235
9236 \makeatletter\hwexamheader\makeatother
9237 }{
9238 \newpage
9239 }

```

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

```

9240 \end{package}

```

### 14.3.3 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}

```

## 14.4 Tikzinput Implementation

```

9241 \@=tikzinput)
9242 \end{package}
9243
9244 %%%%%%%%%%% tikzinput.dtx %%%%%%%%%%%
9245

```

```

9246 \ProvidesExplPackage{tikzinput}{2023/10/13}{3.4.0}{tikzinput package}
9247 \RequirePackage{l3keys2e}
9248
9249 \keys_define:nn { tikzinput } {
9250   image .bool_set:N = \c_tikzinput_image_bool,
9251   image .default:n = false ,
9252   unknown .code:n = {}
9253 }
9254
9255 \ProcessKeysOptions { tikzinput }
9256
9257 \bool_if:NTF \c_tikzinput_image_bool {
9258   \RequirePackage{graphicx}
9259
9260   \providecommand\usetikzlibrary[]{}
9261   \newcommand\tikzinput [2] []{\includegraphics[#1]{#2}}
9262 }{
9263   \RequirePackage{tikz}
9264   \RequirePackage{standalone}
9265
9266   \newcommand \tikzinput [2] [] {
9267     \setkeys{Gin}{#1}
9268     \ifx \Gin@ewidth \Gin@exclamation
9269       \ifx \Gin@eheight \Gin@exclamation
9270         \input { #2 }
9271       \else
9272         \resizebox{!}{ \Gin@eheight }{
9273           \input { #2 }
9274         }
9275       \fi
9276     \else
9277       \ifx \Gin@eheight \Gin@exclamation
9278         \resizebox{ \Gin@ewidth }{!}{
9279           \input { #2 }
9280         }
9281       \else
9282         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
9283           \input { #2 }
9284         }
9285       \fi
9286     \fi
9287   }
9288 }
9289
9290 \newcommand \ctikzinput [2] [] {
9291   \begin{center}
9292     \tikzinput [#1] {#2}
9293   \end{center}
9294 }
9295 </package>

```

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
$\backslash$ \$	2086, 2089, 2093
$\backslash$ ;	7511, 7542, 7555, 7557, 7570
@@ commands:	
$\backslash$ c_@@_problems_prop	9166, 9172, 9179, 9184
$\backslash$ c_@@_total_mins_tl	9167, 9174, 9223
$\backslash$ c_@@_total_problems_int	9169, 9171, 9177, 9178, 9190, 9191
$\backslash$ c_@@_total_pts_tl	9168, 9173, 9194, 9222
$\backslash\backslash$	53, 77, 1055, 7842, 7847, 8202, 8205, 8895, 8962, 9024, 9192, 9193, 9194, 9195
$\backslash$ {	7560
$\backslash$ }	7560
$\backslash$ _	44, 640, 7874, 9067
$\backslash$ _comp	3954, 4160, 4512, 5208, 6027, 6051, 6135
$\backslash$ _customthiscomp	6554, 6555, 6567
$\backslash$ _defcomp	6059, 6993, 7002
$\backslash$ _stex_html_do_output_bool	279, 280, 283, 288, 292, 323, 326, 2155, 7502
$\backslash$ _thiscomp	6537, 6548, 6549
$\backslash$ _varcomp	4493, 4792, 4801, 4958, 4996, 5242, 5684, 5983, 5996, 6009, 6055
$\backslash$	2085, 2088, 2092
A	
$\backslash$ activateexcursion	109
$\backslash$ addbibresource	80, 1955
$\backslash$ addcontentsline	7872
$\backslash$ addmhbibresource	80, 1949
$\backslash$ addtocounter	8048, 8428, 8429
$\backslash$ AddToHook	1037, 1040, 7306, 7373, 7376, 7488, 8317, 9088
$\backslash$ aftergroup	143, 5196, 7371, 7485
$\backslash$ afterprematrestop	108, 7931, 7942
$\backslash$ anscls	8745, 8779, 8813
$\backslash$ apply	88
$\backslash$ arabic	8358, 8361, 9134
$\backslash$ arg	46, 93, 5421
$\backslash$ argarraymap	92, 4221, 4732
$\backslash$ argmap	92, 4220, 4362, 4708, 7543
$\backslash$ argsep	40, 92, 4219, 4357, 4692, 7511, 7512, 7513, 7519
$\backslash$ assign	64, 129, 2991, 3347, 3474
assignment (env.)	73, 116, 9108
$\backslash$ assignmentautorefname	9141
$\backslash$ assignMorphism	129, 2992, 3476
$\backslash$ assumption	7284, 7389
$\backslash$ ast	7534
$\backslash$ AtBeginDocument	1007, 1260, 1446, 1449, 1497, 7743, 7886
$\backslash$ AtEndDocument	648, 1498
$\backslash$ AtEndOfPackageFile	2036, 2050, 2063
$\backslash$ author	8182
$\backslash$ autoref	84, 1642
B	
$\backslash$ backmatter	1461, 1462, 1463
$\backslash$ baselineskip	8171
$\backslash$ beameritemnestingprefix	8256
$\backslash$ begin	96, 135, 136, 141, 201, 1303, 1383, 2047, 2060, 2078, 2235, 2257, 2274, 2524, 2782, 2828, 4755, 6888, 7198, 7231, 7253, 7291, 7362, 7434, 7630, 7875, 8008, 8022, 8032, 8065, 8086, 8094, 8132, 8133, 8134, 8135, 8136, 8137, 8143, 8145, 8217, 8245, 8246, 8398, 8489, 8511, 8563, 8627, 8676, 8728, 8787, 8828, 8843, 8918, 8934, 8982, 8987, 8992, 8997, 9067, 9140, 9190, 9291
$\backslash$ begingroup	359, 1927
$\backslash$ bf	8170
$\backslash$ bfseries	8473
$\backslash$ bgroup	3797, 4838, 5050, 8021, 8110, 8125, 8202, 8205, 8580, 8641, 8690, 8742
$\backslash$ bigcirc	8913
$\backslash$ bigskip	8448, 9155
blindfragment (env.)	82
bool commands:	
$\backslash$ bool_if:NTF	197, 619, 671, 672, 678, 1145, 1155, 1157, 1216, 1237, 1450, 2138, 2393, 2743, 2776, 2960, 3456, 3665, 4853, 4956, 5206, 5240, 5782, 5876, 5882, 6017, 6983, 7259, 7307, 7345, 7364, 7367, 7369, 7374, 7377, 7385, 7427, 7441, 7463, 7489, 7778, 7786, 7823, 7902, 7946, 8107, 8113,

8147, 8162, 8181, 8216, 8226, 8304,	
8310, 8380, 8401, 8435, 8440, 8449,	
8471, 8492, 8500, 8503, 8516, 8517,	
8521, 8522, 8638, 8649, 8687, 8698,	
8739, 8751, 8815, 8820, 8889, 8902,	
8956, 8969, 9067, 9069, 9073, 9257	
\bool_if:nTF . . . . .	283
\bool_if_exist:NTF . . . . .	256, 272
\bool_lazy_any:nTF . . . . .	4049, 4089
\bool_lazy_any_p:n . . . . .	774
\bool_new:N . . . . .	
279, 2135, 2378, 2801, 4764, 5176,	
5751, 5873, 8337, 8375, 8376, 8377	
\bool_not_p:n . . . . .	773, 777
\bool_set_false:N . . . . .	182, 288, 326,
1142, 1166, 2136, 2155, 2379, 2756,	
2823, 3429, 4624, 4767, 5180, 5752,	
5877, 7117, 7266, 7314, 7342, 7483,	
7502, 7503, 7762, 7803, 7962, 8343,	
8378, 8417, 8420, 8865, 8870, 8876	
\bool_set_true:N . . . . .	187, 258, 274,
280, 292, 323, 608, 614, 1139, 2154,	
2740, 2820, 3441, 3451, 3662, 3798,	
4858, 5098, 5177, 5406, 5415, 5531,	
5542, 5640, 5725, 5783, 5794, 5827,	
5856, 5902, 6098, 6509, 6540, 6557,	
6602, 7129, 7296, 7775, 7815, 7960,	
7968, 7969, 7970, 7971, 7972, 7973,	
8413, 8418, 8421, 8476, 8863, 8875	
\bool_while_do:Nn . . . . .	1140
\bool_while_do:nn . . . . .	
772, 1987, 7133, 7145, 7157, 7174, 7186	
\l_tmpa_bool . . . . .	3429, 3441, 3451, 3456
box commands:	
\box_clear:N . . . . .	2158
\c_empty_box . . . . .	3979, 3986, 8817, 8822
\l_tmpa_box . . . . .	2153, 2158, 3677, 4212,
8110, 8125, 8580, 8641, 8690, 8742	
boxed . . . . .	110
C	
\catcode . . . . .	44, 53, 359, 640, 1055,
1695, 2084, 2085, 2086, 2087, 2088,	
2089, 2091, 2092, 2093, 2435, 2436	
\cdot . . . . .	7542, 7555
\centering . . . . .	7876, 8066
\chapter . . . . .	27, 82, 7911, 7912
\chaptername . . . . .	7847, 7908, 7909
\chaptertitlename . . . . .	7847
\circ . . . . .	54
\clearpage . . . . .	1456, 1466, 1477, 1488
clist commands:	
\clist_clear:N . . . . .	94, 406, 3706, 4138,
5099, 6391, 6490, 6805, 6810, 7391	
\clist_count:N . . . . .	6371, 6381
\clist_count:n . . . . .	4740, 6473
\clist_get:NN . . . . .	462, 513
\clist_if_empty:NTF . . . . .	
. . . . .	181, 458, 509, 3814, 4310,
6370, 6416, 6673, 6877, 8830, 8920	
\clist_if_in:NnTF . . . . .	
. . . . .	66, 69, 100, 6971, 8322, 8325,
8328, 8331, 9093, 9096, 9099, 9102	
\clist_if_in:nnTF . . . . .	6499
\clist_item:Nn . . . . .	4319
\clist_item:nn . . . . .	4751
\clist_map_function:NN . . . . .	5100, 6419
\clist_map_function:nN . . . . .	
. . . . .	2693, 2718, 3596, 3616, 3637
\clist_map_inline:Nn . . . . .	
. . . . .	95, 104, 184, 2845, 3816, 4876,
5069, 6198, 6248, 6674, 6787, 6823	
\clist_map_inline:nn . . . . .	
. . . . .	370, 419, 5572, 5671, 5700, 6196, 7073
\clist_pop:NN . . . . .	4321
\clist_put_right:Nn . . . . .	
. . . . .	96, 5113, 5117, 6398, 6517, 6519
\clist_set:Nn . . . . .	38, 92, 4977, 8321, 9092
\clist_set_eq:NN . . . . .	90, 6893
\l_tmpa_clist . . . . .	8321, 8322, 8325, 8328,
8331, 9092, 9093, 9096, 9099, 9102	
\clstinputmhlisting . . . . .	83, 2035
\cmhgraphics . . . . .	83, 2035
\cmhtikzinput . . . . .	118, 2035
\columnbox . . . . .	8084, 8086, 8104
\comp . . . . .	32, 33, 36, 46, 90, 93,
103, 131, 3954, 4160, 4333, 4493,	
4512, 4554, 4560, 4562, 4792, 4801,	
4996, 5225, 5226, 5560, 5684, 5904,	
5909, 5924, 5983, 5996, 6009, 6027,	
6029, 6036, 6135, 6358, 6366, 6549,	
6563, 6564, 6993, 6994, 7002, 7508,	
7510, 7517, 7519, 7521, 7522, 7527,	
7529, 7534, 7536, 7537, 7544, 7545,	
7546, 7547, 7550, 7552, 7556, 7557,	
7558, 7560, 7570, 7577, 7579, 7581	
\compemph . . . . .	103, 6036
\conclude . . . . .	7283, 7389
\conclusion . . . . .	50–52, 99, 101, 6962, 7082, 7098
\copymod . . . . .	62, 63, 3606, 3608, 3610
\copymodule . . . . .	3493, 3495
\counterwithin . . . . .	7915, 7927
\cr . . . . .	9050, 9051
cs commands:	
\cs:w . . . . .	445, 448, 480, 641, 2168,
2397, 2398, 2399, 2406, 2410, 2416,	
5304, 6239, 6289, 6291, 6911, 6924	
\cs_argument_spec:N . . . . .	4003

<code>\cs_end:</code>	. 445, 448, 480, 641, 2168, 2397, 2398, 2399, 2408, 2412, 2416, 5304, 6239, 6289, 6291, 6911, 6924	2201, 2223, 2229, 2247, 2270, 2279, 2295, 2302, 2315, 2328, 2343, 2351, 2360, 2381, 2403, 2424, 2446, 2450, 2458, 2470, 2555, 2559, 2563, 2572, 2580, 2586, 2604, 2611, 2623, 2637, 2644, 2657, 2671, 2688, 2697, 2708, 2713, 2722, 2733, 2738, 2762, 2780, 2794, 2802, 2843, 2852, 2864, 2868, 2880, 2888, 2896, 2911, 2924, 2970, 2976, 2997, 3040, 3073, 3140, 3159, 3179, 3185, 3190, 3217, 3248, 3273, 3299, 3349, 3393, 3460, 3543, 3571, 3581, 3676, 3684, 3747, 3759, 3778, 3823, 3831, 3885, 3894, 3976, 3985, 3990, 3997, 4018, 4035, 4048, 4059, 4061, 4074, 4085, 4177, 4193, 4244, 4256, 4283, 4304, 4309, 4330, 4341, 4350, 4460, 4502, 4544, 4573, 4580, 4601, 4615, 4651, 4656, 4661, 4692, 4765, 4809, 4823, 4836, 4886, 4904, 4917, 4923, 4933, 4941, 5004, 5018, 5031, 5032, 5034, 5078, 5111, 5130, 5142, 5154, 5183, 5205, 5220, 5239, 5264, 5269, 5274, 5341, 5357, 5363, 5398, 5404, 5412, 5467, 5528, 5537, 5553, 5566, 5600, 5611, 5632, 5664, 5696, 5742, 5745, 5755, 5768, 5781, 5804, 5805, 5806, 5810, 5836, 5839, 5865, 5875, 5963, 6015, 6036, 6130, 6155, 6162, 6179, 6185, 6234, 6294, 6302, 6309, 6314, 6368, 6390, 6397, 6401, 6464, 6489, 6515, 6547, 6553, 6572, 6584, 6595, 6613, 6623, 6663, 6672, 6680, 6689, 6699, 6710, 6724, 6748, 6830, 6882, 6928, 6941, 6979, 7050, 7066, 7142, 7230, 7237, 7241, 7252, 7267, 7384, 7407, 7425, 7740, 7836, 7907, 7919, 7958, 7996, 8020, 8025, 8031, 8040, 8112, 8236, 8460, 8554, 8618, 8667, 8718, 8861, 9038
<code>\cs_generate_from_arg_count:NNnn</code>	..... 3807, 4620, 4867, 5060, 5144, 5345, 5377	
<code>\cs_generate_variant:Nn</code>	..... ..... 87, 157, 231, 244, 558, 582, 591, 603, 627, 687, 716, 871, 876, 880, 964, 1126, 1151, 1555, 1883, 2200, 2558, 2562, 2567, 2585, 2634, 2656, 2669, 3124, 5201, 5218, 5535	
<code>\cs_gset:Npn</code>	..... 2431	
<code>\cs_if_eq:NNTF</code>	..... 140, 886, 940, 1422, 2224, 2235, 2238, 2257, 2260, 4005, 4693, 5616, 5618	
<code>\cs_if_exist:NTF</code>	..... 309, 1094, 1328, 1332, 1370, 1373, 1397, 1401, 1430, 1436, 1451, 1461, 1474, 1485, 1641, 1789, 1813, 1833, 1839, 1843, 2106, 2646, 2651, 3981, 3988, 4000, 4431, 4439, 4891, 4898, 5085, 5131, 5295, 5321, 5665, 5904, 5909, 6455, 6752, 6755, 6759, 6762, 6767, 6770, 6774, 6777, 7842, 7847, 7850, 7854, 7858, 7862, 7866, 7908, 7911, 7914, 7920, 7923, 7926, 8068, 8076, 8088, 8098, 8202, 8205, 8383, 9121	
<code>\cs_new:Nn</code>	..... 209, 432, 546, 552, 559, 568, 584, 593, 765, 885, 2209, 2392, 2440, 2951, 3774, 4108, 4121, 4325, 4392, 4409, 4414, 4569, 5140, 5353, 5450, 6176, 6231, 6427, 6433, 6436, 6446, 6868, 8363, 9011	
<code>\cs_new:Npn</code>	544, 621, 622, 630, 631, 881, 2892, 2923, 4114, 4127, 5596, 5969	
<code>\cs_new_nopar:Nn</code>	..... 366, 381	
<code>\cs_new_protected:Nn</code>	43, 52, 60, 65, 75, 81, 84, 136, 158, 216, 227, 237, 246, 264, 286, 329, 335, 349, 355, 437, 444, 457, 471, 479, 498, 508, 523, 545, 624, 634, 646, 651, 670, 682, 693, 709, 718, 733, 791, 798, 828, 863, 873, 877, 899, 916, 929, 935, 939, 966, 970, 986, 1002, 1008, 1021, 1086, 1093, 1116, 1127, 1136, 1153, 1163, 1172, 1199, 1246, 1254, 1300, 1308, 1312, 1322, 1368, 1381, 1391, 1428, 1508, 1518, 1544, 1584, 1618, 1630, 1640, 1648, 1688, 1746, 1757, 1788, 1794, 1798, 1812, 1819, 1861, 1886, 1902, 1926, 1937, 1949, 1976, 2006, 2080, 2096, 2119, 2123, 2127, 2142, 2146, 2152, 2163, 2171,	
<code>\cs_new_protected:Npn</code>	..... ..... 206, 338, 341, 345, 346, 429, 689, 892, 895, 1242, 1326, 1395, 1411, 1499, 1501, 1832, 1838, 1842, 1857, 2215, 2298, 2374, 2857, 2917, 2955, 2983, 3014, 3028, 3111, 3126, 3260, 3589, 4429, 4586, 4593, 4643, 4699, 4708, 4733, 5084, 5097, 5162, 5168, 5172, 5202, 5252, 5293, 5319, 5444, 5580, 5894, 5914, 5921, 6051, 6055, 6059, 6065, 6069, 6073, 6077, 6081, 6085, 6089, 6093, 6097, 6335, 6357, 6362, 6453, 6472, 6485, 6537, 6784, 6821, 7131, 7155, 7172,	

7184, 7227, 7590, 8466, 8609, 8612	8185, 8187, 8189, 8191, 8192, 8194,
\cs_parameter_spec:N . . . . . 548, 562	8196, 8198, 8201, 8202, 8204, 8205,
\cs_prefix_spec:N . . . . . 560	8208, 8210, 8212, 8256, 8358, 8361,
\cs_set:Nn . . . . . 8842, 8857, 8933, 8948	8362, 8814, 8819, 9134, 9197, 9199
\cs_set:Npn . . . . . 449, 484, 489,	\defemph . . . . . 103, 6036
637, 638, 820, 821, 822, 823, 944,	\Definame . . . . . 100, 6937, 6967, 7020, 7026
951, 1095, 1098, 1111, 2177, 2426,	\definame . . . . . 24, 36,
2459, 2673, 2675, 2678, 2691, 2716,	45, 99, 100, 103, 6936, 6966, 7012, 7018
2741, 2755, 3808, 4021, 4214, 4215,	\definiendum . . . . . 24, 36,
4219, 4220, 4221, 4333, 4363, 4604,	45, 99, 100, 103, 6934, 6964, 7006, 7010
4620, 4714, 4739, 4868, 4906, 5061,	definiendum . . . . . 6979
5144, 5346, 5378, 5698, 6492, 6498,	\definiens . . . . . 47,
6559, 7499, 7500, 7501, 7818, 8117	57, 64, 99–101, 6952, 7029, 7047, 7048
\cs_set:Npx . . . . . 2615	\defnotation . . . . .
\cs_set_eq:NN 41, 218, 222, 814, 815,	. . . 36, 45, 100, 6935, 6965, 7001, 7004
816, 1262, 1269, 1387, 2652, 2994,	\detokenize 247, 356, 1661, 8322, 8325,
2995, 4001, 4467, 4471, 4618, 4671,	8328, 8331, 9093, 9096, 9099, 9102
4684, 4899, 5132, 5135, 5209, 5211,	dim commands:
5243, 5245, 5421, 5424, 5452, 5453,	\dim_compare:nNnTF . . . . . 9057
5666, 5669, 5808, 7666, 7678, 8160	\dim_new:N . . . . . 8545
\cs_set_protected:Nn . . . . .	\dim_zero:N . . . . . 8548, 9003
. . . . . 811, 818, 1568, 7837, 7870	\dimexpr . . . . . 8148
\cs_set_protected:Npn . . . . . 351,	do commands:
1281, 1290, 3977, 3978, 4357, 4362,	\_do_comp:nNn . . . . .
4368, 6063, 7667, 7909, 7912, 7921,	. . . . . 4369, 6036, 6052, 6056, 6060
7924, 8161, 8258, 8261, 8264, 9170	\dobracket . . . . . 92
\cs_set_protected:Npx . . . . . 6555	\dobrackets . . . . . 92, 5869, 5922
\cs_undefine:N . . . 213, 223, 270, 1207	\dowithbrackets . . . . . 5921
\l_tmpa_cs . . . . . 4214, 4224, 5698, 5716	\due . . . . . 73, 116
\csname . . . . . 268,	\duration . . . . . 74, 116
297, 356, 2082, 2083, 2084, 2085,	
2086, 2091, 2092, 2093, 2396, 2659	<b>E</b>
\ctikzinput . . . . . 118, 9290	\edef . . . 120, 2084, 2085, 2086, 6549, 8014
\CurrentFile . . . . . 1024, 1026, 1028	\egroup . . . . .
\CurrentFilePath . . . . . 1024, 1025, 1028	3821, 4883, 5076, 8028, 8110, 8127,
\currentgrouplevel . 228, 250, 251, 252,	8202, 8205, 8594, 8655, 8704, 8757
254, 256, 258, 266, 268, 270, 272, 274	\eject . . . . . 9067
\CurrentOption . . . . . 10, 7765, 7766,	\ellipses . . . . . 95,
7767, 7768, 7807, 7808, 8298, 9083	96, 4718, 4719, 5112, 5114, 5672, 5710
\Currentsectionlevel . . . . . 82, 1290	\else . . . . . 296, 308, 1890,
\currentsectionlevel . . . . . 82, 1281	1973, 2211, 5899, 6078, 8256, 8472,
	8577, 8593, 9056, 9271, 9276, 9281
<b>D</b>	\emph 17, 19, 102, 6086, 7214, 7302, 7319,
\DeclareOption . . . . . 10	8599, 8660, 8709, 8762, 8895, 8962
\def . . . . . 91, 141,	\end . . . . . 136, 141, 1309,
303, 307, 310, 312, 358, 561, 1244,	1363, 2047, 2060, 2078, 2238, 2260,
1248, 1265, 1270, 2052, 2055, 2066,	2283, 2543, 2798, 2876, 4757, 6905,
2070, 3954, 4160, 4181, 4491, 4493,	7212, 7238, 7300, 7317, 7368, 7370,
4510, 4512, 4792, 4799, 4801, 4994,	7448, 7452, 7654, 7880, 7934, 7943,
4996, 5225, 5226, 5684, 5924, 5953,	8016, 8028, 8042, 8082, 8096, 8104,
5967, 5983, 5996, 6009, 6027, 6038,	8117, 8119, 8132, 8133, 8134, 8135,
6135, 6562, 6567, 6993, 7002, 7930,	8136, 8137, 8158, 8219, 8253, 8254,
7932, 7997, 7998, 7999, 8003, 8006,	8432, 8509, 8527, 8586, 8591, 8647,
8049, 8152, 8169, 8177, 8182, 8183,	8652, 8696, 8701, 8749, 8754, 8811,

	8847, 8853, 8938, 8944, 8984, 8989, 8994, 8999, 9067, 9154, 9196, 9293	
<code>\endcsname</code>	268, 296, 297, 356, 2082, 2083, 2084, 2085, 2086, 2091, 2092, 2093, 2396, 2659, 8256	
<code>\endgroup</code>	361, 363, 1934	
<code>\endinput</code>	1769, 2285	
<code>\ensuremath</code>	5784, 5795	
environments:		
<code>assignment</code>	73, 116, 9108	
<code>blindfragment</code>	82	
<code>exnote</code>	1, 8667	
<code>extstructure</code>	96, 6194	
<code>extstructure*</code>	96	
<code>frame</code>	1, 7958	
<code>gnote</code>	1, 8716	
<code>hint</code>	1, 8615	
<code>mathstructure</code>	96, 6108	
<code>mcb</code>	1, 8824	
<code>mntinterface</code>	7620	
<code>nassertion</code>	1	
<code>ndefinition</code>	1	
<code>nexample</code>	1	
<code>note</code>	1	
<code>nparagraph</code>	1, 1	
<code>nsproof</code>	1	
<code>problem</code>	1	
<code>sassertion</code>	99	
<code>scb</code>	8912	
<code>sdefinition</code>	99	
<code>sexample</code>	99	
<code>sfragment</code>	82	
<code>smodule</code>	86, 2515	
<code>solution</code>	1, 8544	
<code>sparagraph</code>	99	
<code>sproblem</code>	8363	
<code>sproof</code>	101, 7230	
<code>stex_annotate_env</code>	141	
<code>subproblem</code>	8467	
<code>subproof</code>	7323	
<code>testheading</code>	74, 116	
<code>\eq</code>	32, 42	
<code>\eqstep</code>	7285, 7389	
<code>\equal</code>	31	
<code>\errmessage</code>	7697, 7711, 7718, 7732	
<code>\escapechar</code>	53, 1055	
<code>\everyeof</code>	2164	
<code>\excursion</code>	109, 8214	
<code>\excursiongroup</code>	109, 8231	
<code>\excursionref</code>	109, 8215, 8228	
<code>exnote (env.)</code>	1, 8667	
<code>\exnote</code>	8369, 8715, 8839, 8930	
exp commands:		
<code>\exp_after:wN</code>	356, 445, 448, 480, 547, 698, 887, 2166, 2210, 2211, 2212, 2352, 2397, 2398, 2399, 2406, 2410, 2414, 2415, 2937, 2958, 3574, 3811, 4110, 4123, 4722, 4750, 4871, 5064, 5301, 5303, 5311, 5313, 5359, 5366, 5367, 5368, 5387, 5420, 5583, 5585, 5687, 5688, 5689, 5702, 5897, 5960, 6006, 6022, 6238, 6288, 6291, 6615, 6911, 6924, 6988, 7023	
<code>\exp_args:Ne</code>	535, 540, 554, 555, 886, 940, 1028, 1101, 1128, 1652, 1801, 1814, 1828, 1844, 1847, 1850, 1852, 2346, 2840, 2846, 2936, 2939, 3368, 3396, 3402, 3770, 3779, 3939, 4003, 4005, 4693, 4838, 5035, 5470, 5616, 5618, 5627, 6199, 6249, 6336, 6340, 6418, 6499, 6548, 6554, 6788, 6824, 6917, 7202, 7409, 7741, 9191	
<code>\exp_args:NNe</code>	54, 198, 201, 595, 1073, 1078, 1084, 2460, 2899, 3548, 4042, 4228, 4326, 5502, 5505, 5589, 5636, 5721, 6519, 6627, 6682, 6750, 8248, 8461	
<code>\exp_args:Nne</code>	207, 547, 2028, 2099, 2484, 2524, 2581, 2782, 2860, 4184, 4335, 4374, 4493, 4512, 4801, 4996, 6137, 6574, 6615, 6683, 6888, 7103, 7253, 7431, 7630, 8398, 8489, 8828, 8918, 9190	
<code>\exp_args:NNNo</code>	832, 1176, 3558, 4574, 4576, 4810, 5005	
<code>\exp_args:NNno</code>	267, 684, 832	
<code>\exp_args:Nnno</code>	352, 1728, 1736	
<code>\exp_args:NNNx</code>	1176, 1717	
<code>\exp_args:NNnx</code>	837, 3075, 4810, 5005, 5481, 5487	
<code>\exp_args:NNo</code>	38, 48, 66, 69, 96, 100, 165, 173, 696, 805, 837, 841, 846, 851, 1016, 1179, 1659, 1990, 2000, 3208, 3234, 3555, 3561, 3565, 4276, 4289, 4632, 5715	
<code>\exp_args:Nno</code>	252, 254, 267, 379, 2427, 2962, 2965, 3806, 4042, 4866, 5059, 5165, 5170, 5254, 6045, 6508, 7051, 8385, 8540, 9123, 9163	
<code>\exp_args:NNx</code>	55, 110, 122, 911, 998, 1899, 1947, 2128, 2147, 6641, 8322, 8325, 8328, 8331, 9093, 9096, 9099, 9102	
<code>\exp_args:Nnx</code>	247, 1317, 1593, 1661, 1899, 1947, 2815, 3016, 3021, 3208, 3234, 3760, 3930, 5143, 5344,	

5376, 5386, 5896, 6169, 6845, 7668	<b>F</b>
<code>\exp_args:No</code> . . . . . 142, 162, 163, 164, 211, 302, 440, 607, 613, 751, 1026, 1112, 1212, 1264, 1272, 1294, 1317, 1532, 1717, 1722, 1727, 1759, 1764, 1766, 2074, 2367, 2464, 2519, 2556, 2885, 2897, 2934, 3121, 3136, 3378, 3379, 3572, 3577, 3661, 3765, 3766, 3767, 4040, 4196, 4565, 4740, 4816, 4817, 4818, 4829, 4830, 4831, 5011, 5012, 5013, 5024, 5025, 5026, 5118, 5187, 5190, 5194, 5235, 5374, 5379, 5384, 5389, 5391, 5479, 5507, 5616, 5618, 5645, 5730, 6147, 6157, 6189, 6262, 6374, 6378, 6385, 6412, 6447, 6449, 6521, 6523, 6529, 6604, 6630, 6636, 6638, 6643, 6647, 6653, 6700, 6749, 6992, 7042, 7831, 7890, 7891, 7894, 8321, 8404, 8496, 9092	<code>\fbox</code> . . . . . 9047, 9057
<code>\exp_args:Nx</code> . . . . . . . . 249, 697, 752, 760, 1536, 1642, 1644, 2216, 2521, 3000, 7959, 8862	<code>\fi</code> . . . . . 299, 314, 1894, 1909, 1917, 1973, 2212, 5207, 5241, 5912, 6016, 6078, 6980, 7553, 7936, 8004, 8007, 8106, 8256, 8474, 8581, 8595, 8907, 8974, 9062, 9275, 9285, 9286
<code>\exp_not:N</code> 56, 211, 213, 561, 946, 953, 1111, 1222, 1225, 1952, 2009, 2164, 2210, 2420, 2816, 3371, 3405, 4195, 4582, 5146, 5149, 5185, 5186, 5189, 5193, 5197, 5354, 5429, 5435, 5637, 5641, 5644, 5722, 5726, 5728, 5898, 6256, 6448, 6522, 6528, 6564, 6603, 6629, 6635, 6644, 6655, 7842, 7847	<code>fiboxed</code> . . . . . 105
<code>\exp_not:n</code> . . . . . 211, 544, 1112, 1317, 1532, 2414, 2441, 2464, 2594, 2618, 3377, 3378, 3379, 3765, 3766, 3767, 3959, 4196, 4237, 4238, 4565, 4722, 4750, 4816, 4817, 4818, 4829, 4830, 4831, 5011, 5012, 5013, 5024, 5025, 5026, 5140, 5187, 5190, 5194, 5196, 5198, 5235, 5367, 5374, 5379, 5384, 5389, 5391, 5603, 5645, 5682, 5687, 5730, 6232, 6374, 6412, 6442, 6447, 6449, 6521, 6523, 6525, 6529, 6532, 6558, 6604, 6606, 6630, 6632, 6636, 6638, 6643, 6647, 6653	file commands: <code>\file_if_exist:nTF</code> . . . . . 635, 652, 1165
<code>\expandafter</code> . . . . . . . . 297, 361, 2082, 2083, 2084, 2085, 2086, 2396, 2659, 6078, 7934, 7935	<code>\filepath</code> . . . . . 138, 139
<code>\ExplSyntaxOff</code> 2133, 2573, 7748, 8335, 9106	<code>\fillinsol</code> . . . . . 114, 8367, 9001
<code>\ExplSyntaxOn</code> 2132, 2569, 7745, 8318, 9089	<code>\first</code> . . . . . 137
<code>\extref</code> . . . . . 85, 1556	<code>\fn</code> . . . . . 54
<code>extstructure</code> (env.) . . . . . 96, 6194	<code>\foo</code> . . . . . 15, 54, 88, 141, 143
<code>\extstructure</code> . . . . . 6226, 6228	<code>\foiname</code> . . . . . 15
<code>extstructure*</code> (env.) . . . . . 96	<code>\footnote</code> . . . . . 8907, 8974, 9049
	<code>\footnotesize</code> . . . . . 9192
	<code>\foral</code> . . . . . 46, 51
	<code>\forall</code> . . . . . 46, 7540, 7546, 7557
	<code>frame</code> (env.) . . . . . 1, 7958
	<code>\frameimage</code> . . . . . 108, 8141
	<code>frameimages</code> . . . . . 105
	<code>\frametitle</code> . . . . . 8058
	<code>\frontmatter</code> . . . . . 1451, 1452, 1453
	<code>\fun</code> . . . . . 45
	<code>\funspace</code> . . . . . 37
	<b>G</b>
	<code>\gdef</code> . . . . . 8214
	<code>\given</code> . . . . . 73, 116
	<code>\global</code> . . . . . 1244, 1248, 1265, 1270, 2083, 7586, 7587, 8610, 8613, 9130, 9132
	<code>gnote</code> (env.) . . . . . 1, 8716
	<code>\gnote</code> . . . . . 8370, 8768, 8840, 8931
	<code>gnotes</code> . . . . . 73, 110, 115
	group commands:
	<code>\group_begin:</code> . . . . . 43, 52, 636, 1054, 1695, 2175, 2434, 2471, 2790, 2810, 2834, 3065, 3289, 3330, 3678, 3732, 3897, 3899, 3901, 3903, 3963, 4169, 4487, 4533, 4795, 4950, 4990, 5119, 5123, 5221, 5374, 5530, 5538, 5612, 5633, 5681, 5697, 5900, 5937, 5945, 5956, 5978, 5989, 6002, 6539, 6556, 6912, 6981, 7030, 7083, 7297, 7364, 7386, 7449, 7470, 7498, 8537, 9160
	<code>\group_end:</code> . . . . . 47, 55, 642, 1058, 1742, 2198, 2452, 2485, 2799, 2815, 2877, 3070, 3295, 3336, 3680, 3740, 3897, 3899, 3901, 3903, 3968, 3988, 4172, 4498, 4536, 4806, 4962, 5001, 5090, 5108, 5120, 5124, 5165, 5298, 5309, 5326, 5337, 5359, 5387,



5392, 5408, 5441, 5533, 5550, 5612, 5636, 5689, 5721, 5908, 6033, 6456, 6512, 6542, 6565, 6575, 6641, 6921, 6998, 7044, 7096, 7308, 7374, 7378, 7427, 7464, 7489, 7588, 8541, 9164	<code>\group_insert_after:N</code> . . . . . 257, 273	<code>\ifsolutions</code> . . . . . 111, 8301, 8575, 8588, 8907, 8974, 9046
<b>H</b>		<code>\ifstexhtml</code> . . . . . 28, 81, 140, 296, 8472
<code>\halign</code> . . . . . 9050		<code>\ifvmode</code> 1909, 1917, 5207, 5241, 6016, 6980
<code>\have</code> . . . . . 7389		<code>\ifx</code> 2082, 7933, 8002, 8005, 9268, 9269, 9277
<code>\hbox</code> . . . . . 1256, 3061, 3364, 3797, 3951, 3980, 4351, 4838, 5050, 6210, 6794, 7218, 7242, 8202, 8205, 8883, 9032		<code>\ignorespaces</code> . . . . . 4806, 5001, 8108
hbox commands:		image . . . . . 117
<code>\hbox_set:Nn</code> . . . . . 3677, 4212		<code>\imply</code> . . . . . 51
<code>\hbox_unpack:N</code> 3979, 3986, 8817, 8822		<code>\importmodule</code> . . . . . 38, 49, 55, 93–95, 130, 133, 135, 136, 143, 212, 2988, 3267
<code>\HCode</code> . . . . . 309		<code>\includeassignment</code> . . . . . 9159
<code>\hfil</code> . . . . . 7224, 9050		<code>\includegraphics</code> . . . . . 83, 2045, 9261
<code>\hfill</code> . . . . . 7224		<code>\includeproblem</code> . . . . . 72, 114, 8529
hint (env.) . . . . . 1, 8615		<code>\indent</code> . . . . . 5207, 5241, 6016, 6980
<code>\hint</code> . . . . . 8368, 8666, 8838, 8929		<code>\infpref</code> 90, 91, 4276, 5867, 5870, 5903, 7510
hints . . . . . 73, 110, 115		<code>\inline*</code> . . . . . 100
<code>\hline</code> . . . . . 9190, 9192, 9193, 9194, 9195		<code>\inlineass</code> . . . . . 50, 55, 58, 100
<code>\href</code> . . . . . 1839		<code>\inlinedef</code> . . . . . 50, 100
<code>\hrule</code> . . . . . 7218, 8883		<code>\inlineex</code> . . . . . 100
<code>\hspace</code> . . . . . 9060		<code>\input</code> . . . . . 27, 80, 135, 40, 82, 85, 318, 1031, 1921, 2012, 8319, 8323, 8326, 8329, 8332, 9090, 9094, 9097, 9100, 9103, 9197, 9270, 9273, 9279, 9283
<code>\HTML</code> . . . . . 15, 26		<code>\inputassignment</code> . . . . . 74, 116
<code>\huge</code> . . . . . 9058		<code>\inputref</code> . . . 27, 28, 82, 83, 102, 1884, 8160, 8161, 8223, 8248, 8540, 9163
hwexam commands:		<code>\inputref*</code> . . . . . 107, 8160
<code>\l_hwexam_includeassignment_</code> <code>keys_tl</code> . . . . . 9121, 9122, 9124, 9161		<code>\inputreffalse</code> . . . . . 1884, 1893
<code>\hwexam_kw_*</code> . . . . . 9088		<code>\inputreftrue</code> . . . . . 1891, 1928
<code>\c_hwexam_multiple_bool</code> . . . . . 9081		<code>\insertframenum</code> . . . . . 8208, 8210
<code>\hwexamheader</code> . . . . . 9197, 9236		<code>\insertshortauthor</code> . . . . . 8201
<code>\hwexamminutes</code> . . . . . 9199		<code>\insertshortdate</code> . . . . . 8212
<code>\hyperlink</code> . . . . . 1833, 1834		<code>\insertshorttitle</code> . . . . . 8204
<code>\hypertarget</code> . . . . . 1813, 1814		<code>\inset</code> . . . . . 45
<b>I</b>		int commands:
<code>\id</code> . . . . . 123		<code>\int_case:nn</code> . . . . . 1369
<code>\if</code> . . . . . 2210, 5895		<code>\int_case:nnTF</code> . . . . . 1327, 1396, 1429
<code>\IfBooleanTF</code> . . . . . 3724, 4163, 4484, 5458, 7335, 7442, 7465, 8142		<code>\int_compare:nNnTF</code> . . . . . 249, 2026, 2098, 3549, 4201, 4236, 4746, 5328, 5615, 5626, 5675, 5881, 6371, 6381, 6473, 6856, 7164, 9177
<code>\ifcsname</code> . . . . . 296, 8256		<code>\int_compare_p:nNn</code> . . . . . . . . . . 7134, 7146, 7158, 7175, 7187
<code>\ifdefempty</code> . . . . . 8247		<code>\int_decr:N</code> . . . . . 7165, 7193
<code>\iffalse</code> . . . . . 7539		<code>\int_eval:n</code> . . . . . 266, 268, 270, 272, 274, 5484, 5490, 5880, 6170, 6177, 6304, 8501, 8504, 9130, 9173, 9174, 9191, 9224, 9231
<code>\IfFileExists</code> . . . . . 6, 20, 1722, 1908, 1916, 1989, 1999, 3237, 3242, 3262		<code>\int_gincr:N</code> . . . . . 1510, 5469, 8499, 8557, 8621, 8670, 8721, 9171
<code>\IfInputref</code> . . . . . 28, 83, 1966, 8245		<code>\int_gset:Nn</code> . . . . . 5433
<code>\ifinputref</code> . . . . . 28, 83, 1884, 1973		<code>\int_gzero:N</code> . . . . . 5418, 8422
<code>\ifintest</code> . . . . . 8309		<code>\int_incr:N</code> . . . . . 1319, 1329, 1333, 1357, 1371, 1374, 1377, 1398, 1402,
<code>\ifmmode</code> . . . . . 6078		
<code>\ifnotes</code> . . . . . 106, 7822, 7954		
<code>\ifSGvar</code> . . . . . 109, 8264		

1433, 1439, 3873, 3874, 3875, 3876,  
4662, 4745, 5581, 7139, 7151, 7162,  
7179, 7191, 7607, 7881, 8782, 9178

`\int_new:N` .....  
1277, 1506, 3829, 4017, 4732, 5411,  
5565, 5869, 8467, 8544, 8716, 9169

`\int_set:Nn` 1413, 1414, 1415, 1416,  
1417, 1418, 1420, 3045, 3838, 3842,  
3846, 3850, 3854, 3858, 3862, 3866,  
3870, 3945, 4024, 4065, 4097, 4635,  
4740, 4747, 4785, 4909, 5541, 5870,  
5903, 7132, 7144, 7156, 7173, 7185

`\int_step_function:nN` ... 5147, 5347

`\int_step_inline:nn` .....  
..... 3887, 4268, 4275, 4295, 4556

`\int_use:N` .. 228, 1303, 1383, 1423,  
1443, 1511, 3375, 3763, 4345, 4463,  
4635, 4814, 4827, 5009, 5022, 5257,  
5433, 5470, 5602, 5639, 5724, 5867,  
5868, 6847, 7610, 7671, 7874, 8515,  
8559, 8623, 8672, 8723, 8784, 9190

`\int_zero:N` .....  
.. 3832, 3833, 4617, 4743, 5571, 7600

`\c_max_int` ..... 5867, 5868

`\l_tmpa_int` .....  
4617, 4620, 4635, 4662, 4743, 4745,  
4746, 4747, 4751, 7132, 7135, 7138,  
7139, 7144, 7147, 7150, 7151, 7156,  
7159, 7162, 7164, 7165, 7167, 7168,  
7173, 7176, 7179, 7181, 7185, 7188,  
7191, 7193, 7194, 7600, 7607, 7610

intarray commands:  
`\intarray_gset:Nnn` .....  
..... 7167, 7181, 7194, 7269

`\intarray_gzero:N` ..... 7268

`\intarray_item:Nn` .... 7135, 7138,  
7147, 7150, 7159, 7168, 7176, 7188

`\intarray_new:Nn` ..... 7130

`\interpretmod` ..... 3627, 3629, 3631

`\interpretmodule` ..... 3514, 3516

`\intestfalse` ..... 8313

`\intesttrue` ..... 8311

invokation commands:  
`\invokation_macro` ..... 123, 125, 131

ior commands:  
`\ior_close:N` .... 659, 665, 1059, 1190

`\ior_map_inline:Nn` ..... 1175

`\ior_new:N` ..... 1171

`\ior_open:Nn` ..... 653, 661, 1173

`\ior_open:NnTF` ..... 1053

`\ior_str_get:NN` ..... 1056

`\ior_str_map_inline:Nn` .... 655, 662

`\g_tmpa_ior` ..... 653, 655,  
659, 661, 662, 665, 1053, 1056, 1059

ior commands:  
`\ior_close:N` ..... 648, 658, 1498

`\ior_new:N` ..... 617, 1496

`\ior_now:Nn` .....  
625, 656, 663, 1524, 1795, 7744, 8461

`\ior_open:Nn` ..... 647, 654, 1497

`\g_tmpa_ior` ..... 654, 656, 658

`\isassociative` ..... 49

`\iscommutative` ..... 49

`\item` ..... 7235, 8515, 8791, 8847, 8938

`\itshape` ..... 7228

**J**

`\jobname` ..... 77, 141, 647, 652,  
653, 654, 661, 666, 675, 981, 1497, 1717

`\join` ..... 63

**K**

keys commands:  
`\l_keys_choice_tl` ..... 3695

`\keys_define:nn` ..... 27,  
379, 7759, 7801, 8231, 8280, 9079, 9249

`\l_keys_key_str` 4146, 4149, 6122, 6125

`\l_keys_key_tl` ..... 4147, 6123

`\keys_set:nn` ..... 383, 8240

keyval commands:  
`\keyval_parse:NNn` ..... 6393

**L**

`\label` ..... 84, 121, 132, 1536, 8051

`\labelsep` ..... 8010

`\labelwidth` ..... 8011

`\lambda` ..... 7540, 7551, 7552

`\langle` ..... 7527

`\Large` ..... 7819, 8170, 9141

`\LaTeX` ..... 23

`\latex` ..... 23

`\ldots` ..... 7532, 7577, 7579, 7581

`\leaders` ..... 8172

`\leavevmode` ..... 8066

`\left` ..... 5897

`\leftmargin` ..... 8012

`\let` . 356, 1452, 1453, 1462, 1463, 1725,  
2083, 2165, 2176, 2183, 2194, 2206,  
3313, 3980, 3987, 4492, 4511, 4800,  
4995, 5224, 5942, 5952, 5966, 6028,  
6107, 7586, 7587, 7746, 7747, 8179

`\libinput` ..... 20, 80, 1976

`\libusepackage` ..... 80, 81, 2024, 7950

`\libusetHEME` ..... 7949

`\libusetikzlibrary` ..... 80, 118, 2080

`\linewidth` ..... 8598, 8605,  
8659, 8664, 8708, 8713, 8761, 8766

`\LoadClass` ..... 15, 7787, 7789

<code>\long</code> .....	<a href="#">141</a> , <a href="#">588</a> , <a href="#">8183</a> , <a href="#">8192</a>	<code>mmtinterface</code> (env.) .....	<a href="#">7620</a>
<code>\lstinputlisting</code> .....	<a href="#">83</a> , <a href="#">2059</a>	<code>\MMTrule</code> .....	<a href="#">7596</a>
<code>\lstinputmhlisting</code> .....	<a href="#">83</a> , <a href="#">2035</a>	<code>\mname</code> .....	<a href="#">87</a> , <a href="#">96</a>
<b>M</b>			
<code>\macro</code> .....	<a href="#">135–137</a> , <a href="#">142</a>	mode commands:	
<code>\macroname</code> .....	<a href="#">32</a>	<code>\mode_if_math:TF</code> .....	
<code>\magma</code> .....	<a href="#">53</a>	.. <a href="#">3364</a> , <a href="#">3980</a> , <a href="#">3981</a> , <a href="#">3988</a> , <a href="#">5266</a> , <a href="#">9031</a>	
<code>\maincomp</code> .....	<a href="#">32</a> , <a href="#">33</a> , <a href="#">36</a> , <a href="#">54</a> , <a href="#">90</a> , <a href="#">97</a> , <a href="#">103</a> , <a href="#">3987</a> , <a href="#">4195</a> , <a href="#">4204</a> , <a href="#">4215</a> , <a href="#">4216</a> , <a href="#">4237</a> , <a href="#">4368</a> , <a href="#">4582</a> , <a href="#">5226</a> , <a href="#">5407</a> , <a href="#">5924</a> , <a href="#">6103</a> , <a href="#">6548</a> , <a href="#">6549</a> , <a href="#">6554</a> , <a href="#">6562</a> , <a href="#">6567</a>	<code>\mode_if_vertical:TF</code> .....	<a href="#">3979</a> , <a href="#">3986</a>
<code>\mainmatter</code> .....	<a href="#">1474</a> , <a href="#">1475</a> , <a href="#">1485</a> , <a href="#">1486</a>	<code>\MSC</code> .....	<a href="#">7590</a>
<code>\makeatletter</code> .....	<a href="#">2130</a> , <a href="#">8318</a> , <a href="#">9089</a> , <a href="#">9236</a>	msg commands:	
<code>\makeatother</code> .....	<a href="#">2131</a> , <a href="#">8335</a> , <a href="#">9106</a> , <a href="#">9236</a>	<code>\msg_error:</code> .....	<a href="#">142</a>
<code>\maketitle</code> .....	<a href="#">1269</a> , <a href="#">1270</a>	<code>\msg_error:nn</code> .....	<a href="#">82</a> , <a href="#">1784</a> , <a href="#">7038</a> , <a href="#">7091</a> , <a href="#">8381</a>
<code>\mapsto</code> .....	<a href="#">7549</a> , <a href="#">7550</a>	<code>\msg_error:nnn</code> .....	<a href="#">85</a> , <a href="#">154</a> , <a href="#">193</a> , <a href="#">241</a> , <a href="#">812</a> , <a href="#">948</a> , <a href="#">955</a> , <a href="#">1978</a> , <a href="#">1981</a> , <a href="#">2107</a> , <a href="#">3200</a> , <a href="#">3256</a> , <a href="#">3427</a> , <a href="#">3457</a> , <a href="#">3993</a> , <a href="#">4695</a> , <a href="#">4937</a> , <a href="#">6182</a> , <a href="#">6659</a> , <a href="#">6686</a>
<code>\marginnote</code> .....	<a href="#">8000</a>	<code>\msg_error:nnnn</code> .....	<a href="#">352</a> , <a href="#">830</a> , <a href="#">835</a> , <a href="#">854</a> , <a href="#">889</a> , <a href="#">961</a> , <a href="#">1952</a> , <a href="#">2009</a> , <a href="#">2919</a> , <a href="#">3007</a> , <a href="#">3352</a> , <a href="#">3878</a> , <a href="#">4454</a> , <a href="#">4960</a> , <a href="#">5215</a> , <a href="#">5249</a> , <a href="#">5495</a> , <a href="#">5522</a> , <a href="#">6031</a> , <a href="#">6296</a> , <a href="#">6996</a>
<code>\marginpar</code> .....	<a href="#">8437</a> , <a href="#">8442</a> , <a href="#">8518</a> , <a href="#">8523</a> , <a href="#">8816</a> , <a href="#">8821</a>	<code>\msg_none:nn</code> .....	<a href="#">79</a>
<code>\mathbb</code> .....	<a href="#">7563</a>	<code>\msg_redirect_module:nnn</code> .....	<a href="#">101</a>
<code>\mathbin</code> .....	<a href="#">33</a> , <a href="#">7508</a> , <a href="#">7512</a> , <a href="#">7519</a> , <a href="#">7545</a> , <a href="#">7556</a>	<code>\msg_redirect_name:nnn</code> .....	<a href="#">105</a>
<code>\mathclose</code> .....	<a href="#">33</a> , <a href="#">5909</a> , <a href="#">7510</a> , <a href="#">7517</a> , <a href="#">7521</a> , <a href="#">7527</a> , <a href="#">7529</a> , <a href="#">7537</a> , <a href="#">7544</a> , <a href="#">7556</a> , <a href="#">7560</a>	<code>\msg_set:nnn</code> .....	<a href="#">76</a>
<code>\mathhub</code> .....	<a href="#">78</a> , <a href="#">138</a> , <a href="#">170</a> , <a href="#">30</a> , <a href="#">1042</a>	<code>\msg_warning:nn</code> .....	<a href="#">1072</a>
<code>\mathop</code> .....	<a href="#">33</a> , <a href="#">7547</a> , <a href="#">7558</a>	<code>\msg_warning:nnn</code> .....	<a href="#">1694</a> , <a href="#">1736</a>
<code>\mathopen</code> .....	<a href="#">33</a> , <a href="#">5904</a> , <a href="#">7510</a> , <a href="#">7517</a> , <a href="#">7521</a> , <a href="#">7527</a> , <a href="#">7529</a> , <a href="#">7536</a> , <a href="#">7544</a> , <a href="#">7556</a> , <a href="#">7560</a>	<code>\msg_warning:nnnn</code> .....	<a href="#">434</a> , <a href="#">1728</a>
<code>\mathord</code> .....	<a href="#">33</a>	<code>\mult</code> .....	<a href="#">40</a> , <a href="#">41</a> , <a href="#">91</a> , <a href="#">92</a>
<code>\mathpunct</code> .....	<a href="#">33</a> , <a href="#">4560</a> , <a href="#">5560</a> , <a href="#">7536</a> , <a href="#">7546</a> , <a href="#">7552</a>	<code>\multicolumn</code> .....	<a href="#">9191</a>
<code>\mathrel</code> .....	<a href="#">32</a> , <a href="#">33</a> , <a href="#">7513</a> , <a href="#">7550</a>	<code>\multiple</code> .....	<a href="#">73</a> , <a href="#">115</a>
<code>\mathrm</code> .....	<a href="#">4582</a> , <a href="#">7536</a>	<b>N</b>	
<code>mathstructure</code> (env.) .....	<a href="#">96</a> , <a href="#">6108</a>	<code>nassertion</code> (env.) .....	<a href="#">1</a>
<code>\mathstructure</code> .....	<a href="#">6152</a> , <a href="#">6153</a>	<code>\Nat</code> .....	<a href="#">36</a>
<code>\mathtt</code> .....	<a href="#">7516</a> , <a href="#">7517</a> , <a href="#">7566</a> , <a href="#">7569</a> , <a href="#">7573</a>	<code>ndefinition</code> (env.) .....	<a href="#">1</a>
<code>mcb</code> (env.) .....	<a href="#">1</a> , <a href="#">8824</a>	<code>\neginfpref</code> .....	<a href="#">41</a> , <a href="#">90</a> , <a href="#">91</a> , <a href="#">4246</a> , <a href="#">4249</a> , <a href="#">4258</a> , <a href="#">4261</a> , <a href="#">4274</a> , <a href="#">5086</a> , <a href="#">5322</a> , <a href="#">5333</a> , <a href="#">5867</a>
<code>\mcb</code> .....	<a href="#">8365</a> , <a href="#">8835</a> , <a href="#">8860</a> , <a href="#">8926</a>	<code>\newcommand</code> .....	<a href="#">445</a> , <a href="#">480</a> , <a href="#">1578</a> , <a href="#">1959</a> , <a href="#">1967</a> , <a href="#">1972</a> , <a href="#">2017</a> , <a href="#">2024</a> , <a href="#">2054</a> , <a href="#">2060</a> , <a href="#">2068</a> , <a href="#">2078</a> , <a href="#">2105</a> , <a href="#">7493</a> , <a href="#">7693</a> , <a href="#">7714</a> , <a href="#">7931</a> , <a href="#">8000</a> , <a href="#">8062</a> , <a href="#">8163</a> , <a href="#">8167</a> , <a href="#">8215</a> , <a href="#">8222</a> , <a href="#">8225</a> , <a href="#">8242</a> , <a href="#">8356</a> , <a href="#">8779</a> , <a href="#">8886</a> , <a href="#">8953</a> , <a href="#">8981</a> , <a href="#">8986</a> , <a href="#">8991</a> , <a href="#">8996</a> , <a href="#">9029</a> , <a href="#">9066</a> , <a href="#">9069</a> , <a href="#">9070</a> , <a href="#">9071</a> , <a href="#">9072</a> , <a href="#">9073</a> , <a href="#">9176</a> , <a href="#">9261</a> , <a href="#">9266</a> , <a href="#">9290</a>
<code>\mcc</code> .....	<a href="#">69</a> , <a href="#">112</a> , <a href="#">8841</a> , <a href="#">8881</a>	<code>\newcounter</code> .....	<a href="#">7790</a> , <a href="#">7791</a> , <a href="#">7792</a> , <a href="#">7793</a> , <a href="#">7915</a> , <a href="#">7927</a> , <a href="#">8355</a> , <a href="#">8373</a> , <a href="#">8374</a> , <a href="#">9119</a>
<code>\meaning</code> .....	<a href="#">1024</a> , <a href="#">2396</a> , <a href="#">2659</a> , <a href="#">4241</a> , <a href="#">4526</a> , <a href="#">4611</a> , <a href="#">4637</a> , <a href="#">5164</a> , <a href="#">5169</a> , <a href="#">5372</a> , <a href="#">5382</a> , <a href="#">5383</a>	<code>\NewDocumentCommand</code> .....	<a href="#">448</a> , <a href="#">1542</a> , <a href="#">1774</a> , <a href="#">1779</a> , <a href="#">1826</a> , <a href="#">1898</a> , <a href="#">1946</a> , <a href="#">2489</a> , <a href="#">2568</a> , <a href="#">4949</a> , <a href="#">5457</a> , <a href="#">5936</a> , <a href="#">5944</a> , <a href="#">5955</a> , <a href="#">5977</a> , <a href="#">5988</a> , <a href="#">6001</a> , <a href="#">6911</a> , <a href="#">7001</a> , <a href="#">7006</a> , <a href="#">7012</a> , <a href="#">7020</a> , <a href="#">7029</a> , <a href="#">7072</a> , <a href="#">7082</a> , <a href="#">7100</a> , <a href="#">7426</a> , <a href="#">7462</a> , <a href="#">7477</a> , <a href="#">7596</a> , <a href="#">7657</a> , <a href="#">7949</a> , <a href="#">8141</a> , <a href="#">8536</a> , <a href="#">9159</a>
<code>\medskip</code> .....	<a href="#">8027</a> , <a href="#">8041</a> , <a href="#">8060</a>		
<code>\meet</code> .....	<a href="#">63</a>		
<code>\message</code> .....	<a href="#">26</a> , <a href="#">7780</a> , <a href="#">7783</a> , <a href="#">7940</a>		
<code>\mhframeimage</code> .....	<a href="#">108</a>		
<code>\mhgraphics</code> .....	<a href="#">83</a> , <a href="#">2035</a> , <a href="#">8154</a> , <a href="#">8156</a>		
<code>\mhinput</code> .....	<a href="#">83</a> , <a href="#">1884</a>		
<code>\mhtikzinput</code> .....	<a href="#">118</a> , <a href="#">2035</a>		
<code>\min</code> .....	<a href="#">74</a> , <a href="#">116</a>		
<code>\min</code> .....	<a href="#">8819</a>		
<code>min</code> .....	<a href="#">73</a> , <a href="#">110</a> , <a href="#">115</a>		
<code>\mmlarg</code> .....	<a href="#">337</a>		
<code>\mmlintent</code> .....	<a href="#">337</a>		
<code>\mmtdef</code> .....	<a href="#">7647</a> , <a href="#">7657</a>		
<code>\MMTinclude</code> .....	<a href="#">7591</a>		

<code>\NewDocumentEnvironment</code> .....	<code>\l__notesslides_excursion_id_str</code> .....
483, 1349, 1358, 7482, 7620	8232, 8238
<code>\newenvironment</code> .....	<code>\l__notesslides_excursion_intro_-</code> tl .....
1471, 1482, 7196, 7403, 8114, 8116, 9219	8233, 8237, 8247, 8249
<code>\newif</code> 297, 320, 1884, 7822, 8301, 8309, 9226	<code>\l__notesslides_excursion_-</code> mhrepos_str .....
<code>\newlabel</code> .....	8234, 8239, 8248
7746, 7747	<code>\l__notesslides_frame_allowdisplaybreaks_-</code> bool .....
<code>\newlength</code> .....	7969, 7980
7952, 7953, 7956	<code>\l__notesslides_frame_allowframebreaks_-</code> bool .....
<code>\newline</code> .....	7968, 7977
8598, 8605, 8659, 8664, 8708, 8713, 8761, 8766	<code>\__notesslides_frame_box_begin:</code> . .....
<code>\newpage</code> .....	8020, 8031, 8054
8048, 9073, 9238	<code>\__notesslides_frame_box_end:</code> . .....
<code>\newsavebox</code> .....	8025, 8040, 8056
8084	<code>\l__notesslides_frame_fragile_-</code> bool .....
<code>nexample (env.)</code> .....	7970, 7983
<u>1</u>	<code>\l__notesslides_frame_label_str</code> . .....
<code>\ninputref</code> .....	7967, 7975, 8050, 8051
8161, 8163, 8167	<code>\l__notesslides_frame_shrink_-</code> bool .....
<code>\nobreak</code> .....	7971, 7986
7224	<code>\l__notesslides_frame_squeeze_-</code> bool .....
<code>\noindent</code> .....	7972, 7989
1304, 1909, 1917, 6895, 7214, 8027, 8041, 8064, 8082, 8434, 8599, 8660, 8709, 8762	<code>\l__notesslides_frame_t_bool</code> . .....
<code>\nointerlineskip</code> .....	7973, 7992
8174	<code>\__notesslides_inputref:</code> .....
<code>\notation</code> .. <i>32, 33, 41, 88, 90, 95, 126, 129, 135, 2987, 4153, 7508, 7511, 7512, 7513, 7526, 7528, 7546, 7547, 7551, 7555, 7557, 7558, 7563, 7566</i>	8160, 8161, 8164
<code>note (env.)</code> .....	<code>\__notesslides_notes_env:nnnn</code> . .....
<u>1</u>	8112, 8132, 8133, 8134, 8135, 8136, 8137
<code>notes</code> .....	<code>\l__notesslides_num</code> .....
73, 105, 110, 115	7840, 7842, 7845, 7847, 7850, 7851, 7854, 7855, 7858, 7859, 7862, 7863, 7866, 7867, 7874
<code>\notesfalse</code> .....	<code>\__notesslides_setup_itemize:</code> . .....
7834	7996, 8053
notesslides commands:	<code>\l__notesslides_tmp</code> .....
<code>\c__notesslides_notes_bool</code> .....	<code>\g__notesslides_variables_prop</code> . .....
7761, 7762, 7775, 7778, 7786, 7802, 7803, 7815, 7823, 7946, 8107, 8113, 8147, 8162, 8181, 8216, 8226	8257, 8259, 8262, 8265
<code>\c__notesslides_sectocframes_bool</code> .....	<code>\notesslidesfont</code> .....
7804, 7902	8023, 8038, 8179
<code>\c__notesslides_topsect_str</code> 7805, 7828, 7831, 7887, 7890, 7891, 7894	<code>\notesslidesfooter</code> .....
notesslides internal commands:	8027, 8041, 8177
<code>\c__notesslides_class_str</code> .....	<code>\notesslidestitleemph</code> .....
7757, 7760, 7766, 7787	8060, 8169
<code>\__notesslides_define_chapter:</code> . .....	<code>\notesttrue</code> .....
7892, 7895, 7907	7824
<code>\__notesslides_define_part:</code> . .....	<code>nparagraph (env.)</code> .....
7896, 7919	<u>1</u> , <u>1</u>
<code>\__notesslides_do_label:n</code> 7837, 7873	<code>nsproof (env.)</code> .....
<code>\__notesslides_do_sectocframes:</code> . .....	<u>1</u>
7836, 7903	<code>\null</code> .....
<code>\__notesslides_do_yes_param:Nn</code> . .....	7224
7958, 7977, 7980, 7983, 7986, 7989, 7992	<code>\num</code> .....
<code>\c__notesslides_docopt_str</code> . . . 7763	143
<code>\c__notesslides_document_str</code> . . . .....	<code>\number</code> .....
7930, 7933	73, 116
<code>\__notesslides_eat:</code> . 8117, 8121, 8124	<code>\numberline</code> .....
<code>\__notesslides_excursion_args:n</code> . .....	7872
8236, 8243	<code>\numberproblemsin</code> .....
	8355
	<b>O</b>
	<code>\objective</code> .....
	7714
	<code>\OMDoc</code> .....
	26
	<code>\omdoc</code> .....
	15

<code>\oplus</code>	7518, 7519	<code>problem (env.)</code>	1
<b>P</b>			
<code>\PackageError</code>	8266	<code>problem commands:</code>	
<code>\pagebreak</code>	8449, 9157	<code>\g_problem_id_counter</code>	8544, 8557, 8559, 8621, 8623, 8670, 8672, 8721, 8723
<code>\pagenumbering</code>	1458, 1468, 1479, 1490	<code>\l_problem_inputproblem_keys_tl</code>	8383, 8384, 8386, 8538
<code>\par</code>	142, 360, 1302, 1306, 7197, 7224, 7267, 7324, 8027, 8041, 8064, 8069, 8077, 8082, 8089, 8099, 8434, 8445, 8448, 8473, 8598, 8605, 8659, 8664, 8708, 8713, 8761, 8766, 8805, 8808, 8855, 8946, 9140, 9145, 9155, 9157	<code>\problem_mcc_box_tl</code>	8827, 8847, 8857, 8881
<code>\paragraph</code>	27, 82	<code>\problem_scc_box_tl</code>	8913, 8924, 8938, 8948
<code>\parsep</code>	7233, 8513, 8789, 8845, 8936	<code>problem@kw@points commands:</code>	
<code>\part</code>	27, 82, 7923, 7924	<code>\problem@kw@points:</code>	8806
<code>\partname</code>	7842, 7920, 7921	<code>\problemheader</code>	8434, 8452
<code>\parttitlename</code>	7842	<code>problems internal commands:</code>	
<code>\PassOptionsToClass</code>	7765, 7766	<code>\__problems_activate_macros:</code>	8363, 8426, 8475
<code>\PassOptionsToPackage</code>	10, 7767, 7768, 7779, 7782, 7807, 7808, 7825, 8298, 9083	<code>\l__problems_anscls_int</code>	8716, 8782, 8784
<code>\pause</code>	8062	<code>\c__problems_boxed_bool</code>	8294
<code>\PDF</code>	15, 26	<code>\__problems_do_yes_param:Nn</code>	8861
<code>\pdfbookmark</code>	7874	<code>\__problems_exnote_start:n</code>	8667, 8685, 8688
<code>\pdfdest</code>	1538	<code>\l__problems_fillin_solution_tl</code>	9002, 9023, 9043, 9048, 9052
<code>peek commands:</code>		<code>\__problems_fillinsol:nn</code>	9032, 9034, 9038
<code>\peek_charcode:NTF</code>	3582, 5080, 5093, 5156, 5158, 5280, 5287, 6318, 6324, 6465, 6476	<code>\__problems_gnote_start:n</code>	8718, 8737, 8740
<code>\peek_charcode_remove:NTF</code>	5079, 5155, 5271, 5276, 5278, 5285, 5358, 5746, 6323, 6573	<code>\c__problems_gnotes_bool</code>	8284, 8739, 8751
<code>\phantom</code>	9058	<code>\l__problems_has_min_bool</code>	8377, 8420, 8421, 8503, 8522
<code>\Pi</code>	7540	<code>\l__problems_has_pts_bool</code>	8376, 8417, 8418, 8500, 8517
<code>\plus</code>	39–41, 44, 90–92	<code>\__problems_hint_start:n</code>	8618, 8636, 8639
<code>\precondition</code>	7693	<code>\c__problems_hints_bool</code>	8286, 8638, 8649
<code>\prematrestop</code>	108, 7930	<code>\l__problems_in_problem_bool</code>	8375, 8378, 8380, 8413, 8471, 8476
<code>\premise</code>	52, 101, 6963, 7100, 7107	<code>\__problems_mccline:n</code>	8842, 8857, 8899
<code>prg commands:</code>		<code>\c__problems_min_bool</code>	8292, 8440, 8521, 8820
<code>\prg_new_conditional:Nnn</code>	232, 282, 534, 539, 748, 758, 2137, 2547, 2551, 3655, 3666, 3670, 5501, 5614, 5625	<code>\l__problems_min_tl</code>	8415, 8419, 8429, 8441, 8442, 8462, 8478, 8504
<code>\prg_new_protected_conditional:Nnn</code>	768	<code>\c__problems_notes_bool</code>	8282, 8687, 8698
<code>\prg_return_false:</code>	234, 284, 537, 542, 749, 753, 759, 761, 784, 788, 2138, 2549, 2553, 3656, 3671, 5519, 5523, 5620, 5621, 5622, 5628, 5629	<code>\__problems_parse_fillin_- arg:nnnn</code>	9006, 9007, 9008, 9011
<code>\prg_return_true:</code>	234, 284, 537, 542, 751, 753, 761, 788, 2138, 2549, 2553, 3667, 5517, 5620, 5628	<code>\l__problems_path_seq</code>	8395, 8396, 8481, 8482, 8486, 8487
<code>\printexcursion</code>	109	<code>\l__problems_prob_imports_tl</code>	8352
<code>\printexcursions</code>	8214, 8223, 8244, 8252	<code>\l__problems_prob_refnum_int</code>	8353

<code>\c_problems_pts_bool</code> .....	<code>\prop_put:Nnn</code> ... 1182, 1183, 1184,
..... 8290, 8435, 8516, 8815	1185, 1186, 2899, 2903, 3371, 3405,
<code>\l_problems_pts_tl</code> .. 8414, 8416,	4811, 5006, 6690, 6713, 6727, 9179
8428, 8436, 8437, 8462, 8477, 8501	<code>\prop_remove:Nn</code> ..... 3467
<code>\_problems_record_problem:</code> ....	<code>\prop_set_eq:NN</code> 1089, 1206, 1210, 1612
..... 8430, 8460	<code>\prop_to_keyval:N</code> .....
<code>\__problems_sccline:n</code> 8933, 8948, 8966	... 1192, 1195, 1220, 2397, 2398,
<code>\__problems_solution_start:n</code> ...	2399, 2406, 2410, 2926, 2929, 5430
..... 8554, 8573, 8576	<code>\protect</code> ..... 7872, 8268
<code>\c_problems_solutions_bool</code> ....	<code>\protected</code> ..... 123,
..... 8288, 8304	141, 358, 572, 573, 1270, 5953, 5967
<code>\g_problems_subproblem_int</code> ....	<code>\providecommand</code> .. 2040, 2047, 7938, 9260
..... 8422, 8467, 8499, 8515	<code>\ProvidesExplClass</code> ..... 5, 7754
<code>\c_problems_test_bool</code> .....	<code>\ProvidesExplPackage</code> .....
.. 8296, 8310, 8449, 9067, 9069, 9073	..... 19, 7798, 8277, 9076, 9246
<code>\ProcessKeysOptions</code> .....	<code>\pts</code> ..... 8814
..... 39, 7771, 7811, 8303, 9086, 9255	<code>pts</code> ..... 73, 110, 115
<code>\ProcessOptions</code> ..... 11	
<code>\prod</code> ..... 7547, 7558	
<code>\prop</code> ..... 42	
prop commands:	
<code>\prop_clear:N</code> . 1174, 2881, 2882, 6664	
<code>\prop_const_from_keyval:Nn</code> . 110, 122	
<code>\prop_gclear:N</code> .....	
..... 2320, 2321, 2322, 2432, 5416	
<code>\prop_get:NnN</code> ..... 1208	
<code>\prop_get:NnNTF</code> .....	
..... 189, 903, 904, 1800, 1904,	
1905, 1980, 2935, 2956, 5502, 8265	
<code>\prop_gput:Nnn</code> .....	
2460, 2581, 2593, 2595, 2628, 5445,	
5481, 5487, 5505, 7068, 8259, 9172	
<code>\prop_gset_eq:NN</code> .... 1191, 1222, 1225	
<code>\prop_gset_from_keyval:Nn</code> .....	
..... 1194, 1219, 2425, 2427, 5429	
<code>\prop_if_exist:NTF</code> .....	
..... 901, 1100, 1117, 1603,	
1697, 1799, 1864, 1977, 3080, 5428	
<code>\prop_if_in:NnTF</code> .....	
138, 173, 805, 6624, 6681, 6711, 6725	
<code>\prop_item:Nn</code> .....	
..... 139, 921, 1101, 1214, 1621,	
1700, 1866, 1869, 1878, 2042, 2056,	
2071, 2861, 3017, 3022, 3083, 3108,	
3119, 3134, 6616, 6656, 6684, 8262	
<code>\prop_map_break:</code> ..... 2621	
<code>\prop_map_break:n</code> 2640, 2676, 2679,	
2757, 3041, 4062, 4925, 4928, 7055	
<code>\prop_map_function:NN</code> ... 2418, 2948	
<code>\prop_map_inline:Nn</code> .. 2428, 2605,	
2638, 2682, 2700, 2703, 2725, 2728,	
2766, 2912, 2931, 2971, 2999, 3003,	
4078, 4918, 6278, 6504, 7052, 9184	
<code>\prop_new:N</code> ..... 8257, 9166	
	<code>\quad</code> ..... 7851, 7855, 7859,
	7863, 7867, 9030, 9036, 9148, 9151
	quark commands:
	<code>\quark_new:N</code> ..... 2161, 2444
	quark internal commands:
	<code>\q__stex_smsmode_break</code> .....
	..... 2161, 2164, 2224
	<b>R</b>
	<code>\rangle</code> ..... 7527
	<code>\realization</code> ..... 3538, 3540
	<code>\realize</code> ..... 63, 64, 3648, 3650, 3652
	<code>\ref</code> ..... 84, 1644
	<code>\refstepcounter</code> ..... 8391, 9132
	<code>\relax</code> .... 359, 640, 641, 1453, 1463,
	1538, 1695, 2082, 2168, 2206, 2210,
	2435, 2436, 3980, 3987, 8148, 8150
	<code>\renamedecl</code> .... 64, 129, 2993, 3391, 3475
	<code>\renewcommand</code> ..... 7947, 8048, 8058
	<code>\renewenvironment</code> .....
	.. 8001, 8045, 8063, 8085, 8108, 8110
	repo commands:
	<code>\repo_prop</code> ..... 138, 139
	<code>\reqpts</code> ..... 74, 116
	<code>\requiremodule</code> ..... 93, 94, 2989, 3340
	<code>\RequirePackage</code> .....
	.. 4, 13, 18, 23, 24, 147, 149, 198,
	201, 5893, 7755, 7773, 7795, 7799,
	7813, 7826, 7827, 8139, 8278, 8316,
	9077, 9087, 9247, 9258, 9263, 9264
	<code>\resizebox</code> ..... 9189, 9272, 9278, 9282
	<code>\rhd</code> ..... 8003, 8006
	<code>\right</code> ..... 5898
	<code>\Rightarrow</code> ..... 7459
	<code>\rightmargin</code> . 7234, 8514, 8790, 8846, 8937

<code>\rule</code> .....	8598, 8605, 8659, 8664, 8708, 8713, 8761, 8766	<code>\seq_map_function:NN</code> .	723, 727, 6415
rustex commands:		<code>\seq_map_inline:Nn</code> .....	265, 918, 922, 1597, 1632, 1954, 2011, 2187, 2681, 4076, 4717, 4744, 5709, 6208, 6255, 6264, 6792, 6930, 6957, 7275, 7330, 7606
<code>\rustex_direct_HTML:n</code> .....	8070, 8078, 8090, 8100	<code>\seq_new:N</code> .....	226, 1505, 1551, 2118, 2141, 2291
<code>\rustex_if:TF</code> .....	8068, 8069, 8076, 8077, 8088, 8089, 8098, 8099	<code>\seq_pop:NN</code> .....	720
<code>\rustexBREAK</code> .....	2134	<code>\seq_pop_left:NN</code> .....	169, 703, 781, 782, 833, 838, 844, 849, 919, 923, 925, 1994, 3551, 3553, 3560, 3564
<b>S</b>		<code>\seq_pop_left:NNTF</code> .....	1178, 4286, 4288, 4296
<code>sassertion (env.)</code> .....	99	<code>\seq_pop_right:NN</code> .....	161, 171, 738, 793, 795, 800, 802, 804, 1146, 2366, 3076, 4038, 4575, 4577, 6406
<code>scb (env.)</code> .....	8912	<code>\seq_push:Nn</code> .....	726
<code>\scb</code> .....	8366, 8836, 8927, 8951	<code>\seq_put_left:Nn</code> .....	704, 724, 2699, 2724, 6712, 6726
<code>\scc</code> .....	8932, 8952	<code>\seq_put_right:Nn</code> .....	165, 191, 229, 741, 796, 806, 809, 981, 1991, 1995, 2001, 2325, 2355, 2368, 2660, 2765, 2847, 2978, 4269, 4276, 4297, 4299, 4557, 4719, 4721, 5601, 5637, 5673, 5680, 5705, 5711, 5715, 5722, 6145, 6200, 6204, 6250, 6682, 6789, 6825, 6839
<code>\scriptsize</code> .....	8000	<code>\seq_reverse:N</code> .....	6407
<code>\scriptstyle</code> .....	8006	<code>\seq_set_eq:NN</code> .	729, 769, 770, 792, 799, 917, 980, 1984, 4726, 4754, 5720
<code>sdefinition (env.)</code> .....	99	<code>\seq_set_split:Nnn</code> .....	160, 684, 794, 801, 832, 837, 1138, 1177, 1985, 2365, 3075, 3548, 3558, 4037, 4285, 4289, 4574, 4576, 6405, 7598
<code>\second</code> .....	137	<code>\seq_use:Nn</code> .....	196, 199, 202, 766, 796, 809, 837, 1180, 2370, 3077, 3556, 3562, 3566, 4043, 4355, 4560, 4703, 4727, 5436, 5559, 6872, 7205, 7256, 7412
<code>\section</code> .....	26, 27, 82	<code>\l_tmpa_seq</code> .....	160, 161, 165, 169, 170, 171, 183, 191, 196, 199, 202, 4555, 4557, 4560, 4574, 4575, 4576, 4577, 4716, 4719, 4721, 4726, 4754, 6403, 6405, 6406, 6407, 6415, 7597, 7598, 7605, 7606
<code>\sectiontitleemph</code> .....	7818, 7877	<code>\seqmap</code> .....	96, 5172, 5627
<code>sectocframes</code> .....	105	<code>\setbox</code> .....	135, 8110, 8125, 8580, 8641, 8690, 8742
<code>\selectlanguage</code> .....	140, 142	<code>\setcounter</code> .....	9130, 9133
seq commands:		<code>\setkeys</code> . . . . .	2044, 2058, 2073, 8152, 9267
<code>\seq_clear:N</code> .....	183, 683, 721, 1074, 1983, 2178, 2689, 2714, 2739, 2844, 2883, 4199, 4555, 4716, 5570, 5634, 5699, 5708, 6144, 6195, 6245, 6403, 6665, 6666, 6786, 6822, 7597	<code>\setlength</code> .....	7232, 7233, 7234, 7952, 7953, 7957, 8010, 8011, 8012,
<code>\seq_count:N</code> . . . . .	2026, 2098, 3549, 6856		
<code>\seq_gclear:N</code> .....	2172, 2173, 5417		
<code>\seq_gclear_new:N</code> .....	984, 985		
<code>\seq_get:NN</code> .....	1015		
<code>\seq_get_right:NN</code>	159, 8396, 8482, 8487		
<code>\seq_gpop:NN</code> .....	1010		
<code>\seq_gpush:Nn</code> .....	998		
<code>\seq_gput_left:Nn</code> .....	1547, 1552		
<code>\seq_gput_right:Nn</code> .....	1550, 2128, 2147, 5446		
<code>\seq_gset_eq:NN</code> .....	997, 1013, 1017, 2871, 2872, 2873, 7736		
<code>\seq_gset_split:Nnn</code> .....	5435		
<code>\seq_if_empty:NNTF</code> . . . . .	170, 719, 737, 749, 759, 788, 834, 840, 845, 850, 1009, 1012, 1118, 1951, 2008, 4039, 6871, 7204, 7255, 7411, 7605, 7735		
<code>\seq_if_empty_p:N</code> . . . . .	775, 776, 1987		
<code>\seq_if_exist:NNTF</code> .....	1592, 5434		
<code>\seq_if_in:NnNTF</code> . . . . .	1545, 1546, 1659, 1661, 1990, 2000, 2272, 2281, 2658, 2698, 2723, 2764, 6516, 6627, 6750		
<code>\seq_item:Nn</code> .....	750, 760, 2027, 2099, 4312, 4318, 5503, 6857		
<code>\seq_map_break:</code> . . . . .	1594, 1598, 2676		
<code>\seq_map_break:n</code> . . . . .	1598, 2679, 4062		



	8512, 8513, 8514, 8788, 8789, 8790, 8844, 8845, 8846, 8935, 8936, 8937		
<code>\setlicensing</code>	107	<code>\sproofend</code>	7216, 7304, 7321
<code>\setmetatheory</code>	2470	<code>\square</code>	7217, 8882
<code>\setnotation</code>	33, 91, 4429	<code>\sr</code>	20, 24, 88, 5924
<code>\setsectionlevel</code>	27, 82, 121, 1411, 7829, 7831, 7888, 7890	<code>\sref</code>	84, 85, 99, 1556, 8218
<code>\setSGvar</code>	109, 8258, 8268	<code>\sreflabel</code>	84, 86, 121, 1505
<code>\setslidelogo</code>	107	<code>\srefsetin</code>	84, 1578
<code>\setsource</code>	107	<code>\srefsym</code>	89, 1826
<code>sexample (env.)</code>	99	<code>\srefsymuri</code>	89, 1857
<code>\sf</code>	8170	<code>\startsolutions</code>	111, 8609
<code>\sffamily</code>	8179	<code>\stepcounter</code>	1392, 7871, 8047
<code>sfragment (env.)</code>	82	<code>\sTeX</code>	14, 15, 81
<code>sfragment</code> commands:		<code>\stex</code>	15, 24, 81
<code>\sfragment_do_level:nn</code>	1300, 1312, 1328, 1332, 1336, 1337, 1338, 1339, 1340, 7870	<code>stex</code> commands:	
<code>\sfragment_end:</code>	1308, 1322, 1354	<code>\l_stex_aB_args_seq</code>	4703, 4717, 4726, 4727, 4744, 4754, 5559, 5570, 5601, 5634, 5637, 5673, 5680, 5699, 5705, 5709, 5720, 5722
<code>\skipfragment</code>	82, 1395, 1399, 1403	<code>\stex_activate_module:n</code>	122, 2336, 2385, 2657, 2657, 2669, 2839, 3182, 3280, 3323, 3523, 6216, 6272, 6797, 7643
<code>\slideframewidth</code>	7956, 7957, 8033, 8148	<code>\_stex_add_definiens:nn</code>	2995, 7042, 7050, 7277, 7332
<code>\slideheight</code>	7953	<code>\_stex_add_definiens_inner:nnnnnnnn</code>	7055, 7066
<code>slides</code>	105	<code>\stex_add_module_notation:nnnnn</code>	122
<code>\slidewidth</code>	7952, 8036, 8066, 8148, 8150, 8154	<code>\l_stex_all_modules_seq</code>	122, 2178, 2291, 2325, 2355, 2658, 2660, 2681, 4076
<code>\smacro</code>	92, 93	<code>\l_stex_allow_semantic_bool</code>	4956, 5098, 5176, 5177, 5180, 5206, 5240, 5406, 5415, 5531, 5542, 5640, 5725, 5783, 5794, 5827, 5856, 6017, 6098, 6509, 6540, 6557, 6602, 6983
<code>\small</code>	7228	<code>\stex_annotate:nn</code>	339, 342, 1283, 1292, 1304, 1748, 1968, 1969, 3364, 3800, 3803, 3810, 3815, 3817, 4358, 4364, 4373, 4384, 4395, 4396, 4399, 4400, 4404, 4860, 4863, 4870, 4874, 4877, 5053, 5056, 5063, 5067, 5070, 5101, 5547, 5554, 5648, 5757, 5770, 5787, 5812, 5820, 5841, 5849, 6043, 6429, 6437, 6442, 6650, 6896, 6917, 6994, 7060, 7093, 7105, 7202, 7244, 7247, 7289, 7343, 7352, 7356, 7409, 7478, 7494, 7603, 7608, 8208, 8793, 8816, 8821, 8901, 8904, 8968, 8971, 9041
<code>\smallskip</code>	7224, 8437, 8442, 8518, 8523, 8598, 8659, 8708, 8761, 9145	<code>\stex_annotate:nnn</code>	140, 141
<code>smodule (env.)</code>	86, 2515	<code>\_stex_annotate_force_break:n</code>	328, 329, 335, 1305, 3363, 3797, 3815, 4857, 4875, 5051, 5068, 5555, 5653, 5762, 5775,
<code>\smodule</code>	2990		
<code>\Sn</code>	20, 89, 5924		
<code>\sn</code>	20, 24, 89, 5924		
<code>\Sns</code>	20, 89, 5924		
<code>\sns</code>	20, 89, 5924		
<code>\solution</code>	73		
<code>solution (env.)</code>	1, 8544		
<code>\solution</code>	8364, 8608, 8837, 8928		
<code>solutions</code>	73, 110, 115		
<code>\solutionsfalse</code>	8307, 8613		
<code>\solutionstrue</code>	8305, 8610		
<code>\source</code>	137		
<code>sparagraph (env.)</code>	99		
<code>\spfblock</code>	7287, 7487		
<code>\spfjust</code>	7288, 7493, 7496		
<code>spfsketch</code>	7196		
<code>\spfsketchenvautorefname</code>	7214		
<code>\spfstep</code>	7282, 7389		
<code>\spfstepautorefname</code>	7323, 7404		
<code>\spfstepenvautorefname</code>	7404		
<code>sproblem (env.)</code>	8363		
<code>\sproblemautorefname</code>	8453		
<code>sproof (env.)</code>	101, 7230		
<code>\sproofautorefname</code>	7302		



5792, 5817, 5825, 5846, 5854, 6441,  
6896, 6918, 7061, 7094, 7604, 9041  
\stex\_annotate\_invisible:n .....  
..... 141, 330,  
332, 1362, 1384, 2532, 2789, 2833,  
5203, 5459, 7242, 7601, 7638, 8494  
\stex\_annotate\_invisible:nn 1256,  
1423, 1431, 1437, 1443, 1909, 1917,  
3061, 3285, 3326, 3362, 3396, 3437,  
3447, 3779, 4351, 4838, 5035, 5793,  
5826, 5855, 6210, 6266, 6794, 7076,  
7592, 7639, 7707, 7728, 8406, 9015  
\stex\_annotate\_invisible:nnn .. 141  
\l\_stex\_argnames\_seq ..... 124  
\stex\_args\_end: ..... 5420,  
5444, 5447, 5969, 5973, 6023, 6989  
\stex\_assign\_do:n .....  
..... 2854, 3344, 3349, 3577  
\l\_stex\_assoc\_args\_count .....  
..... 124, 3829, 3833, 3875, 3876  
\l\_stex\_brackets\_dones\_bool ....  
.. 4624, 5873, 5876, 5877, 5901, 5902  
\stex\_capitalize:n .....  
..... 1294, 5960, 5963, 6006, 7023  
\stex\_check\_term:n .....  
..... 124, 125, 2177, 3356,  
3675, 3676, 3684, 3895, 4331, 7500  
\c\_stex\_check\_terms\_bool .....  
..... 33, 3662, 3665, 7503  
\stex\_close\_module: ... 122, 2392,  
2392, 2541, 2796, 7584, 7651, 7653  
\l\_stex\_current\_archive ..... 1086  
\l\_stex\_current\_archive\_prop ...  
..... 134, 139, 921, 931,  
936, 1089, 1100, 1101, 1110, 1199,  
1603, 1604, 1697, 1700, 1799, 1800,  
1864, 1866, 1869, 1904, 1905, 1977,  
1980, 2042, 2056, 2071, 3080, 3083  
\l\_stex\_current\_args\_tl .....  
131, 4546, 4550, 5229, 5419, 5420, 5423  
\l\_stex\_current\_arity\_str .....  
..... 131, 4556, 5118, 5145,  
5147, 5228, 5328, 5346, 5347, 5378  
\l\_stex\_current\_doc\_uri 139, 965,  
967, 968, 1515, 1520, 1522, 1526,  
1592, 1594, 1634, 1651, 1652, 6875  
\l\_stex\_current\_domain\_str .....  
..... 129, 2803, 2806, 2808,  
2816, 2821, 2830, 2839, 2870, 2885,  
2897, 2934, 2946, 3414, 3462, 3483,  
3503, 3523, 3527, 3600, 3620, 3641  
\g\_stex\_current\_file .....  
. 137–139, 159, 930, 932, 937, 985,  
988, 990, 992, 993, 994, 997, 998,  
1013, 1016, 1017, 1607, 1705, 2344,  
3090, 3096, 3097, 3145, 3164, 3208,  
3234, 7735, 7736, 8395, 8481, 8486  
\l\_stex\_current\_language\_str 139,  
140, 135, 137, 172, 177, 2526, 3195,  
3198, 3222, 3225, 7632, 8400, 8491  
\l\_stex\_current\_module ..... 123, 125  
\l\_stex\_current\_module\_str .....  
. 122, 143, 2179, 2290, 2307, 2324,  
2325, 2340, 2353, 2354, 2355, 2356,  
2395, 2396, 2397, 2398, 2399, 2405,  
2407, 2411, 2416, 2418, 2525, 2535,  
2548, 2556, 2560, 2581, 2587, 2593,  
2595, 2605, 2612, 2617, 2626, 2628,  
2638, 2662, 2665, 2784, 2795, 2829,  
2840, 2937, 2940, 3369, 3403, 3430,  
3439, 3449, 3733, 3771, 3780, 3879,  
3940, 3949, 3958, 3964, 4503, 4522,  
4523, 6101, 6139, 6144, 6145, 6165,  
6257, 6279, 6840, 6863, 7051, 7052,  
7053, 7054, 7067, 7068, 7622, 7623,  
7626, 7631, 7650, 7652, 7665, 7679  
\l\_stex\_current\_ns\_uri ..... 139,  
971, 972, 2362, 2505, 3146, 3151,  
3152, 3165, 3170, 3171, 7504, 7585  
\l\_stex\_current\_redo\_tl .....  
..... 5210, 5223, 5234, 5237,  
5244, 5374, 6374, 6412, 6447, 6597  
\l\_stex\_current\_return\_tl .....  
131, 3987, 5230, 5296, 5308, 5329, 5379  
\stex\_current\_section\_level ....  
..... 1279, 1285, 1294, 1320, 7882  
\l\_stex\_current\_symbol\_str .....  
..... 131, 4183, 4213,  
4332, 4372, 4383, 4492, 4511, 4574,  
4800, 4952, 4954, 4960, 4995, 5085,  
5088, 5103, 5131, 5132, 5215, 5227,  
5249, 5265, 5294, 5295, 5298, 5304,  
5320, 5321, 5324, 5413, 5495, 5522,  
5539, 5641, 5642, 5685, 5726, 5727,  
5759, 5772, 5789, 5814, 5822, 5843,  
5851, 5904, 5909, 5982, 5995, 6008,  
6018, 6031, 6039, 6043, 6045, 6100,  
6455, 6456, 6984, 6992, 6994, 6996  
\l\_stex\_current\_term\_tl 4957, 5178,  
5232, 5540, 5643, 5644, 5645, 5728,  
5756, 5769, 5784, 5795, 5811, 5828,  
5840, 5857, 6026, 6448, 6454, 6644  
\stex\_current\_this: . 5224, 6097, 6107  
\l\_stex\_current\_this\_tl .....  
..... 6099, 6103, 6117, 6120  
\l\_stex\_current\_type\_tl .....  
..... 131, 5100, 5231, 6316,  
6371, 6378, 6381, 6385, 6700, 6749

<code>\stex_deactivate_macro:Nn</code> .....	.. 2994, 6821, 6836, 7200, 7271, 7326
..... 142, 349,	
349, 2577, 2984, 2985, 2986, 2987,	
2988, 2989, 2990, 3271, 3340, 3474,	
3475, 3476, 3493, 3514, 3538, 3606,	
3627, 3648, 3744, 3972, 4190, 4541,	
6152, 6226, 6288, 7004, 7010, 7018,	
7026, 7047, 7080, 7098, 7107, 7382,	
7455, 7475, 7480, 7487, 7496, 7594,	
7618, 7691, 8608, 8666, 8715, 8768,	
8813, 8835, 8836, 8837, 8838, 8839,	
8840, 8860, 8911, 8926, 8927, 8928,	
8929, 8930, 8931, 8951, 8978, 9065	
<code>\stex_debug:nn</code> .....	120,
65, 65, 102, 106, 177, 196, 247,	
514, 639, 706, 713, 860, 930, 933,	
968, 972, 983, 1024, 1032, 1060,	
1065, 1085, 1088, 1121, 1164, 1192,	
1203, 1213, 1233, 1249, 1264, 1521,	
1586, 1591, 1631, 1635, 1651, 1654,	
1660, 1662, 1666, 1669, 1675, 1677,	
1680, 1689, 1690, 1716, 1763, 1765,	
1767, 1862, 1868, 1873, 1880, 2230,	
2232, 2248, 2250, 2254, 2271, 2273,	
2280, 2282, 2304, 2317, 2330, 2332,	
2354, 2394, 2472, 2474, 2483, 2587,	
2612, 2620, 2624, 2659, 2925, 2957,	
2961, 2964, 3074, 3079, 3081, 3089,	
3096, 3101, 3105, 3144, 3163, 3180,	
3186, 3193, 3209, 3220, 3235, 3253,	
3261, 3263, 3350, 3357, 3394, 3415,	
3431, 3435, 3445, 3463, 3465, 3547,	
3550, 3554, 3896, 3898, 3900, 3902,	
4019, 4036, 4075, 4086, 4088, 4241,	
4273, 4284, 4334, 4522, 4526, 4602,	
4611, 4616, 4637, 4786, 4905, 4945,	
5164, 5169, 5265, 5270, 5275, 5294,	
5297, 5300, 5320, 5372, 5381, 5405,	
5413, 5529, 5880, 6163, 6235, 6315,	
6855, 6858, 6860, 6864, 7040, 7053,	
7067, 7102, 7276, 7331, 7430, 7627	
<code>\c_stex_debug_clist</code> .....	
... 28, 38, 66, 69, 90, 94, 96, 100, 104	
<code>\c_stex_default_metatheory</code> 2176, 7587	
<code>\l_stex_default_notation</code> .....	
... 4554, 4559, 4562, 4564, 4565,	
4582, 5135, 5309, 5314, 5335, 5669	
<code>\stex_do_default_notation:</code> .....	4544, 5134, 5307, 5668
<code>\stex_do_default_notation_op:</code> ...	4544, 4545, 4580, 5332
<code>\_stex_do_deprecation:n</code> .....	120, 432, 432, 2310, 3720, 3824
<code>\_stex_do_for_list:</code> .....	
<code>\_stex_do_id:</code> .....	132, 437,
437, 1352, 6900, 6915, 7201, 7280,	
7340, 7349, 7358, 7438, 8425, 9136	
<code>\stex_do_up_to_module:n</code> .....	123, 2555, 2555, 2558, 2565, 3322, 6303
<code>\l_stex_doheader_sect</code> .....	121, 1277, 1303, 1319, 1327, 1329,
1333, 1357, 1369, 1371, 1374, 1377,	
1383, 1396, 1398, 1402, 1413, 1414,	
1415, 1416, 1417, 1418, 1420, 1423,	
1429, 1433, 1439, 1443, 7874, 7881	
<code>\_stex_eat_exclamation_point:</code> ...	3775, 5298, 5309, 5745, 5747
<code>\_stex_end:</code> .....	421, 429, 2892, 2900
<code>\_stex_every_file:</code> .....	999, 1002, 1007, 1019, 7739
<code>\stex_every_module:n</code> .....	122, 2292, 2295,
2579, 3310, 3494, 3515, 3539, 3607,	
3628, 3649, 3745, 3973, 4191, 4542,	
6153, 6227, 6290, 6300, 7595, 7619	
<code>\g_stex_every_module_tl</code> .....	2293, 2296, 2308
<code>\l_stex_every_symbol_tl</code> 5179, 5184,	
5186, 5187, 5193, 5194, 5197, 5235	
<code>\stex_execute_in_module:n</code> .....	122, 2309, 2383, 2563,
2563, 2567, 2598, 2630, 3279, 3522,	
3957, 6143, 6203, 6215, 6254, 6271	
<code>\stex_fatal_error:n</code> 142, 81, 81, 1119	
<code>\stex_fatal_error:nnn</code> .....	142, 81, 84, 87, 1130, 2032, 2101
<code>\l_stex_feature_name_str</code> .....	129, 2836, 2840,
2869, 2940, 2945, 2961, 2962, 2964,	
2965, 3008, 3353, 3369, 3403, 3407	
<code>\stex_file_in_smsmode:nn</code> .....	135, 2152, 2171, 2200, 2346, 3249
<code>\stex_file_resolve:Nn</code> .....	137,
138, 688, 693, 709, 716, 878, 979,	
988, 990, 993, 1048, 1076, 1078,	
1606, 1624, 1714, 3097, 3191, 3218	
<code>\stex_file_set:Nn</code> .. 137, 138, 682,	
682, 687, 700, 711, 874, 921, 1016, 1073	
<code>\stex_file_split_off_ext:NN</code> .....	137, 791, 791, 896, 2344, 3090
<code>\stex_file_split_off_lang:NN</code> ...	137, 791,
798, 893, 2345, 3091, 8395, 8481, 8486	
<code>\stex_file_use:N</code> .....	137,
138, 706, 713, 765, 765, 842, 847,	
852, 906, 912, 927, 930, 983, 994,	
998, 1053, 1073, 1079, 1084, 1128,	

1131, 1164, 1165, 1167, 1201, 1607,  
1608, 1620, 1699, 1705, 1710, 1715,  
1717, 1865, 1872, 1877, 1889, 1892,  
1916, 1921, 1932, 1988, 1998, 2038,  
2042, 2052, 2056, 2066, 2071, 2347,  
3093, 3096, 3097, 3099, 3120, 3135,  
3145, 3164, 3192, 3208, 3219, 3234  
\stex\_filestack\_pop: .....  
... 137, 1008, 1008, 1033, 1040, 2197  
\stex\_filestack\_push:n .....  
..... 137, 986, 986, 1026, 1028, 2181  
\l\_stex\_fors\_seq ..... 2844,  
2847, 6822, 6825, 6839, 6856, 6857,  
6871, 6872, 6930, 6957, 7204, 7205,  
7255, 7256, 7275, 7330, 7411, 7412  
\stex\_get\_env:Nn .....  
.... 142, 51, 52, 60, 88, 301, 605,  
611, 975, 977, 1044, 1046, 1051, 3659  
\stex\_get\_in\_morphism:n .....  
131, 2846, 2997, 2997, 3343, 3387, 3572  
\\_stex\_get\_mathstructure:n .....  
..... 2805, 6180, 6185  
\stex\_get\_mathstructure:n .....  
..... 6179, 6179, 6197, 6246, 6785  
\l\_stex\_get\_structure\_module\_str  
.. 2806, 6181, 6186, 6190, 6204, 6255  
\\_stex\_get\_symbol:n . 3991, 3997, 6187  
\stex\_get\_symbol:n ..... 124,  
125, 131, 1827, 3990, 3990, 4155,  
4430, 4946, 5253, 5939, 5947, 5958,  
6824, 6982, 7033, 7086, 7695, 7716  
\l\_stex\_get\_symbol\_args\_tl .....  
..... 124, 126,  
234, 3046, 3376, 3764, 3886, 3888,  
3889, 4025, 4066, 4098, 4109, 4111,  
4124, 4550, 4815, 4828, 4910, 5010,  
5023, 5258, 5423, 6168, 6848, 7672  
\l\_stex\_get\_symbol\_arity\_int ...  
..... 124, 126, 234,  
3045, 3375, 3763, 3808, 3832, 3838,  
3842, 3846, 3850, 3854, 3858, 3862,  
3866, 3870, 3873, 3874, 3875, 3876,  
3887, 3945, 4017, 4024, 4065, 4097,  
4201, 4236, 4268, 4275, 4295, 4345,  
4463, 4785, 4814, 4827, 4868, 4909,  
5009, 5022, 5061, 5257, 6847, 7671  
\l\_stex\_get\_symbol\_def\_tl .....  
..... 124, 2865, 3047,  
3351, 4026, 4067, 4099, 4911, 5259  
\l\_stex\_get\_symbol\_invoke\_cs ...  
..... 124, 3050, 3380,  
4029, 4070, 4102, 4914, 5262, 6189  
\l\_stex\_get\_symbol\_macro\_str ...  
..... 131, 3044, 3374  
\\_l\_stex\_get\_symbol\_mod\_str .....  
..... 124, 1829, 2848, 2858, 3042,  
3358, 3362, 3372, 3397, 3406, 3949,  
3955, 3998, 4022, 4063, 4095, 4161,  
4165, 4180, 4343, 4432, 4436, 4440,  
4444, 4451, 4455, 4468, 4472, 4523,  
4530, 4907, 5255, 6019, 6826, 6985,  
7035, 7088, 7681, 7687, 7708, 7729  
\\_l\_stex\_get\_symbol\_name\_str .....  
..... 124, 1829,  
2848, 2859, 2998, 3002, 3006, 3043,  
3350, 3352, 3358, 3362, 3372, 3394,  
3397, 3406, 3407, 3950, 3955, 3992,  
3999, 4023, 4064, 4096, 4161, 4165,  
4179, 4180, 4343, 4432, 4436, 4440,  
4444, 4451, 4455, 4468, 4472, 4480,  
4485, 4488, 4524, 4530, 4908, 4934,  
4936, 4942, 4944, 5256, 5949, 5960,  
5982, 5993, 5995, 6006, 6008, 6019,  
6020, 6021, 6022, 6188, 6826, 6985,  
6986, 6987, 6988, 7015, 7023, 7031,  
7035, 7076, 7084, 7088, 7682, 7687,  
7694, 7696, 7708, 7715, 7717, 7729  
\stex\_get\_symbol\_or\_var:n .....  
..... 125, 4904, 4941  
\l\_stex\_get\_symbol\_return\_tl ...  
..... 124, 3049, 3379, 4028,  
4069, 4101, 4787, 4913, 4976, 5261  
\l\_stex\_get\_symbol\_type\_tl .....  
124, 3048, 3378, 4027, 4068, 4100,  
4912, 5260, 6190, 6198, 6248, 6787  
\stex\_get\_var:n ..... 125, 4479,  
4904, 4933, 5980, 5991, 6004, 7074  
\c\_stex\_home\_file ..... 138, 1042  
\stex\_if\_check\_terms: .....  
..... 124, 3655, 3666, 3670  
\stex\_if\_check\_terms:TF 124, 3654,  
3675, 4157, 4482, 4789, 4985, 4988  
\stex\_if\_check\_terms\_p: ... 124, 3654  
\stex\_if\_do\_html: ..... 282  
\stex\_if\_do\_html:TF ..... 140,  
279, 291, 1255, 1282, 1291, 1537,  
2523, 2543, 2781, 2797, 2827, 2875,  
3060, 3284, 3325, 3361, 3395, 3754,  
3941, 3953, 4159, 4529, 4783, 4791,  
4837, 4984, 6209, 6265, 6793, 6851,  
7075, 7272, 7290, 7300, 7317, 7327,  
7361, 7368, 7370, 7408, 7629, 7654,  
7686, 8393, 8432, 8484, 8509, 8562,  
8572, 8584, 8590, 8626, 8635, 8645,  
8651, 8675, 8684, 8694, 8700, 8727,  
8736, 8747, 8753, 8792, 8826, 8852,  
8900, 8917, 8943, 8967, 9012, 9040  
\stex\_if\_do\_html\_p: ..... 140

`\stex_if_file_absolute:N` [137](#), [748](#), [758](#)  
`\stex_if_file_absolute:NTF` .....  
    ..... [137](#), [746](#), [992](#), [1077](#)  
`\stex_if_file_absolute_p:N` . [137](#), [746](#)  
`\stex_if_file_starts_with:NN` [137](#), [768](#)  
`\stex_if_file_starts_with:NNTF` ..  
    ..... [137](#), [768](#), [1200](#)  
`\stex_if_html_backend:TF` .....  
    ..... [140](#), [296](#), [321](#),  
    [328](#), [337](#), [620](#), [1255](#), [1299](#), [1361](#),  
    [1380](#), [1427](#), [1939](#), [1966](#), [3654](#), [5207](#),  
    [5241](#), [5753](#), [5809](#), [5838](#), [6016](#), [6042](#),  
    [6894](#), [6904](#), [6980](#), [7774](#), [7814](#), [7939](#),  
    [8019](#), [8067](#), [8075](#), [8087](#), [8097](#), [8207](#)  
`\stex_if_html_backend_p:` ... [140](#), [296](#)  
`\stex_if_in_module:` ..... [2547](#)  
`\stex_if_in_module:TF` .....  
    ..... [122](#), [2299](#), [2547](#), [2563](#), [4461](#)  
`\stex_if_in_module_p:` ... [122](#), [2547](#)  
`\stex_if_module_exists:n` ..... [2551](#)  
`\stex_if_module_exists:nTF` .....  
    ..... [122](#), [2316](#), [2329](#), [2551](#), [3148](#), [3151](#),  
    [3167](#), [3170](#), [3255](#), [6199](#), [6249](#), [6788](#)  
`\stex_if_module_exists_p:n` [122](#), [2551](#)  
`\stex_if_smsmode:` ..... [2137](#)  
`\stex_if_smsmode:TF` .....  
    ..... [135](#), [438](#), [1247](#), [1523](#), [2135](#),  
    [2202](#), [2335](#), [2534](#), [2542](#), [3064](#), [3207](#),  
    [3233](#), [3288](#), [3329](#), [3481](#), [3488](#), [3501](#),  
    [3509](#), [3525](#), [3533](#), [3598](#), [3618](#), [3639](#),  
    [3731](#), [3962](#), [4168](#), [4532](#), [6887](#), [6903](#),  
    [6916](#), [6929](#), [6956](#), [7059](#), [8506](#), [8508](#)  
`\stex_if_smsmode_p:` ..... [135](#), [2135](#)  
`\stex_ignore_spaces_and_pars:` ...  
    . [142](#), [358](#), [358](#), [361](#), [2573](#), [2574](#), [8446](#)  
`\l_stex_import_archive_str` . [130](#),  
    [2475](#), [2480](#), [2812](#), [3057](#), [3082](#), [3086](#),  
    [3106](#), [3107](#), [3108](#), [3276](#), [3303](#), [3319](#)  
`\stex_import_module_uri:nn` .....  
    ..... [130](#), [2473](#), [2809](#),  
    [3055](#), [3073](#), [3073](#), [3274](#), [3300](#), [3317](#)  
`\l_stex_import_name_str` .....  
    ..... [130](#), [2477](#), [2482](#), [2814](#), [3059](#), [3067](#),  
    [3076](#), [3278](#), [3292](#), [3305](#), [3321](#), [3333](#)  
`\l_stex_import_ns_str` .... [2483](#),  
    [2486](#), [2816](#), [3062](#), [3066](#), [3149](#), [3152](#),  
    [3155](#), [3168](#), [3171](#), [3174](#), [3182](#), [3280](#),  
    [3283](#), [3286](#), [3291](#), [3323](#), [3327](#), [3332](#)  
`\l_stex_import_path_str` .....  
    ..... [130](#), [2476](#), [2481](#),  
    [2813](#), [3058](#), [3077](#), [3088](#), [3092](#), [3096](#),  
    [3097](#), [3098](#), [3101](#), [3277](#), [3304](#), [3320](#)  
`\stex_import_require_module:nnn` .  
    ..... [130](#), [2479](#), [2811](#),  
    [3056](#), [3111](#), [3111](#), [3124](#), [3275](#), [3318](#)  
`\stex_import_require_module_-`  
    safe:nnn ..... [3126](#), [3302](#)  
`\l_stex_import_uri_str` .....  
    ..... [130](#), [3087](#), [3108](#), [3114](#),  
    [3119](#), [3129](#), [3134](#), [3142](#), [3144](#), [3148](#),  
    [3149](#), [3155](#), [3161](#), [3163](#), [3167](#), [3168](#),  
    [3174](#), [3193](#), [3200](#), [3220](#), [3255](#), [3256](#)  
`\stex_in_archive:nn` [134](#), [1093](#), [1093](#),  
    [1887](#), [1938](#), [1963](#), [2021](#), [2074](#), [2112](#)  
`\stex_in_invisible_html_bool` ...  
    ..... [3798](#), [4858](#), [5751](#), [5752](#), [5782](#)  
`\l_stex_in_meta_bool` [2378](#), [2379](#), [2384](#)  
`\l_stex_inparray_bool` ..... [5882](#)  
`\stex_input_with_hooks:n` .....  
    .. [1021](#), [1889](#), [1892](#), [1913](#), [1930](#), [1932](#)  
`\_stex_invoke_notation:w` .....  
    ..... [5170](#), [5287](#), [5288](#), [5293](#), [5330](#)  
`\stex_invoke_outer_field:` .... [6173](#)  
`\stex_invoke_sequence:` .....  
    ..... [5014](#), [5027](#), [5078](#), [5078](#), [5619](#)  
`\stex_invoke_sequence_in:` .... [132](#)  
`\stex_invoke_sequence_range:` .. [132](#)  
`\stex_invoke_structure:` .....  
    ..... [6141](#), [6189](#), [6309](#)  
`\stex_invoke_symbol` ..... [131](#)  
`\stex_invoke_symbol:` .....  
    ..... [125](#), [131](#), [3768](#),  
    [4819](#), [4832](#), [5264](#), [5264](#), [6849](#), [7676](#)  
`\_stex_invoke_symbol:nnnnnnN` ...  
    ..... [123](#), [131](#),  
    [2616](#), [4006](#), [4020](#), [4021](#), [5205](#), [5205](#),  
    [5218](#), [5254](#), [6715](#), [6729](#), [6737](#), [6742](#)  
`\_stex_invoke_symbol:nnnnnnN\l_-`  
    stex\_current\_symbol\_str .... [131](#)  
`\stex_invoke_text_symbol:` [3937](#), [3985](#)  
`\_stex_invoke_variable:nnnnnn` . [5239](#)  
`\_stex_invoke_variable:nnnnnnN` ..  
    ..... [131](#), [4825](#), [5020](#), [5239](#), [5617](#)  
`\_stex_is_sequentialized:n` .....  
    ..... [5611](#), [5638](#), [5713](#), [5723](#)  
`\stex_iterate_break:` [123](#), [2672](#), [2675](#)  
`\stex_iterate_break:n` . [123](#), [2672](#),  
    [2678](#), [2755](#), [3436](#), [3446](#), [3461](#), [4094](#)  
`\stex_iterate_morphisms:nn` .....  
    ..... [2738](#), [2738](#), [3414](#), [3430](#)  
`\stex_iterate_notations:nn` .....  
    ..... [126](#), [2713](#), [2713](#), [2934](#), [6749](#)  
`\stex_iterate_symbols:n` .....  
    ..... [123](#), [2671](#), [2671](#), [4087](#)  
`\stex_iterate_symbols:nn` .....  
    ..... [123](#), [2688](#), [2688](#), [2897](#), [3464](#), [6157](#), [6700](#)  
`\l_stex_key_answerclass_str` ....  
    ..... [8547](#), [8551](#), [8565](#), [8600](#), [8601](#)

<code>\l_stex_key_archive_str</code> .....	<code>\l_stex_key_macroname_str</code> .....
..... 132, 386, 389, 1570, 1588, 1602, 1611, 1612, 1689, 1696, 1710	.. 6804, 6814, 6831, 6833, 6842, 6846
<code>\l_stex_key_argnames_clist</code> .... 124	<code>\l_stex_key_method_tl</code> .....
<code>\l_stex_key_args_str</code> .....	.. 7116, 7125, 7246, 7247, 7393, 7397
..... 124, 3687, 3692, 3701, 3737, 3781, 3834, 3839, 3843, 3847, 3851, 3855, 3859, 3863, 3867, 3871, 3890, 4507, 4840, 4971, 4972, 5037, 6806	<code>\l_stex_key_mhrepos_str</code> .....
<code>\l_stex_key_argtypes_clist</code> .....	.. 8342, 8350, 8530, 8532, 8540, 9163
..... 3706, 3711, 3814, 3816, 3903, 4873, 4876, 5066, 5069, 6810	<code>\l_stex_key_min_tl</code> .....
<code>\l_stex_key_assoc_str</code> 3690, 3695, 3788, 3789, 4844, 4845, 5041, 5042	..... 8340, 8346, 8415, 8504, 8523
<code>\l_stex_key_autogradable_bool</code> ...	<code>\l_stex_key_name_str</code> .....
..... 8337, 8343, 8348, 8401, 8492	..... 124, 125, 3700, 3708, 3733, 3734, 3748, 3749, 3762, 3771, 3780, 3824, 3879, 3908, 3912, 3921, 3922, 3924, 3932, 3940, 3950, 3958, 3964, 3965, 4480, 4503, 4504, 4522, 4524, 4775, 4776, 4786, 4793, 4796, 4811, 4813, 4826, 4839, 4888, 4891, 4892, 4896, 4898, 4899, 4968, 4969, 4991, 5006, 5008, 5021, 5036, 6803, 6812, 6832, 6833, 6837, 6840, 6846, 6854, 6863, 6892, 7392, 7399, 7415, 7416, 7429, 7430, 7431, 7660, 7661, 7670, 7682, 8341, 8347, 8394, 8396, 8399, 8480, 8482, 8485, 8487, 8490
<code>\l_stex_key_continues_tl</code> . 7113, 7122	<code>\l_stex_key_number_tl</code> .....
<code>\l_stex_key_def_tl</code> 125, 3703, 3715, 3736, 3765, 3802, 3803, 3899, 3910, 3914, 3934, 4506, 4816, 4829, 4862, 4863, 5011, 5024, 5055, 5056, 6808	..... 9109, 9113, 9129, 9130
<code>\l_stex_key_deprecate_str</code> .....	<code>\l_stex_key_op_tl</code> .....
..... 412, 414, 433, 434	... 3946, 4136, 4142, 4194, 4195, 4196, 4203, 4204, 4211, 4216, 4347, 4381, 4385, 4443, 4448, 4465, 4470, 4473, 4895, 4897, 4900, 4978, 4980
<code>\l_stex_key_due_tl</code> .....	<code>\l_stex_key_post_tl</code> .. 5928, 5932, 5949, 5960, 5993, 6006, 7015, 7023
..... 9111, 9115, 9150, 9151	<code>\l_stex_key_pre_tl</code> ... 5927, 5931, 5949, 5960, 5993, 6006, 7015, 7023
<code>\l_stex_key_feedback_str</code> .....	<code>\l_stex_key_prec_str</code> .. 126, 3948, 4135, 4141, 4245, 4248, 4251, 4257, 4260, 4263, 4266, 4272, 4284, 4285
..... 8776, 8798, 8799, 8808, 8809	<code>\l_stex_key_proofend_tl</code> .....
<code>\l_stex_key_feedback_tl</code> ... 8773, 8869, 8874, 8894, 8895, 8961, 8962	..... 7112, 7121, 7223, 7224
<code>\l_stex_key_file_str</code> .....	<code>\l_stex_key_proot_tl</code> .....
..... 132, 387, 390, 1571, 1587, 1590, 1607, 1622, 1650, 1665, 1676, 1689, 1701, 1705, 1711, 1783	.. 8772, 8775, 8795, 8796, 8805, 8806
<code>\l_stex_key_for_clist</code> 2845, 6805, 6813, 6823, 7110, 7119, 7391, 7396	<code>\l_stex_key_pts_tl</code> .....
<code>\l_stex_key_for_str</code> .....	.. 8339, 8345, 8407, 8408, 8414, 8501, 8518
..... 6893	<code>\l_stex_key_reorder_str</code> 3689, 3693, 3791, 3792, 4850, 4851, 5047, 5048
<code>\l_stex_key_from_tl</code> .....	<code>\l_stex_key_return_tl</code> .....
..... 7111, 7120	... 3704, 3710, 3767, 3805, 3808, 3901, 4787, 4818, 4831, 4865, 4868, 4976, 5013, 5026, 5058, 5061, 6809
<code>\l_stex_key_Ftext_tl</code> .....	<code>\l_stex_key_role_str</code> .....
..... 8872, 8878, 8892, 8959	..... 3688, 3696, 3794, 3795, 3926, 4847, 4848, 5044, 5045, 6843
<code>\l_stex_key_functions_tl</code> . 7115, 7123	<code>\l_stex_key_root_tl</code> .....
<code>\l_stex_key_given_tl</code> .....	..... 5933
..... 9110, 9114, 9147, 9148	
<code>\l_stex_key_hide_bool</code> 7117, 7126, 7259	
<code>\l_stex_key_id_str</code> .....	
..... 132, 394, 396, 439, 440, 6874, 6875, 6942, 6944, 7337, 7346, 7435, 8556, 8558, 8564, 8620, 8622, 8628, 8669, 8671, 8677, 8720, 8722, 8729, 8781, 8783, 8791, 8794	
<code>\l_stex_key_intent_args_clist</code> ...	
..... 4138, 4144, 4310, 4319, 4321	
<code>\l_stex_key_intent_str</code> .....	
..... 3947, 4137, 4143, 4233, 4313	

<code>\l_stex_key_short_tl</code> .....	<code>\c_stex_languages_clist</code> . 29, 181, 184
..... 1314, 1317, 1344, 1346	<code>\c_stex_languages_prop</code> .....
<code>\l_stex_key_sig_str</code> .....	..... 139, 110, 138, 139, 173, 189, 805
... 122, 2305, 2347, 2495, 2510, 2527	<code>\g_stex_last_feature_str</code> .....
<code>\l_stex_key_style_clist</code> .... 133,	..... 128, 2795, 6157
406, 408, 458, 462, 509, 513, 6877,	<code>\stex_macro_body:N</code> . 141, 544, 546, 564
6878, 6971, 8830, 8831, 8920, 8921	<code>\stex_macro_definition:N</code> 141, 559, 559
<code>\l_stex_key_T_bool</code> .....	<code>\l_stex_macroname_str</code> .....
8870,	..... 125, 227, 2785,
8875, 8876, 8889, 8902, 8956, 8969	2786, 3721, 3725, 3727, 3761, 3782,
<code>\l_stex_key_term_tl</code> .....	3783, 3920, 3931, 4520, 4774, 4812,
.. 7114, 7124, 7243, 7244, 7394, 7398	4824, 4841, 4842, 4967, 5007, 5019,
<code>\l_stex_key_testspace_dim</code> .....	5038, 5039, 6142, 6842, 7659, 7669
..... 8545, 8548,	<code>\_stex_main_archive:</code> .... 1199, 1238
8550, 8579, 9003, 9005, 9057, 9060	<code>\c_stex_main_archive_prop</code> . 134, 1199
<code>\l_stex_key_title_str</code> .....	<code>\c_stex_main_file</code> .....
6815	..... 137, 138, 974, 1013, 1079, 7736
<code>\l_stex_key_title_tl</code> .....	<code>\_stex_map_args:N</code> .....
..... 400, 402, 1572, 1750, 1751,	... 3809, 4108, 4108, 4210, 4306,
2517, 6891, 6896, 8403, 8404, 8411,	4377, 4552, 4869, 5062, 5425, 6171
8495, 8496, 8599, 8600, 8660, 8661,	<code>\_stex_map_notation_args:N</code> .....
8709, 8710, 8762, 8763, 9142, 9143	..... 4108, 4121, 4410
<code>\l_stex_key_Ttext_tl</code> .....	<code>\stex_map_uri:Nnnnn</code> 138, 811, 818, 2362
..... 8871, 8877, 8890, 8957	<code>\c_stex_mathhub_file</code> 138, 918, 1042,
<code>\l_stex_key_type_tl</code> .....	1118, 1128, 1131, 1141, 1200, 1620,
125,	1699, 1710, 1865, 1877, 1889, 1892,
3702, 3714, 3735, 3766, 3799, 3800,	1916, 1921, 1932, 1984, 2038, 2042,
3897, 3909, 3913, 4505, 4817, 4830,	2052, 2056, 2066, 2071, 3120, 3135
4859, 4860, 5052, 5053, 6807, 6816	<code>\c_stex_mathhub_main_manifest_</code>
<code>\l_stex_key_variant_str</code> .....	<code>prop</code> .....
..... 126, 4134, 4140, 4147,	1206, 1207
4149, 4178, 4231, 4344, 4353, 4385,	<code>\stex_mathml_arg:nn</code> .....
4490, 4509, 4798, 4889, 4897, 4993	141
<code>\l_stex_key_wikidata_str</code> .....	<code>\stex_mathml_intent:nn</code> .....
..... 3705, 3712, 3785, 3786	141
<code>\stex_keys_define:nnnn</code> .....	<code>\_stex_maybe_brackets:nn</code> .....
..... 132, 366, 366,	.. 4605, 4622, 5086, 5322, 5333, 5875
385, 393, 399, 405, 411, 417, 1343,	<code>\stex_metagroup_do_in:nn</code> .....
1556, 1562, 2494, 3686, 3699, 3907,	..... 123, 143, 232, 237,
4133, 4500, 4766, 5926, 6116, 6802,	244, 2556, 3368, 3402, 7666, 7667, 7678
7109, 7390, 7966, 8338, 8529, 8546,	<code>\stex_metagroup_new:n</code> .....
8616, 8771, 8868, 9001, 9108, 9207	. 143, 226, 227, 231, 2307, 2356, 2840
<code>\stex_keys_set:nn</code> .....	<code>\l_stex_metatheory_uri</code> 2176, 2382,
132,	2385, 2469, 2485, 2499, 2501, 2528,
133, 381, 381, 1350, 1574, 1775,	2529, 7585, 7586, 7587, 7634, 7635
1780, 2516, 3723, 3918, 3919, 4154,	<code>\c_stex_module_</code> .....
4478, 4519, 4773, 4966, 5938, 5946,	123, 125
5957, 5979, 5990, 6003, 6131, 6884,	<code>\stex_module_add_code:n</code> .....
6913, 7007, 7013, 7021, 7199, 7270,	123, 2559, 2559, 2562, 2564, 2574, 7642
7325, 7428, 7624, 7658, 8046, 8385,	<code>\stex_module_add_morphism:nnn</code> .. 122
8389, 8470, 8539, 8555, 8578, 8619,	<code>\stex_module_add_morphism:nnnn</code> ..
8668, 8719, 8780, 8825, 8887, 8916,	..... 130, 2580, 2580,
8954, 9039, 9123, 9127, 9162, 9220	2585, 2944, 3282, 6213, 6269, 7645
<code>\stex_kpsewhich:Nn</code> 142, 43, 43, 54, 61	<code>\stex_module_add_notation:nnnnn</code> .
<code>\c_stex_language_abbrevs_prop</code> ...	..... 125, 2623,
..... 139, 110	2623, 2634, 2936, 2939, 4342, 4462
<code>\stex_language_from_file:</code> .....	<code>\stex_module_add_symbol:nnnnnN</code> .
..... 140, 158, 158, 1004	..... 122, 2586

<code>\stex_module_add_symbol:nnnnnnN</code>	<code>\_stex_notation_set_default:n ...</code>
..... 123, 2586, 2958, 2962, 2965, 3760, 3930, 6137, 6169, 6845, 7668	..... 4164, 4429, 4450, 4460, 4485
<code>\stex_module_setup:n</code> .....	<code>\stex_notation_top:nnw</code> .....
122, 2298, 2298, 2521, 2791, 7505, 7625	128
<code>\_stex_module_setup_top_nosig:n</code> .	<code>\stex_persist:n</code> .....
..... 2306, 2315, 2375, 7621	122, 141, 617, 621, 622, 624, 627, 630, 631, 637, 638, 1193, 1218, 2404
<code>\l_stex_morphism_morphisms_seq</code> ..	<code>\c_stex_persist_mode_bool</code> .....
..... 129, 2873, 2883, 2978	..... 141, 31, 608, 671, 1237
<code>\l_stex_morphism_renames_prop</code> ...	<code>\_stex_persist_read_now:</code> .....
..... 129, 2872, 2882, 2929, 2935, 2948, 2956, 3003, 3405	..... 670, 1217, 7738
<code>\l_stex_morphism_symbols_prop</code> ...	<code>\c_stex_persist_write_mode_bool</code> .
..... 129, 2861, 2871, 2881, 2899, 2903, 2912, 2926, 2931, 2999, 3017, 3022, 3371, 3467	141, 32, 614, 619, 672, 678, 1216, 2393
<code>\stex_new_statement:nn</code> .....	<code>\stex_pseudogroup:nn</code> .....
..... 6821	142, 143, 206, 206, 287, 1097, 2661, 5414
<code>\stex_new_statement:nnn</code> .....	<code>\stex_pseudogroup_restore:N</code> ....
..... 6882, 6948, 6954, 6969, 6970	..... 142, 143, 206, 209, 1110, 2665
<code>\stex_new_stylable_cmd:nnnn</code> ....	<code>\stex_pseudogroup_with:nn</code> .....
..... 133, 444, 444, 3054, 3267, 3316, 3342, 3386, 3412, 3593, 3613, 3634, 3722, 3917, 4153, 4477, 4518, 4772, 4965, 7197, 7591	..... 142, 216, 216, 819, 941, 2384, 2672, 2690, 2715, 4020, 4701, 4710, 4735, 5901, 5915, 6037
<code>\stex_new_stylable_env:nnnnnnn</code> ..	<code>\c_stex_pwd_file</code> .....
..... 133, 444, 479, 2515, 3477, 3499, 3519, 6108, 6194, 6243, 6883, 7294, 7312, 7324, 8379, 8469, 8571, 8634, 8683, 8735, 8824, 8915, 9120	138, 974, 994, 1200, 1201, 1717, 1872
<code>\_stex_next_symbol:n</code> .....	<code>\stex_reactivate_macro:N</code> 142, 349,
.. 5182, 5183, 5201, 6520, 6642, 7002	355, 2579, 2991, 2992, 2993, 3310, 3312, 3495, 3516, 3540, 3608, 3629, 3650, 3745, 3973, 4191, 4542, 6153, 6228, 6291, 6934, 6935, 6936, 6937, 6938, 6952, 6961, 6962, 6963, 6964, 6965, 6966, 6967, 7281, 7282, 7283, 7284, 7285, 7286, 7287, 7288, 7595, 7619, 7647, 8364, 8365, 8366, 8367, 8368, 8369, 8370, 8745, 8841, 8932
<code>\c_stex_no_frontmatter_bool</code> 35, 1450	<code>\stex_ref_new_doc_target:n</code> .....
<code>\l_stex_notation</code> .....	..... 121, 132, 440, 1505, 1518, 1542
126	<code>\_stex_ref_new_id:n</code> . 1508, 1519, 6943
<code>\_stex_notation_add:</code> .....	<code>\stex_ref_new_sym_target:n</code> .....
.. 3952, 4158, 4330, 4341, 4528, 7685	121, 1788, 1812, 1821, 6931, 6958, 6992
<code>\l_stex_notation_args_tl</code> .. 4122, 4200, 4234, 4305, 4311, 4317, 4526	<code>\stex_ref_new_sym_target:nn</code> 121, 1819
<code>\_stex_notation_check:</code> 4157, 4330, 4330, 4482, 4527, 4789, 4988, 7684	<code>\stex_ref_new_symbol:n</code> .....
<code>\_stex_notation_do_html:n</code> . 3955, 4161, 4330, 4350, 4530, 4793, 7687	..... 121, 1788, 2429, 3770, 3939
<code>\l_stex_notation_downprec</code> . 5541, 5869, 5870, 5880, 5881, 5901, 5903	<code>\l_stex_ref_url_str</code> . 1515, 1536, 1538
<code>\l_stex_notation_macrocode_cs</code> ...	<code>\_stex_renamedecl_do:nn</code> .....
..... 125–127, 4184, 4229, 4241, 4335, 4346, 4375, 4435, 4464, 4469, 4493, 4512, 4801, 4890, 4893, 4996	..... 3388, 3393, 3590
<code>\_stex_notation_make_args:</code> 4185, 4330, 4336, 4409, 4494, 4513, 4802	<code>\stex_require_archive:n</code> .....
<code>\stex_notation_parse:n</code> .....	..... 130, 134, 1087, 1116, 1116, 1126, 1611, 1876, 3107, 3117, 3132
..... 3951, 4156, 4193, 4193, 4481, 4525, 4788, 4979, 4982, 7683	<code>\stex_resolve_path_pair:Nnn</code> ....
<code>\stex_notation_parse_and_then:nw</code>	..... 134, 1861, 1861, 1883
..... 126, 128	<code>\_stex_return_args:nn</code> .....
	..... 3774, 3809, 4869, 5062
	<code>\l_stex_return_notation_tl</code> .....
	..... 5222, 5399, 5400, 6358, 6366, 6528, 6529, 6609, 6635, 6636



`\stex_set_current_archive:n` . . . . .  
. . . [134](#), [1086](#), [1086](#), [1106](#), [1212](#), [3251](#)  
`\stex_set_current_namespace:` . . . . .  
. . . . . [139](#), [970](#), [970](#), [1005](#)  
`\stex_set_document_uri:` . . . . .  
. . . . . [139](#), [966](#), [966](#), [1003](#)  
`\stex_set_language:n` . . . . .  
. . . . . [140](#), [136](#), [136](#), [157](#), [174](#), [2508](#)  
`\stex_set_notation_macro:nnnnn` . . . . .  
. . . . . [125](#), [126](#), [2623](#), [2640](#), [2644](#), [2644](#), [2656](#)  
`\stex_sms_allow:N` . . . . . [135](#), [2119](#),  
[2119](#), [2130](#), [2131](#), [2132](#), [2133](#), [2134](#)  
`\stex_sms_allow_env:n` . . . . .  
. . . . . [136](#), [2119](#), [2127](#), [2546](#), [3497](#), [3518](#),  
[3542](#), [6151](#), [6225](#), [6287](#), [6908](#), [7656](#)  
`\stex_sms_allow_escape:N` . . . . .  
. . . . . [136](#), [2119](#), [2123](#), [2493](#), [2578](#), [3309](#),  
[3347](#), [3391](#), [3610](#), [3631](#), [3652](#), [3746](#),  
[3974](#), [4192](#), [4543](#), [6924](#), [7048](#), [7692](#)  
`\stex_sms_allow_import:Nn` . . . . .  
. . . . . [136](#), [2140](#), [2142](#), [3311](#)  
`\stex_sms_allow_import_env:nn` . . . . .  
. . . . . [136](#), [2140](#), [2146](#)  
`\g_stex_sms_import_code` . . . . .  
. . . . . [136](#), [2151](#), [2174](#), [2192](#), [3301](#)  
`\stex_smsmode_do:` . . . . .  
. . . . . [136](#), [2165](#), [2167](#), [2201](#), [2206](#), [2491](#),  
[2539](#), [2575](#), [3269](#), [3338](#), [3345](#), [3389](#),  
[3486](#), [3506](#), [3530](#), [3604](#), [3625](#), [3646](#),  
[3742](#), [3970](#), [4174](#), [4538](#), [6110](#), [6219](#),  
[6276](#), [6901](#), [6922](#), [7045](#), [7648](#), [7689](#)  
`\_stex_split_slash:` . . . . .  
. . . . . [5969](#), [5973](#), [6022](#), [6988](#)  
`\_stex_sref_do_aux:n` . . . . .  
. . . . . [1794](#), [1801](#), [1805](#), [1808](#), [7501](#)  
`\stex_str_if_ends_with:nn` . . . . . [136](#), [534](#)  
`\stex_str_if_ends_with:nnTF` . . . . .  
. . . . . [136](#), [534](#), [987](#), [3421](#), [3444](#), [4077](#)  
`\stex_str_if_ends_with_p:nn` . . . . .  
. . . . . [136](#), [534](#), [4052](#), [4092](#)  
`\stex_str_if_starts_with:nn` [136](#), [539](#)  
`\stex_str_if_starts_with:nnTF` . . . . .  
. . . . . [136](#), [420](#), [539](#), [2898](#), [6336](#), [6340](#), [7051](#)  
`\stex_str_if_starts_with_p:nn` . . . . .  
. . . . . [136](#), [539](#)  
`\stex_structural_feature_-`  
  `module:nn` . . . . . [128](#), [2780](#), [2780](#), [6147](#)  
`\stex_structural_feature_module_-`  
  `end:` [128](#), [2780](#), [2794](#), [6112](#), [6221](#), [6283](#)  
`\stex_structural_feature_-`  
  `morphism:nnn` . . . . . [2801](#)  
`\stex_structural_feature_-`  
  `morphism:nnnn` . . . . . [130](#)  
`\stex_structural_feature_-`  
  `morphism:nnnnn` . . . . . [129](#), [2802](#),  
[3479](#), [3500](#), [3521](#), [3594](#), [3614](#), [3635](#)  
`\stex_structural_feature_-`  
  `morphism_check_total:` . . . . .  
. . . . . [2911](#), [3508](#), [3532](#), [3623](#), [3644](#)  
`\stex_structural_feature_-`  
  `morphism_end:` . . . . . [129](#), [2801](#), [2868](#),  
[3491](#), [3512](#), [3536](#), [3603](#), [3624](#), [3645](#)  
`\stex_style_apply:` . . . . .  
. . . . . [133](#), [134](#), [444](#), [449](#), [484](#),  
[489](#), [2537](#), [2542](#), [3069](#), [3294](#), [3335](#),  
[3484](#), [3489](#), [3504](#), [3510](#), [3528](#), [3534](#),  
[3601](#), [3621](#), [3642](#), [3739](#), [3967](#), [4171](#),  
[4497](#), [4535](#), [4805](#), [5000](#), [6898](#), [6904](#),  
[7209](#), [7279](#), [7299](#), [7316](#), [7336](#), [7354](#),  
[7357](#), [7366](#), [8424](#), [8431](#), [8506](#), [8508](#),  
[8568](#), [8585](#), [8589](#), [8631](#), [8646](#), [8650](#),  
[8680](#), [8695](#), [8699](#), [8732](#), [8748](#), [8752](#),  
[8849](#), [8851](#), [8940](#), [8942](#), [9135](#), [9138](#)  
`\stex_suppress_html:n` . . . . .  
. . . . . [140](#), [286](#), [286](#), [2152](#), [4224](#)  
`\_stex_symdecl_check_terms:` . . . . .  
. . . . . [125](#), [3752](#), [3894](#), [3894](#), [3929](#), [4780](#)  
`\stex_symdecl_do:` . . . . .  
. . . . . [124](#), [125](#), [3751](#), [3823](#),  
[3823](#), [3928](#), [4779](#), [4974](#), [6844](#), [7663](#)  
`\_stex_symdecl_html:` . . . . .  
. . . . . [3755](#), [3778](#), [3942](#), [6851](#)  
`\stex_symdecl_top:n` . . . . .  
. . . . . [125](#), [3729](#), [3747](#), [3747](#), [4521](#)  
`\_stex_symdef_styledefs:` . . . . . [4502](#), [4534](#)  
`\_stex_term_arg:nnn` . . . . .  
. . . . . [5532](#), [5536](#), [5543](#), [5553](#)  
`\_stex_term_arg:nnnnn` . . . . .  
. . . . . [4425](#), [4666](#), [4674](#),  
[5536](#), [5537](#), [5568](#), [5602](#), [5639](#), [5724](#)  
`\_stex_term_arg_aB:nnnnn` . . . . .  
. . . . . [4679](#), [4687](#), [4694](#), [5558](#), [5566](#)  
`\_stex_term_do_aB_clist:` . . . . . [4701](#),  
[4702](#), [4711](#), [4715](#), [4736](#), [4741](#), [5563](#),  
[5575](#), [5604](#), [5654](#), [5683](#), [5731](#), [5734](#)  
`\_stex_term_oma:nnn` . . . . .  
. . . . . [4618](#), [5424](#), [5809](#), [5810](#), [5836](#)  
`\_stex_term_oma_or_omb:nnn` . . . . .  
. . . . . [4618](#), [4623](#), [4671](#), [4684](#)  
`\_stex_term_omb:nnn` . . . . . [4671](#),  
[4684](#), [5452](#), [5453](#), [5838](#), [5839](#), [5865](#)  
`\_stex_term_oms:nnn` . . . . .  
. . . . . [4385](#), [5209](#), [5211](#), [5745](#),  
[5755](#), [5804](#), [5808](#), [5940](#), [5950](#), [5961](#)  
`\_stex_term_oms_or_omv:nnn` . . . . .  
. . . . . [3987](#), [4606](#), [5087](#), [5209](#),



5211, 5243, 5245, 5323, 5334, 5407,  
 5745, 5808, 6375, 6413, 6451, 6507  
 \stex\_term\_omv:nnn .....  
 ..... 4958, 5243, 5245,  
 5745, 5768, 5805, 5984, 5997, 6010  
 \stex\_undefined: ..... 41  
 \stex\_uri\_add\_module:NNn .....  
 ..... 139, 939, 939, 964, 1520, 7585  
 \stex\_uri\_from\_current\_file:Nn ..  
 ..... 139, 929, 929, 967  
 \stex\_uri\_from\_current\_file\_-  
 nolang:Nn ..... 139, 929, 935, 971  
 \stex\_uri\_from\_repo\_file ..... 139  
 \stex\_uri\_from\_repo\_file:NNNn ...  
 ..... 138, 139, 892, 895, 931, 1625  
 \stex\_uri\_from\_repo\_file\_-  
 nolang:NNNn ... 139, 892, 892, 936  
 \stex\_uri\_resolve:Nn ..... 138,  
 877, 877, 880, 2485, 2501, 2505, 7504  
 \stex\_uri\_set:Nn 138, 827, 873, 876, 927  
 \stex\_uri\_use:N ..... 138, 860,  
 881, 885, 933, 968, 972, 1515, 1521,  
 1522, 1526, 1592, 1594, 1627, 1634,  
 1651, 1652, 2385, 2529, 3146, 3151,  
 3152, 3165, 3170, 3171, 6875, 7635  
 \stex\_vardecl\_notation\_macro: ..  
 ... 125, 4483, 4790, 4886, 4886, 4989  
 \stex\_variable:nnnnnnnN . 4765, 4906  
 \l\_stex\_variables\_prop .....  
 ..... 125, 4763, 4811, 4918, 5006  
 stex internal commands:  
 \\_\_stex\_annotate\_env\_str ... 301, 302  
 \\_\_stex\_aux\_apply\_patch:n .. 450, 457  
 \\_\_stex\_aux\_apply\_patch\_begin:n .  
 ..... 485, 508  
 \\_\_stex\_aux\_apply\_patch\_end:n ...  
 ..... 490, 523  
 \\_\_stex\_aux\_args:n . 548, 552, 555, 558  
 \\_\_stex\_aux\_end: ..... 544, 545, 548  
 \\_\_stex\_aux\_params:n 562, 593, 600, 603  
 \\_\_stex\_aux\_patch:nnn ..... 446, 471  
 \\_\_stex\_aux\_patch:nnnn ..... 481, 498  
 \\_\_stex\_aux\_prefix:n .. 560, 568, 582  
 \\_\_stex\_aux\_prefix\_long:n .....  
 ..... 574, 578, 584, 591  
 \\_\_stex\_aux\_split\_at\_bracket:w ..  
 ..... 421, 429  
 \\_\_stex\_aux\_start: ..... 544, 547  
 \l\_stex\_aux\_tl .....  
 ..... 371, 372, 373, 375, 378, 379  
 \\_\_stex\_debug:nn ..... 67, 70, 75  
 \l\_stex\_debug\_cl ..... 90, 92, 95  
 \\_\_stex\_debug\_env\_str .... 88, 89, 92  
 \\_\_stex\_doc\_check\_topsect: .....  
 ..... 1428, 1434, 1440, 1446  
 \\_\_stex\_doc\_do\_section:n .....  
 ..... 1326, 1330, 1334, 1351  
 \\_\_stex\_doc\_maketitle: ... 1269, 1274  
 \\_\_stex\_doc\_orig\_backmatter ....  
 ..... 1462, 1465, 1483  
 \\_\_stex\_doc\_orig\_frontmatter ...  
 ..... 1452, 1455, 1472  
 \\_\_stex\_doc\_set\_title:n .. 1246, 1262  
 \\_\_stex\_doc\_skip\_fragment:n ....  
 ..... 1391, 1397,  
 1401, 1405, 1406, 1407, 1408, 1409  
 \\_\_stex\_doc\_skip\_section: .....  
 ..... 1359, 1381, 1387  
 \\_\_stex\_doc\_skip\_section\_i: ....  
 ..... 1368, 1371, 1374, 1382, 1387  
 \\_\_stex\_doc\_title\_html: .....  
 ..... 1251, 1254, 1266  
 \g\_\_stex\_doc\_title\_tl .....  
 .. 1241, 1243, 1250, 1256, 1261, 1264  
 \\_\_stex\_expr\_aB\_arg:nnnnn 5573, 5580  
 \\_\_stex\_expr\_aB\_simple\_arg:nnnnn  
 ..... 5591, 5600  
 \\_\_stex\_expr\_add\_prop\_arg:nnw ...  
 ..... 5420, 5444, 5447  
 \\_\_stex\_expr\_arg:n ..... 5421, 5457  
 \l\_stex\_expr\_arg\_counter\_int ...  
 ..... 5411, 5418, 5433, 5469, 5470  
 \\_\_stex\_expr\_arg\_do:nnn .....  
 ..... 5471, 5477, 5493, 5528, 5535  
 \\_\_stex\_expr\_arg\_inner:nn .....  
 ..... 5460, 5463, 5467, 5473  
 \\_\_stex\_expr\_assoc\_make\_seq:nnn .  
 ..... 5635, 5664, 5743  
 \\_\_stex\_expr\_assoc\_seq:nnnnnnn ..  
 ..... 5584, 5632  
 \\_\_stex\_expr\_check:n ..... 5501  
 \\_\_stex\_expr\_check:nTF ... 5470, 5476  
 \\_\_stex\_expr\_check\_b:nn .. 5425, 5450  
 \l\_stex\_expr\_count\_int .....  
 .. 5565, 5571, 5581, 5602, 5639, 5724  
 \l\_\_stex\_expr\_cs .... 5666, 5669, 5689  
 \l\_\_stex\_expr\_customs\_prop .....  
 ..... 5416, 5428, 5429,  
 5430, 5445, 5481, 5487, 5502, 5505  
 \l\_\_stex\_expr\_customs\_seq .....  
 .. 5417, 5434, 5435, 5436, 5446, 5503  
 \\_\_stex\_expr\_do\_aB\_clist: .....  
 ..... 5558, 5563, 5604, 5683  
 \\_\_stex\_expr\_do\_ab\_next:nnn ....  
 ..... 5424, 5426, 5452, 5453  
 \\_\_stex\_expr\_do\_headterm:nn ....  
 ..... 5733, 5765, 5778, 5781, 5806

<code>\__stex_expr_do_ref:nNn</code> . . .	5940,	<code>\__stex_features_do_decls:</code>	2884, 2896
	5948, 5959, 5981, 5992, 6005, 6015	<code>\__stex_features_do_elaboration:</code>	
<code>\__stex_expr_do_seqmap:nnnnnn</code> . . .			2874, 2924
	5589, 5696	<code>\__stex_features_do_for_list:</code> . . .	
<code>\__stex_expr_end:</code> . . . . .	5586, 5596		2843, 2994
<code>\__stex_expr_gobble:nnnnnnnn</code> . . .		<code>\__stex_features_do_morph:nnnn</code> . . .	
	5586, 5596		2972, 2976
<code>\l__stex_expr_iarg_tl</code>	5676, 5678, 5689	<code>\__stex_features_do_morphisms:n</code> . . .	
<code>\__stex_expr_invoke_custom:n</code> . . .			2885, 2970, 2980
	5271, 5285, 5412	<code>\__stex_features_elab_check:</code> . . .	
<code>\__stex_expr_invoke_math:</code>	5266, 5274		2932, 2955
<code>\__stex_expr_invoke_op_custom:n</code> . . .		<code>\l__stex_features_feature_str</code> . . .	
	5271, 5278, 5404		2835, 2947
<code>\__stex_expr_invoke_op_notation:w</code>		<code>\__stex_features_get_check:nnnn</code> . . .	
	5280, 5281, 5319		3000, 3028
<code>\__stex_expr_invoke_return:</code> . . . . .		<code>\l__stex_features_implicit_bool</code> . . .	
	5360, 5363		2801, 2820, 2823, 2960
<code>\__stex_expr_invoke_return_-maybe:n</code> . . . . .	5302, 5312, 5341	<code>\__stex_features_reactivate:</code> . . .	
<code>\__stex_expr_invoke_return_next:</code> . . . . .	5348, 5357		2838, 2983
<code>\__stex_expr_invoke_text:</code>	5266, 5269	<code>\__stex_features_rename:nn</code>	2948, 2951
<code>\__stex_expr_is_seqmap:n</code> . . . . .	5625	<code>\__stex_features_rename_all:</code> . .	2888
<code>\__stex_expr_is_seqmap:nTF</code> . . . . .	5588	<code>\__stex_features_renamed_-check:nnnnn</code> . . . . .	3004, 3014
<code>\__stex_expr_is_varseq:n</code> . . . . .	5614	<code>\__stex_features_set_definiens_-macros:</code> . . . . .	2853, 2857
<code>\__stex_expr_is_varseq:nTF</code>	5582, 5701	<code>\__stex_features_set_definiens_-macros_i:nnnnnnn</code> . . . . .	2860, 2864
<code>\l__stex_expr_left_bracket_str</code> . . . . .	5871, 5897, 5905, 5915, 5916	<code>\__stex_features_setup:</code> . . . . .	2837, 2880
<code>\l__stex_expr_old_seq</code> . . . . .	5708, 5711, 5715, 5720	<code>\__stex_features_split_qm:w</code> . . . . .	2923, 2941
<code>\l__stex_expr_reset_tl</code> . . . . .	5182, 5185, 5189, 5190, 5196	<code>\l__stex_features_tmp</code> . . . . .	2935, 2937, 2956, 2957, 2958
<code>\__stex_expr_ret_cs</code> . . . . .	5345, 5350, 5377, 5382, 5387	<code>\__stex_features_total_check:</code> . . . . .	2913, 2917
<code>\__stex_expr_return_arg:n</code>	5347, 5353	<code>\__stex_groups_do:</code> . . . . .	257, 264, 273
<code>\l__stex_expr_return_args_tl</code> . . . . .	5342, 5354, 5359, 5368, 5384, 5389	<code>\__stex_groups_do_in:nn</code>	239, 246, 267
<code>\__stex_expr_return_notation:n</code> . . . . .	5365, 5398	<code>\__stex_groups_exists:n</code> . . . . .	232
<code>\l__stex_expr_return_this_tl</code> . . . . .	5343, 5359, 5364, 5368, 5372, 5373, 5383, 5391	<code>\__stex_groups_exists:nTF</code> . . . . .	238
<code>\l__stex_expr_right_bracket_str</code> . . . . .	5872, 5898, 5910, 5915, 5917	<code>\l__stex_groups_ids_seq</code>	226, 229, 265
<code>\__stex_expr_setup:nnnnnn</code> . . . . .	5208, 5220, 5242	<code>\__stex_groups_tmp</code> . . . . .	248, 255, 261
<code>\__stex_expr_varseq_in_map:nnnnnnnn</code> . . . . .	5703, 5742	<code>\l__stex_importmodule_archive_-str</code>	3113, 3118, 3128, 3133, 3250, 3251
<code>\__stex_features_add_definiens:nn</code> . . . . .	2852, 2995	<code>\__stex_importmodule_check_-file:nn</code> . . . . .	3194, 3195, 3196, 3197, 3198, 3199, 3221, 3222, 3223, 3224, 3225, 3226, 3260
<code>\__stex_features_break:</code> . . . . .	2853, 2857	<code>\__stex_importmodule_get_from_-file:nnn</code> . . . . .	3154, 3190
<code>\__stex_features_check_break:nnnnnnnn</code> . . . . .	3016, 3021, 3030, 3033, 3040	<code>\__stex_importmodule_get_from_-file_safe:nnn</code> . . . . .	3173, 3217
<code>\__stex_features_clean:nw</code>	2892, 2900	<code>\__stex_importmodule_get_-module:nnn</code> . . . . .	3115, 3121, 3179
		<code>\__stex_importmodule_get_module_-safe:nnn</code> . . . . .	3130, 3136, 3185

<code>\__stex_importmodule_get_module_-uri:nnn</code> . . . . .	3140, 3181	<code>\__stex_iterate_it_decl_check:nnnn</code> . . . . .	2701, 2708
<code>\__stex_importmodule_get_module_-uri_safe:nnn</code> . . . . .	3159, 3187	<code>\__stex_iterate_it_decl_i:n</code> . . . . .	2693, 2697, 2710
<code>\__stex_importmodule_import_-module:nn</code> . . . . .	3268, 3273, 3313	<code>\__stex_iterate_it_not_check:nnnn</code> . . . . .	2729, 2733
<code>\__stex_importmodule_import_-module_presms:nn</code> . . . . .	3299, 3313	<code>\__stex_iterate_it_not_i:n</code> . . . . .	2718, 2722, 2735
<code>\__stex_importmodule_load_file:n</code> . . . . .	3211, 3214, 3238, 3243, 3248	<code>\__stex_iterate_iterate_morphism:nn</code> . . . . .	2746, 2750, 2759, 2762
<code>\l__stex_importmodule_path_seq</code> . . . . .	3090, 3091, 3093	<code>\l__stex_iterate_mods_seq</code> . . . . .	2689, 2698, 2699, 2714, 2723, 2724, 2739, 2764, 2765
<code>\__stex_importmodule_seq</code> . . . . .	3075, 3076, 3077	<code>\__stex_iterate_morphism_cs:nnnn</code> . . . . .	2741, 2767
<code>\l__stex_importmodule_seq</code> . . . . .	3097, 3099, 3191, 3192, 3218, 3219	<code>\__stex_iterate_not_cs:nnnnn</code> . . . . .	2715, 2716, 2726
<code>\l__stex_importmodule_str</code> . . . . .	3120, 3121, 3135, 3136, 3192, 3208, 3219, 3234, 3237, 3242, 3249, 3253, 3261, 3262, 3264	<code>\__stex_iterate_sym_cs:nnnnnnnN</code> . . . . .	2672, 2673, 2683, 2690, 2691, 2704
<code>\__stex_inputs_bibresource:n</code> . . . . .	1949, 1961, 1963	<code>\l__stex_iterate_todo_tl</code> . . . . .	2745, 2749, 2763, 2776
<code>\l__stex_inputs_gin_repo_str</code> . . . . .	2035, 2065, 2069, 2074	<code>\l__stex_lang_lang_str</code> . . . . .	139, 142, 149
<code>\l__stex_inputs_id_seq</code> . . . . .	1985, 1987, 1994	<code>\l__stex_lang_str</code> . . . . .	171, 172, 173, 174
<code>\l__stex_inputs_id_str</code> . . . . .	1980, 1985	<code>\l__stex_lang_turkish_bool</code> . . . . .	182, 187, 197
<code>\__stex_inputs_inputref:nn</code> . . . . .	1937, 1947	<code>\l__stex_mathhub_bool</code> . . . . .	1139, 1140, 1142, 1145, 1155, 1157, 1166
<code>\__stex_inputs_inputref_html:nn</code> . . . . .	1902, 1940	<code>\__stex_mathhub_check_manifest:</code> . . . . .	1144, 1153
<code>\__stex_inputs_inputref_pdf:nn</code> . . . . .	1926, 1941	<code>\__stex_mathhub_check_manifest:n</code> . . . . .	1154, 1156, 1158, 1163
<code>\__stex_inputs_libinput:n</code> . . . . .	2006, 2019, 2021	<code>\l__stex_mathhub_cs</code> . . . . .	1094, 1095, 1098, 1101, 1103, 1107, 1111, 1112
<code>\l__stex_inputs_libinput_files_-seq</code> . . . . .	1951, 1954, 1983, 1990, 1991, 2000, 2001, 2008, 2011, 2026, 2027, 2098, 2099	<code>\__stex_mathhub_do_manifest:n</code> . . . . .	1122, 1127
<code>\__stex_inputs_mhinput:nn</code> . . . . .	1886, 1899	<code>\__stex_mathhub_find_manifest:n</code> . . . . .	1128, 1136, 1151, 1201
<code>\l__stex_inputs_path_seq</code> . . . . .	1984, 1988, 1995, 1998	<code>\l__stex_mathhub_key</code> . . . . .	1178, 1179, 1181
<code>\l__stex_inputs_path_str</code> . . . . .	1988, 1989, 1990, 1991, 1994, 1995, 1998, 1999, 2000, 2001	<code>\c__stex_mathhub_manifest_ior</code> . . . . .	1171, 1173, 1175, 1190
<code>\l__stex_inputs_tmp_str</code> . . . . .	2027, 2029	<code>\l__stex_mathhub_manifest_str</code> . . . . .	1129, 1137, 1167, 1173, 1202
<code>\__stex_inputs_up_archive:nn</code> . . . . .	1950, 1976, 2007, 2025, 2097	<code>\__stex_mathhub_parse_manifest:n</code> . . . . .	1133, 1172, 1205
<code>\__stex_inputs_usetikzlibrary:n</code> . . . . .	2096, 2110, 2112	<code>\l__stex_mathhub_prop</code> . . . . .	1174, 1182, 1183, 1184, 1185, 1186, 1191, 1192, 1195
<code>\__stex_inputs_usetikzlibrary_-i:nn</code> . . . . .	2080, 2099	<code>\l__stex_mathhub_seq</code> . . . . .	1138, 1141, 1146, 1164, 1165, 1167, 1177, 1178, 1180
<code>\l__stex_iterate_continue_bool</code> . . . . .	2740, 2743, 2756, 2776	<code>\l__stex_mathhub_str</code> . . . . .	1044, 1046, 1048, 1051, 1052, 1056, 1057, 1062, 1068, 1071, 1076, 1079, 1209, 1210, 1212, 1219, 1223, 1226

<code>\l__stex_mathhub_tl</code> .....	1146	<code>\__stex_morphisms_do_morph_-</code>	
<code>\l__stex_mathhub_val</code> .....		<code>assign:nnn</code> .....	3419, 3422, 3460
..	1180, 1182, 1183, 1184, 1185, 1186	<code>\__stex_morphisms_do_parsed_-</code>	
<code>\__stex_module_setup_end:</code>	2353, 2374	<code>assign:</code> .....	3568, 3571
<code>\__stex_module_setup_get_uri_-</code>		<code>\__stex_morphisms_do_parsed_-</code>	
<code>str:n</code> .....	2303, 2360	<code>newname:</code> .....	3574, 3581
<code>\__stex_module_setup_load_meta:</code> .		<code>\__stex_morphisms_do_parsed_-</code>	
.....	2312, 2381	<code>newname:w</code> .....	3583, 3585, 3589
<code>\__stex_module_setup_load_sig:</code> ..		<code>\__stex_morphisms_end:</code> ...	3574, 3589
.....	2333, 2343	<code>\l__stex_morphisms_morphism_dom_-</code>	
<code>\l__stex_module_setup_ns_str</code> ...		<code>str</code> ...	3413, 3426, 3438, 3448, 3462
.....	2304, 2306, 2361, 2363, 2370	<code>\l__stex_morphisms_name_str</code> ....	
<code>\l__stex_module_setup_seg</code> .....		.....	3544, 3553, 3554, 3555, 3559, 3560, 3561, 3572
.....	2366, 2367, 2368	<code>\l__stex_morphisms_newname_str</code> ..	
<code>\l__stex_module_setup_seq</code> .....		.....	3545, 3564, 3565, 3573, 3574
.....	2365, 2366, 2368, 2370	<code>\l__stex_morphisms_next_tl</code> .....	
<code>\__stex_module_setup_setup_-</code>		.....	3551, 3556, 3558
<code>nested:n</code> .....	2299, 2351	<code>\__stex_morphisms_parse_assign:n</code>	
<code>\__stex_module_setup_setup_top:n</code>		.....	3543, 3596, 3616, 3637
.....	2299, 2302	<code>\l__stex_morphisms_seq</code> .....	
<code>\__stex_module_setup_setup_top_-</code>		.....	3548, 3549, 3551, 3553, 3556, 3558, 3560, 3562, 3564, 3566
<code>sig:n</code> .....	2306, 2328	<code>\__stex_notations_add:nnnnn</code> ....	
<code>\l__stex_module_setup_sigfile</code> ...		.....	4652, 4657, 4661
.....	2344, 2345, 2347	<code>\__stex_notations_add_last:nnnnn</code>	
<code>\__stex_module_setup_split_-</code>		.....	4645, 4656
<code>module:n</code> .....	2353, 2374	<code>\__stex_notations_add_missing_-</code>	
<code>\__stex_modules_activate_-</code>		<code>args:nn</code> .....	4210, 4325
<code>i:nnnnnnnn</code> .....	2607, 2611	<code>\__stex_notations_add_next:nnnnnn</code>	
<code>\__stex_modules_activate_not:nn</code> .		.....	4647, 4651
.....	2631, 2637	<code>\l__stex_notations_after_tl</code> ....	
<code>\__stex_modules_activate_sym:n</code> ..		.....	4631, 4658
.....	2599, 2604	<code>\__stex_notations_args_end:</code> ....	
<code>\__stex_modules_export:n</code> .	2570, 2572	.....	4111, 4114, 4117, 4124, 4127, 4130, 4640, 4643, 4653
<code>\__stex_modules_persist_module:</code> .		<code>\l__stex_notations_args_tl</code> .....	
.....	2393, 2403, 7499	.....	4547, 4551, 4564
<code>\__stex_modules_persist_not_-</code>		<code>\__stex_notations_augment_arg:nn</code>	
<code>i:nn</code> .....	2419, 2440	.....	4552, 4569
<code>\l__stex_modules_restore_mod_str</code>		<code>\__stex_notations_check_aB_-</code>	
.....	2433, 2461	<code>arg:Nn</code> .....	4692, 4700, 4709, 4734
<code>\__stex_modules_restore_module:nnnn</code>		<code>\l__stex_notations_clist_count_-</code>	
.....	2405, 2424	<code>int</code> .....	4732, 4740, 4746
<code>\__stex_modules_restore_not:n</code> ..		<code>\l__stex_notations_code_tl</code> .....	
.....	2437, 2446	... ..	4588, 4603, 4611, 4619, 4633, 4634, 4637, 4665, 4673, 4678, 4686
<code>\__stex_modules_restore_not_i:n</code>		<code>\__stex_notations_complex:nnnnnn</code>	
.....	2447, 2450, 2467	.....	4597, 4615
<code>\__stex_modules_restore_not_-</code>		<code>\__stex_notations_const_precs:</code> ..	
<code>ii:nnnnn</code> .....	2454, 2458	.....	4202, 4244
<code>\__stex_modules_set_metatheory:nn</code>		<code>\l__stex_notations_cs</code> .....	
.....	2470, 2490	.....	4604, 4609, 4620, 4629
<code>\__stex_modules_tl</code> .....	2426, 2427		
<code>\l__stex_modules_tl</code> .....	2459, 2464		
<code>\l__stex_morphisms_ass_tl</code> .....	3546, 3562, 3566, 3576, 3577		

<code>\__stex_notations_do_argname:nn</code> .	<code>\l__stex_path_auth_str</code> . . . . .
. . . . . 4306, 4309	. . . . . 833, 841, 846, 851
<code>\__stex_notations_do_argnames:</code> . .	<code>\l__stex_path_b_seq</code> 770, 776, 782, 788
. . . . . 4280, 4304	<code>\l__stex_path_b_tl</code> . . . . . 782, 783
<code>\__stex_notations_fun_precs:</code> . . .	<code>\__stex_path_canonicalize:N</code> . . . . .
. . . . . 4207, 4256	. . . . . 701, 712, 718
<code>\__stex_notations_make_arg:nmnn</code> .	<code>\__stex_path_colonslash</code> 827, 832, 837
. . . . . 4410, 4414	<code>\l__stex_path_do_hooks_pre_tl</code> . . .
<code>\__stex_notations_make_arg_-</code>	. . . . . 1022, 1023, 1035, 1038
<code>html:nn</code> . . . . . 4377, 4392	<code>\__stex_path_dodots:n</code> . 723, 727, 733
<code>\__stex_notations_make_name:</code> . . .	<code>\l__stex_path_file</code> . 900, 906, 912, 917
. . . . . 4549, 4573, 4581	<code>\__stex_path_from_repo_file:NNNNn</code>
<code>\__stex_notations_map_args_i:w</code> . .	. . . . . 893, 896, 899
. . . . . 4110, 4114, 4117	<code>\l__stex_path_maybewin_str</code> . . . . .
<code>\__stex_notations_map_args_ii:w</code> .	. . . . . 750, 751, 752
. . . . . 4123, 4127, 4130	<code>\l__stex_path_mod</code> . . . . . 844, 847, 852
<code>\__stex_notations_map_cs:</code> . . . . .	<code>\__stex_path_module:n</code> . . . . . 138,
. . . . . 4363, 4365,	815, 819, 822, 867, 881, 942, 944, 946
4712, 4714, 4722, 4737, 4739, 4750	<code>\l__stex_path_name</code> . . . . . 849, 852
<code>\l__stex_notations_missing_str</code> . .	<code>\__stex_path_name:n</code> . . . . . 138,
. . . . . 4208, 4326	816, 819, 823, 868, 881, 942, 951, 953
<code>\l__stex_notations_missing_tl</code> . . .	<code>\l__stex_path_path</code> . . . . .
. . . . . 4209, 4238, 4327	. . . . . 838, 839, 842, 847, 852, 921, 922
<code>\l__stex_notations_name_str</code> . . . . .	<code>\__stex_path_path:n</code> . . . . .
. . . . . 4575, 4576, 4577, 4582	. . . . . 138, 814, 819, 821, 866, 881, 942
<code>\l__stex_notations_opprec_tl</code> 4232,	<code>\__stex_path_relativize:N</code> . . 909, 916
4246, 4249, 4251, 4258, 4261, 4263,	<code>\l__stex_path_return_tl</code> . . . . .
4267, 4274, 4287, 4293, 4299, 4354	. . . . . 771, 777, 784, 787, 789
<code>\__stex_notations_parse_notation_-</code>	<code>\l__stex_path_seq</code> 721, 724, 729, 737,
<code>args:nmnnw</code> . . . . . 4640, 4643, 4653	738, 741, 794, 795, 796, 801, 802,
<code>\__stex_notations_parse_precs:</code> . .	804, 806, 809, 917, 919, 923, 925, 927
. . . . . 4278, 4283	<code>\g__stex_path_stack</code> . . . . .
<code>\l__stex_notations_pre_tl</code> . . . . .	. . . . . 984, 998, 1009, 1010, 1012, 1015
. . . . . 4618, 4633, 4670, 4683	<code>\l__stex_path_str</code> . . . . .
<code>\l__stex_notations_precs_seq</code> . . .	. . . . . 691, 694, 696, 697, 698, 700, 703,
. . . . . 4199, 4269,	710, 711, 720, 722, 726, 738, 793,
4276, 4297, 4299, 4312, 4318, 4355	794, 795, 800, 801, 802, 804, 805,
<code>\__stex_notations_process_-</code>	806, 975, 977, 979, 1010, 1015, 1016
<code>notation:nmnnnn</code> . . . . . 4587, 4593	<code>\l__stex_path_tl</code> . . . . . 919, 923, 925
<code>\l__stex_notations_seq</code> . . . . .	<code>\l__stex_path_uri</code> . . . . .
. . . . . 4285, 4286, 4288, 4289, 4296	. . . . . 902, 903, 904, 909, 927
<code>\__stex_notations_simple:nmnnn</code> . .	<code>\__stex_path_uri_set:NnN</code> 828, 874, 878
. . . . . 4595, 4601	<code>\__stex_path_uri_set:Nmnnn</code> . . . . .
<code>\l__stex_notations_str</code> . . . . .	. . . . . 841, 846, 851, 863, 871, 905, 911
. . . . . 4286, 4287, 4288, 4290, 4296, 4297	<code>\__stex_path_uri_use:w</code> . . . . . 881, 887
<code>\__stex_notations_styledefs:</code> . . .	<code>\l__stex_path_win_drive</code> . . . . .
. . . . . 4170, 4177	. . . . . 690, 695, 702, 704
<code>\__stex_path:</code> . . . . . 689, 698	<code>\__stex_path_win_take:w</code> . . . . . 689, 698
<code>\l__stex_path_a_seq</code> . . . 769, 775, 781	<code>\__stex_persist_env_str</code> . . . . .
<code>\l__stex_path_a_tl</code> . . . . . 781, 783	. . . . . 605, 606, 607, 611, 612, 613
<code>\__stex_path_auth:n</code> . . . . .	<code>\__stex_persist_load_file:n</code> . . . . .
. . . . . 138, 811, 814, 815,	. . . . . 634, 666, 675
816, 819, 820, 865, 881, 886, 940, 942	<code>\__stex_persist_read_and_write:</code> .
	. . . . . 651, 673

<code>\c_stex_persist_sms_iow</code> .....	<code>\_stex_refs_do_sref_in:n</code> .....
..... 617, 625, 647, 648, 663	..... 1656, 1671, 1682, 1688, 1786
<code>\_stex_persist_write_only:</code> .....	<code>\_stex_refs_do_url_link:nn</code> .....
..... 646, 660, 667, 678	..... 1838, 1852
<code>\_stex_proof_add_counter:</code> 7172, 7353	<code>\l_stex_refs_file</code> .....
<code>\_stex_proof_begin_proof:nn</code> .....	.. 1606, 1608, 1624, 1626, 1714, 1715
..... 7267, 7295, 7313	<code>\l_stex_refs_file_str</code> .....
<code>\l_stex_proof_counter_intarray</code> .....	1698, 1704, 1709, 1714, 1715, 1716,
..... 7130, 7135,	1717, 1722, 1726, 1728, 1736, 1748
7138, 7147, 7150, 7159, 7167, 7168,	<code>\g_stex_refs_files_seq</code> .....
7176, 7181, 7188, 7194, 7268, 7269	..... 1505, 1545, 1550, 1597, 1659
<code>\_stex_proof_end_list:</code> .....	<code>\_stex_refs_find_uri:n</code> .....
.. 7237, 7308, 7369, 7379, 7445, 7468	..... 1584, 1776, 1781
<code>\_stex_proof_html:</code> .. 7241, 7263, 7419	<code>\_stex_refs_find_uri_in-</code>
<code>\_stex_proof_html_env:n</code> .....	file:nnn .....
..... 7252, 7273, 7328	1593, 1598, 1630
<code>\l_stex_proof_in_spfblock_bool</code> ..	<code>\_stex_refs_find_uri_in_prop-</code>
... 7266, 7296, 7307, 7314, 7342,	file:N .....
7345, 7364, 7367, 7369, 7374, 7377,	1604, 1613, 1618
7385, 7427, 7441, 7463, 7483, 7489	<code>\l_stex_refs_id_str</code> .....
<code>\_stex_proof_inblock_restore:</code> ..	..... 1724, 1759, 1765, 1766
..... 7371, 7384, 7485	<code>\c_stex_refs_iow</code> .....
<code>\_stex_proof_inc_counter:</code> .....	..... 1496, 1497, 1498, 1524
..... 7155, 7367, 7458, 7460	<code>\_stex_refs_new_symbol:n</code> 1790, 1798
<code>\l_stex_proof_inc_counter_bool</code> 7129	<code>\l_stex_refs_prop</code> .....
<code>\_stex_proof_insert_number:</code> .....	1612, 1613
..... 7131, 7351, 7458, 7460	<code>\_stex_refs_restore_target:nnnnn</code>
<code>\_stex_proof_make_step_macro:Nnnnn</code>	..... 1725, 1757
..... 7425, 7458, 7459, 7460	<code>\l_stex_refs_return_tl</code> .....
<code>\_stex_proof_number_as_string:N</code>	..... 1723, 1727, 1733, 1747
..... 7142, 7338, 7347, 7436	<code>\_stex_refs_set_keys_b:n</code> .....
<code>\_stex_proof_proof_box_tl</code> 7112, 7216	..... 1568, 1655, 1670, 1681, 1782
<code>\_stex_proof_remove_counter:</code> .....	<code>\l_stex_refs_str</code> 1511, 1513, 1515,
..... 7184, 7378	1520, 1522, 1527, 1800, 1802, 6944
<code>\_stex_proof_start_list:n</code> .....	<code>\_stex_refs_sym_aux:nn</code> .....
..... 7230, 7296, 7351, 7445, 7468	..... 1829, 1842, 1847, 1858
<code>\_stex_proof_step_html:nn</code> 7407,	<code>\l_stex_refs_unnamed_counter-</code>
7443, 7445, 7451, 7466, 7468, 7472	int .....
<code>\_stex_refs_add_doc_ref:nn</code> .....	1506, 1510, 1511
..... 1522, 1544, 1555	<code>\l_stex_refs_uri</code> .....
<code>\l_stex_refs_default_archive_-</code>	..... 1520, 1521, 1625, 1627
str .....	<code>\l_stex_refs_uri_str</code> .....
1564, 1570, 1579	... 1585, 1596, 1608, 1619, 1624,
<code>\l_stex_refs_default_file_str</code> ..	1627, 1634, 1649, 1659, 1660, 1661,
..... 1565, 1571, 1580	1662, 1663, 1666, 1667, 1669, 1675,
<code>\l_stex_refs_default_title_tl</code> ..	1677, 1678, 1680, 1690, 1691, 1692,
..... 1566, 1572, 1581	1719, 1730, 1738, 1758, 1763, 1764
<code>\_stex_refs_do_autoref:n</code> .....	<code>\_stex_seqs_add:</code> .....
..... 1640, 1652, 1663,	4986, 5004
1667, 1678, 1692, 1718, 1729, 1737	<code>\l_stex_seqs_args_tl</code> ... 5062, 5064
<code>\_stex_refs_do_internal_link:nn</code>	<code>\_stex_seqs_check_terms:</code> 4985, 5032
..... 1832, 1844, 1850	<code>\l_stex_seqs_clist</code> .....
<code>\_stex_refs_do_return:nnnn</code> .....	..... 5099, 5106, 5113, 5117
..... 1746, 1760, 1768	<code>\l_stex_seqs_cs</code> .....
<code>\_stex_refs_do_sref:nn</code> .. 1648, 1777	..... 5060, 5064, 5120, 5124,
	5132, 5135, 5144, 5151, 5164, 5165
	<code>\_stex_seqs_do_all:w</code> ... 5158, 5168
	<code>\_stex_seqs_do_first:</code> ... 5081, 5142

<code>\__stex_seqs_do_first_arg:n</code> . . . .	<code>\__stex_statements_do_defref:nn</code> .
. . . . . 5140, 5147	. . . . . 6979, 7008, 7014, 7022
<code>\__stex_seqs_do_first_next:</code> . . . .	<code>\__stex_statements_force_id:</code> . . .
. . . . . 5149, 5154	. . . . . 6838, 6941, 6949, 6972
<code>\__stex_seqs_do_one:w</code> . . . 5156, 5162	<code>\__stex_statements_html_keyvals:nn</code>
<code>\__stex_seqs_do_op:w</code> . . . . 5080, 5084	. . . . . 6868, 6889, 6917
<code>\__stex_seqs_doop_arg:n</code> . . 5100, 5111	<code>\__stex_statements_setup:nn</code> . . . .
<code>\__stex_seqs_doop_range:w</code> 5093, 5097	. . 6830, 6950, 6955, 6969, 6973, 6976
<code>\l__stex_seqs_first_args_tl</code> . . . .	<code>\__stex_statements_setup_def:</code> . . .
. . . . . 5146, 5164, 5165, 5169, 5170	. . . . . 6928, 6951, 6974
<code>\__stex_seqs_get_index_notation:n</code>	<code>\l__stex_statements_uri_str</code> . . . .
. . . . . 5092, 5130, 5163	6853, 6857, 6858, 6863, 6864, 7034,
<code>\__stex_seqs_html:</code> . . . . . 4984, 5034	7037, 7040, 7042, 7087, 7090, 7093
<code>\__stex_seqs_macro:</code> . . . . . 4987, 5018	<code>\l__stex_structures_assigned_seq</code>
<code>\__stex_seqs_make_args:</code> . . 4997, 5031	. . . . . 6516, 6627, 6666, 6682
<code>\l__stex_seqs_range_clist</code> . . . . .	<code>\__stex_structures_begin:nn</code> . . . .
. . . . . 4977, 5012, 5025	. . . . . 6109, 6130, 6207, 6262
<code>\g__stex_smsmode_allowed_escape_-</code>	<code>\__stex_structures_check_-</code>
<code>tl</code> . . . . . 2117, 2124, 2253	<code>def:nnnnnnnn</code> . . . . . 6281, 6294
<code>\g__stex_smsmode_allowed_import_-</code>	<code>\l__stex_structures_clist</code> . . . . .
<code>env_seq</code> 2141, 2147, 2187, 2236, 2239	. . . . . 6490, 6511, 6517, 6519
<code>\g__stex_smsmode_allowed_import_-</code>	<code>\l__stex_structures_comp_cs</code> . . . .
<code>tl</code> . . . . . 2140, 2143, 2184, 2231	. . . . . 6559, 6564
<code>\g__stex_smsmode_allowed_tl</code> . . . .	<code>\l__stex_structures_cs</code> . . . . .
. . . . . 2116, 2120, 2249	. . . . . 6492, 6498, 6505
<code>\g__stex_smsmode_allowedenvs_seq</code>	<code>\__stex_structures_current_type:</code>
. . . . . 2118, 2128, 2258, 2261	. . . . . 6446, 6577, 6648
<code>\g__stex_smsmode_bool</code> . . . . .	<code>\l__stex_structures_current_-</code>
. . . . . 2135, 2136, 2138, 2154	<code>type_tl</code> . . . . 6373, 6409, 6449, 6454
<code>\__stex_smsmode_check_begin:Nn</code> . .	<code>\__stex_structures_do_assign:nn</code> .
. . . . . 2236, 2258, 2270	. . . . . 6393, 6397
<code>\__stex_smsmode_check_cs:NNn</code> . . .	<code>\__stex_structures_do_assign_-</code>
. . . . . 2209, 2217	<code>list:n</code> . . . . . 6369, 6390
<code>\__stex_smsmode_check_end:Nn</code> . . .	<code>\__stex_structures_do_decl:nnnnnnnn</code>
. . . . . 2239, 2261, 2279	. . . . . 6704, 6724
<code>\__stex_smsmode_do:w</code> . . . . .	<code>\__stex_structures_do_decl_-</code>
. . . . . 2203, 2215, 2217, 2219,	<code>nomacro:nnnnnnnn</code> . . . . 6702, 6710
2241, 2251, 2263, 2276, 2283, 2286	<code>\__stex_structures_do externals:</code>
<code>\__stex_smsmode_do_aux:N</code> . 2217, 2223	. . . . . 6113, 6155, 6222, 6284
<code>\__stex_smsmode_do_aux_curr:N</code> . . .	<code>\__stex_structures_end:</code> . . . . .
. . . . . 2183, 2194, 2225	. . . . . 6337, 6341, 6357, 6362
<code>\__stex_smsmode_do_aux_imports:N</code>	<code>\l__stex_structures_exstruct_-</code>
. . . . . 2183, 2229	<code>name_str</code> . . . . . 6257, 6262, 6306
<code>\__stex_smsmode_do_aux_normal:N</code> .	<code>\__stex_structures_extend_-</code>
. . . . . 2194, 2247	<code>structure:nn</code> . . . . . 6234, 6257
<code>\l__stex_smsmode_importmodules_-</code>	<code>\__stex_structures_extend_-</code>
<code>seq</code> . . . . . 2172	<code>structure_i:NnnnnnnN</code> . 6231, 6239
<code>\__stex_smsmode_in_smsmode:n</code> . . .	<code>\__stex_structures_external_-</code>
. . . . . 2152, 2182, 2192, 2193	<code>decl:nnnn</code> . . . . . 6158, 6162
<code>\l__stex_smsmode_sigmodules_seq</code> 2173	<code>\l__stex_structures_extmod_str</code> . .
<code>\__stex_smsmode_smsmode_do:</code> . . . .	. . . . . 6232, 6236
. . . . . 2165, 2201	<code>\l__stex_structures_extname_-</code>
<code>\__stex_smsmode_start_smsmode:n</code> .	<code>count</code> . . . . . 6300, 6304, 6306
. . . . . 2163, 2190, 2195	



<code>\l__stex_structures_field_name_-str</code> . . . . .	6614, 6618, 6619, 6650	<code>\__stex_structures_present_i:w</code> . . . . .	6460, 6466, 6472
<code>\l__stex_structures_fields_clist</code> . . . . .	6370, 6391, 6398, 6416, 6419, 6673, 6674	<code>\__stex_structures_present_ii:nw</code> . . . . .	6477, 6485
<code>\__stex_structures_get_field_-name:n</code> . . . . .	6613, 6625	<code>\l__stex_structures_prop</code> . . . . .	6504, 6616, 6624, 6656, 6664, 6681, 6684, 6690, 6711, 6713, 6725, 6727
<code>\l__stex_structures_imports_seq</code> . . . . .	6195, 6200, 6208, 6245, 6250, 6264, 6786, 6789, 6792	<code>\__stex_structures_prop_do_-decls:</code> . . . . .	6668, 6699
<code>\__stex_structures_invokation_-type:n</code> . . . . .	6321, 6368	<code>\__stex_structures_prop_do_-notations:</code> . . . . .	6669, 6748
<code>\__stex_structures_invoke_-field:n</code> . . . . .	6591, 6623	<code>\l__stex_structures_redo_tl</code> . . . . .	6643, 6667, 6693, 6740, 6751, 6766
<code>\__stex_structures_invoke_maybe_-field:nn</code> . . . . .	6580, 6584	<code>\l__stex_structures_replace_-this_tl</code> . . . . .	6156
<code>\__stex_structures_invoke_this:n</code> . . . . .	6330, 6572	<code>\l__stex_structures_seq</code> . . . . .	6665, 6712, 6726, 6750
<code>\__stex_structures_invoke_top:n</code> . . . . .	6311, 6314, 6338, 6342, 6345, 6348, 6350	<code>\l__stex_structures_set_comp_tl</code> . . . . .	6310, 6359, 6363, 6521, 6638
<code>\__stex_structures_make_mod:n</code> . . . . .	6415, 6427	<code>\__stex_structures_set_custom_-comp:n</code> . . . . .	6364, 6553
<code>\__stex_structures_make_oml:n</code> . . . . .	6419, 6433	<code>\__stex_structures_set_customcomp:</code> . . . . .	6337, 6362
<code>\__stex_structures_make_oml:nn</code> . . . . .	6434, 6436	<code>\__stex_structures_set_this:n</code> . . . . .	6586, 6595
<code>\__stex_structures_make_prop:</code> . . . . .	6458, 6585, 6663	<code>\__stex_structures_set_thiscomp:</code> . . . . .	6310, 6547
<code>\__stex_structures_make_prop_-assign:</code> . . . . .	6459, 6588, 6672	<code>\__stex_structures_set_thisnotation:</code> . . . . .	6341, 6357
<code>\__stex_structures_make_prop_-assign:nn</code> . . . . .	6675, 6680	<code>\__stex_structures_shift_-args:nn</code> . . . . .	6171, 6176
<code>\__stex_structures_make_prop_-assign_replace:nnnn</code> . . . . .	6683, 6689	<code>\l__stex_structures_this_tl</code> . . . . .	6322, 6522, 6523, 6526, 6541, 6598, 6601, 6608, 6629, 6630, 6633, 6647
<code>\__stex_structures_make_type:n</code> . . . . .	6378, 6383, 6385, 6401	<code>\__stex_symdecl_add_decl:</code> . . . . .	3753, 3759
<code>\__stex_structures_maybe_-notation:w</code> . . . . .	6325, 6327, 6453	<code>\l__stex_symdecl_args_tl</code> . . . . .	3809, 3811
<code>\__stex_structures_merge:nw</code> . . . . .	6319, 6335	<code>\l__stex_symdecl_cs</code> . . . . .	3807, 3811, 4001, 4003, 4005, 4031
<code>\l__stex_structures_more_-nextsymbol_tl</code> . . . . .	6626, 6628, 6653	<code>\__stex_symdecl_do_args:</code> . . . . .	3826, 3885
<code>\l__stex_structures_name_str</code> . . . . .	6101, 6118, 6123, 6125, 6132, 6133, 6138, 6139, 6144, 6145, 6148, 6164, 6170	<code>\__stex_symdecl_env_str</code> . . . . .	3659, 3660, 3661
<code>\__stex_structures_new_extstruct_-name:</code> . . . . .	6244, 6302	<code>\__stex_symdecl_get_from_one_-string:n</code> . . . . .	4040, 4085
<code>\__stex_structures_present:</code> . . . . .	6464, 6589	<code>\__stex_symdecl_get_symbol_from_-cs:</code> . . . . .	4007, 4018
<code>\__stex_structures_present:mn</code> . . . . .	6468, 6474, 6479, 6486, 6489	<code>\__stex_symdecl_get_symbol_from_-modules:nn</code> . . . . .	4042, 4074
<code>\__stex_structures_present_-entry:nn</code> . . . . .	6494, 6500, 6515	<code>\__stex_symdecl_get_symbol_from_-string:n</code> . . . . .	4008, 4010, 4013, 4035
		<code>\l__stex_symdecl_name</code> . . . . .	4038, 4044
		<code>\__stex_symdecl_parse_arity:</code> . . . . .	3825, 3831
		<code>\l__stex_symdecl_seq</code> . . . . .	4037, 4038, 4039, 4043



\_stex_symdecl_set_textsymdecl_	\stexstyleextstructure	104
macro:nnn	\stexstyleimportmodule	104, 133
\_stex_symdecl_sym_from_str_	\stexstyleinterpretmod	104
i:nnnn	\stexstyleinterpretmodule	104
\_stex_symdecl_sym_i_finish:nnnnnnN	\stexstylemathstructure	104
.....	\stexstylemcb	8856
\_stex_symdecl_sym_i_gobble:nnnnnn	\stexstyleMMTinclud	104
.....	\stexstylemodule	104, 134
\_stex_vars_add:	\stexstylenotation	104
\_l_stex_vars_args_tl	\stexstyleparagraph	104
\_l_stex_vars_bind_bool	\stexstyleproof	104
.....	\stexstylerealization	104
\_stex_vars_check_var:nnnnnnnnN	\stexstylerealize	104
.....	\stexstylerenamedecl	104
\_l_stex_vars_cs	\stexstylerequiremodule	104
\_stex_vars_get_var:n	\stexstylescb	8947
.....	\stexstylespfsketch	104
\_stex_vars_html:	\stexstylesubproof	104
\_stex_vars_macro:	\stexstylesymdecl	104
\_stex_vars_set_vars:nnnnnnN	\stexstylesymdef	104
.....	\stexstyletextsymdecl	104
\TeX/ComputerScience/Software	\stexstyleusemodule	104
stex_annotate_env (env.)	\stexstylevardef	104
\stexcommentfont	\stexstylevarnotation	104
.....	\stexstylevarseq	104
.....	\stopsolutions	111, 8609
\stexdoctitle	str commands:	
.....	\_c_backslash_str	696
.....	\_c_colon_str	827, 882, 1177, 2363
\STEXexport	\_c_dollar_str	4319
.....	\_c_hash_str	595, 2625, 4326
\stexhtmlfalse	\_c_percent_str	54, 1044
.....	\_str_case:Nn	1181
\stexhtmltrue	\_str_case:nn	4663, 5451
.....	\_str_case:nnTF	1412, 3835, 4393,
\STEXInternalAssocArgMarkerI	.....	4415, 5479, 5507, 7699, 7720, 7838
.....	\_str_clear:N	386, 387, 394, 412,
\STEXInternalAssocArgMarkerII	.....	695, 902, 1062, 1137, 1585, 1903,
.....	.....	2069, 2179, 2361, 2495, 2803, 2998,
\STEXInternalNotation	.....	3086, 3113, 3128, 3413, 3544, 3545,
.....	.....	3687, 3688, 3689, 3690, 3700, 3701,
\STEXInternalSetSrefSymURL	.....	3705, 3725, 3908, 3947, 3948, 3998,
.....	.....	3999, 4134, 4135, 4136, 4137, 4907,
.....	.....	4934, 4942, 5539, 6118, 6186, 6803,
\STEXInternalSrefRestoreTarget	.....	6804, 6806, 6853, 7031, 7084, 7143,
.....	.....	7392, 7623, 7694, 7715, 7967, 8238,
.....	.....	8239, 8341, 8342, 8530, 8547, 8772
\STEXInternalSymbolAfterInvokationTL	\_str_const:Nn	7757
.....	\_str_count:n	536, 541
.....	\_str_gset:Nn	1502, 1815, 1823
\STEXInternalTermMathArgiii	\_str_gset_eq:NN	1057, 2869, 2870
.....	\_str_if_empty:NTF	89,
\STEXInternalTermMathAssocArgiiii	.....	433, 439, 606, 612, 702, 722, 909,
.....	.....	1050, 1052, 1071, 1129, 1202, 1590,
\STEXInternalTermMathOMAi		
.....		
\STEXInternalTermMathOMBiii		
.....		
\STEXInternalTermMathOMSOrOMviii		
.....		
\STEXinvisible		
.....		
\STEXRestoreNotsEnd		
.....		
\stexstyle		
.....		
\stexstyleassertion		
.....		
\stexstyleassign		
.....		
\stexstyleassignMorphism		
.....		
\stexstylecopymod		
.....		
\stexstylecopymodule		
.....		
\stexstyledefinition		
.....		
\stexstyleexample		
.....		

1596, 1602, 1649, 1650, 1665, 1676,	3155, 3161, 3168, 3171, 3174, 3192,
1696, 1719, 1730, 1738, 1758, 1783,	3219, 3264, 3462, 3555, 3561, 3565,
1849, 2305, 2548, 2785, 2808, 3002,	3695, 3727, 3749, 3839, 3843, 3847,
3006, 3088, 3250, 3426, 3559, 3573,	3851, 3855, 3859, 3863, 3867, 3871,
3660, 3748, 3782, 3785, 3788, 3791,	3920, 3922, 3924, 3926, 4022, 4023,
3794, 3921, 3992, 4146, 4245, 4257,	4063, 4064, 4095, 4096, 4147, 4179,
4266, 4313, 4775, 4841, 4844, 4847,	4208, 4213, 4332, 4372, 4383, 4520,
4850, 4936, 4944, 4968, 4971, 5038,	4774, 4776, 4908, 4952, 4954, 4967,
5041, 5044, 5047, 6039, 6122, 6132,	4969, 4972, 5227, 5228, 5641, 5685,
6181, 6188, 6618, 6831, 6832, 6837,	5726, 6018, 6021, 6100, 6123, 6133,
6854, 6874, 6942, 7037, 7090, 7337,	6142, 6236, 6300, 6304, 6306, 6614,
7346, 7415, 7429, 7435, 7660, 7696,	6619, 6843, 6857, 6863, 6984, 6987,
7717, 7828, 7887, 8050, 8394, 8480,	7034, 7087, 7339, 7348, 7437, 7659,
8485, 8556, 8600, 8620, 8669, 8720,	7661, 8558, 8622, 8671, 8722, 8783
8762, 8781, 8795, 8798, 8805, 8808	
<code>\str_if_empty:nTF</code> . . . . . 472, 499,	<code>\str_set_eq:NN</code> . . . . .
683, 734, 829, 1509, 1863, 2498, 3399	1068, 1570, 1571, 2535, 2806, 2836,
<code>\str_if_eq:NNTF</code> . . . . . 172, 783	3949, 3950, 4149, 4178, 4480, 4490,
<code>\str_if_eq:nnTF</code> . . . . .	4509, 4523, 4524, 4798, 4993, 5982,
. . . . . 146, 162, 163, 164, 186,	5995, 6008, 6125, 6190, 6833, 6842,
302, 535, 540, 570, 586, 595, 607,	6892, 6944, 7323, 7404, 7622, 7626,
613, 697, 735, 736, 752, 1633, 1717,	7650, 7652, 7665, 7679, 7681, 7682
1759, 1764, 1766, 1820, 2285, 2367,	<code>\str_uppercase:n</code> . . . . . 7959
2606, 2639, 2744, 3015, 3020, 3029,	<code>\l_tmpa_str</code> . . . . . 159, 160, 161,
3032, 3208, 3234, 3418, 3434, 3661,	162, 163, 164, 165, 169, 185, 189,
4248, 4260, 4272, 4924, 4927, 5118,	190, 191, 193, 1903, 1904, 1905,
7054, 7891, 7894, 7959, 8118, 8862	1910, 1918, 2821, 2824, 2829, 2836
<code>\str_if_eq_p:nn</code> 4050, 4051, 4090, 4091	<code>\ subparagraph</code> . . . . . 27, 82
<code>\str_if_exist:NNTF</code> . . . . . 233, 1846	subproblem (env.) . . . . . 8467
<code>\str_if_in:NnTF</code> . . . . 4326, 6020, 6986	subproof (env.) . . . . . 7323
<code>\str_item:Nn</code> . . . . . 697, 752, 3890	<code>\subproof</code> . . . . . 7281, 7382
<code>\str_lowercase:n</code> . . . . . 8862	<code>\subproofautorefname</code> . . . . . 7323
<code>\str_map_break:</code> . . . . . 3836	<code>\subsection</code> . . . . . 26, 27, 82
<code>\str_map_break:n</code> . 3837, 3841, 3845,	<code>\subsubsection</code> . . . . . 27, 82
3849, 3853, 3857, 3861, 3865, 3869	<code>\svar</code> . . . . . 95, 3775, 4949
<code>\str_map_inline:Nn</code> . . . . . 3834	<code>\symbol</code> . . . . . 89
<code>\str_new:N</code> . . . . .	<code>\symdecl</code> . . . . . 23,
. . . 135, 1564, 1565, 2035, 2290, 3721	24, 31, 32, 39, 40, 47, 87–89, 99,
<code>\str_put_right:Nn</code> . . . . . 7150	104, 124, 129, 135, 142, 143, 2984,
<code>\str_range:Nnn</code> . . . . . 2029	3721, 7523, 7525, 7554, 7562, 7565
<code>\str_range:nnn</code> . . . . . 536, 541, 575	<code>\symdef</code> . . . . . 32, 33, 35, 40, 41, 88,
<code>\str_replace_all:Nnn</code> . . . . . 696	95, 124, 129, 135, 2986, 4500, 7507,
<code>\str_set:Nn</code> . . . . .	7510, 7516, 7518, 7520, 7522, 7532,
. . . 48, 56, 137, 185, 228, 690, 691,	7533, 7534, 7535, 7541, 7549, 7560,
694, 710, 827, 1084, 1167, 1179,	7569, 7570, 7573, 7576, 7578, 7580
1180, 1511, 1513, 1515, 1579, 1580,	<code>\Symname</code> . . . . . 89, 100, 5924
1608, 1619, 1627, 1634, 1698, 1704,	<code>\symname</code> 19, 20, 24, 62, 89, 90, 100, 102, 5924
1709, 1715, 1724, 1865, 1872, 1877,	<code>\symref</code> . . . . . 19,
1988, 1998, 2027, 2065, 2324, 2340,	20, 24, 48, 88, 89, 93, 100, 102, 5924
2363, 2433, 2662, 2816, 2821, 2824,	<code>\symrefemph</code> . . . . . 102, 103, 6036
2835, 2858, 2859, 3042, 3043, 3044,	<code>\symuse</code> . 48, 89, 5173, 5252, 5729, 6410,
3077, 3082, 3087, 3092, 3098, 3106,	6417, 6428, 6510, 6575, 6645, 6646
3108, 3114, 3118, 3119, 3120, 3129,	sys commands:
3133, 3134, 3135, 3142, 3149, 3152,	<code>\sys_get_shell:nnN</code> . . . . . 45

<code>\sys_if_platform_windows:TF</code> . . . . .		<code>\defemph@uri</code> . . . . .	103, 6036
. . . . .	51, 688, 747, 974, 1043	<code>\define@key</code> . . . . .	2037, 2051, 2064
<b>T</b>			
<code>\target</code> . . . . .	134, 137–139	<code>\Gin@eheight</code> . . . . .	9269, 9272, 9277, 9282
<code>\test</code> . . . . .	73, 115	<code>\Gin@ewidth</code> 8152, 8153, 9268, 9278, 9282	
<code>test</code> . . . . .	110	<code>\Gin@exclamation</code> . . . . .	9268, 9269, 9277
<code>\testbigspace</code> . . . . .	9072	<code>\Gin@mhrepos</code> . . . . .	
<code>\testemptypage</code> . . . . .	73, 115	. . . . .	2038, 2041, 2045, 2066, 2070, 2075
<code>\testemptypage</code> . . . . .	9066	<code>\hwexam@bonuspts</code> . . . . .	9230
<code>testheading (env.)</code> . . . . .	74, 116	<code>\hwexam@checktime</code> . . . . .	9224
<code>\testheading</code> . . . . .	9197	<code>\hwexam@duration</code> 9200, 9203, 9209, 9214	
<code>\testmedspace</code> . . . . .	9071	<code>\hwexam@kw@due</code> . . . . .	9151
<code>\testnewpage</code> . . . . .	73, 115	<code>\hwexam@kw@forgrading</code> . . . . .	9192
<code>\testnewpage</code> . . . . .	9073	<code>\hwexam@kw@given</code> . . . . .	9148
<code>\testsmallspace</code> . . . . .	73, 73, 73, 115, 115, 115	<code>\hwexam@kw@grade</code> . . . . .	9193
<code>\testsmallspace</code> . . . . .	9070	<code>\hwexam@kw@probs</code> . . . . .	9193
<code>\testspace</code> . . . . .	73, 115	<code>\hwexam@kw@pts</code> . . . . .	9194
<code>\testspace</code> . . . . .	8579, 9069	<code>\hwexam@kw@reached</code> . . . . .	9195
<code>\TeX</code> . . . . .	23, 24	<code>\hwexam@kw@sum</code> . . . . .	9193
<code>\tex</code> . . . . .	23, 24	<code>\hwexam@kw@testemptypage</code> . . . . .	9067
<code>\TeX</code> and <code>L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub></code> commands:		<code>\hwexam@min</code> . . . . .	9201, 9208, 9213, 9224
<code>\@</code> . . . . .	2084, 2087, 2091	<code>\hwexam@minutes@kw</code> . . . . .	9201
<code>\@addtoreset</code> . . . . .	8357	<code>\hwexam@reqpts</code> 9210, 9215, 9227, 9231	
<code>\@arabic</code> . . . . .	8210	<code>\hwexam@tools</code> . . . . .	9211, 9216
<code>\@author</code> . . . . .	8189	<code>\hwexam@totalmin</code> . . . . .	9223, 9224
<code>\@auxout</code> . . . . .	1795, 7744, 8461	<code>\hwexam@totalpts</code> . . . . .	9222, 9231
<code>\@bonuspointsfalse</code> . . . . .	9228	<code>\if@bonuspoints</code> . . . . .	9226
<code>\@bonuspointstrue</code> . . . . .	9233	<code>\if@rustex</code> . . . . .	306
<code>\@currenthref</code> . . . . .	7339, 7348, 7437	<code>\itemize@inner</code> . . . . .	7999, 8005, 8014
<code>\@currentcounter</code> . . . . .	1528	<code>\itemize@label</code> . . . . .	8003, 8006, 8009
<code>\@currentlabel</code> . . . . .	1529, 7338,	<code>\itemize@level</code> 7997, 8002, 8005, 8014	
7339, 7347, 7348, 7436, 7437, 8049		<code>\itemize@outer</code> . . . . .	7998, 8002
<code>\@currentlabelname</code> . . . . .	1531, 1532	<code>\lst@mhrepos</code> . . . . .	2052, 2055, 2059
<code>\@currenvir</code> . . . . .	7933, 7934	<code>\ltx@ifpackageloaded</code> 141, 145, 2036,	
<code>\@dblarg</code> . . . . .	8182, 8191	2050, 2063, 7217, 8320, 8882, 9091	
<code>\@gobble</code> . . . . .	361	<code>\m@switch</code> . . . . .	5895
<code>\@ifnextchar</code> . . . . .	360	<code>\mdf@patchamsthm</code> . . . . .	8023
<code>\@ifstar</code> . . . . .	8161	<code>\metakeys@show@keys</code> . . . . .	8000
<code>\@mainmatterfalse</code> . . . . .	1457, 1467	<code>\notesslides@slidelabel</code> . . . . .	8026, 8041
<code>\@mainmattertrue</code> . . . . .	1478, 1489	<code>\ns@author</code> . . . . .	8182, 8183
<code>\@notprerr</code> . . . . .	140, 1422	<code>\ns@title</code> . . . . .	8191, 8192
<code>\@onlypreamble</code> . . . . .	140, 1422	<code>\pgf@temp</code> . . . . .	2081, 2082, 2083
<code>\@rustexfalse</code> . . . . .	298	<code>\pgfkeys@spdef</code> . . . . .	2081
<code>\@title</code> . . . . .	1271, 1272, 8198	<code>\pgfutil@empty</code> . . . . .	2083
<code>\activate@excursion</code> . . . . .	8222, 8227	<code>\pgfutil@InputIfFileExists</code> . . . . .	2090
<code>\bbl@loaded</code> . . . . .	8321, 9092	<code>\prematurestop@endsfragment</code> . . . . .	
<code>\beamer@shortauthor</code> . . . . .	8185, 8187, 8202	. . . . .	7932, 7935, 7941
<code>\beamer@shorttitle</code> . . . . .	8194, 8196, 8205	<code>\problem@kw@*</code> . . . . .	8317
<code>\c@chapter</code> . . . . .	7914	<code>\problem@kw@correct</code> . . . . .	8890, 8957
<code>\c@framenummer</code> . . . . .	8210	<code>\problem@kw@feedback</code> . . . . .	8809
<code>\c@part</code> . . . . .	7926	<code>\problem@kw@grading</code> . . . . .	8762
<code>\compemph@uri</code> 103, 131, 4369, 6028, 6036		<code>\problem@kw@hint</code> . . . . .	8660
<code>\correction@table</code> . . . . .	9176	<code>\problem@kw@minutes</code> . . . . .	8442, 8523, 8821
		<code>\problem@kw@note</code> . . . . .	8709
		<code>\problem@kw@pts</code> . . . . .	8437, 8518, 8816

<code>\problem@kw@solution</code> .....	8599	<code>\thistitle</code> .....	103, 133, 2517, 2518, 2519, 6891, 8411, 8454, 8455
<code>\problem@kw@wrong</code> .....	8892, 8959	<code>\thistype</code> .....	3735, 4505
<code>\problem@restore</code> .....	8462, 8466, 9170	<code>\thisvarname</code> .....	.. 4488, 4492, 4796, 4800, 4991, 4995
<code>\stex@backend</code> .....	140, 296	<code>\throwaway</code> .....	135
<code>\symrefemph@uri</code> .....		<code>\tikzinput</code> ...	118, 2075, 9261, 9266, 9292
.... 102, 103, 5940, 5950, 5961, 6036		tikzinput commands:	
<code>\varemp@uri</code> .....		<code>\c_tikzinput_image_bool</code>	34, 9250, 9257
.... 103, 131, 5981, 5994, 6007, 6036		<code>\tiny</code> .....	8027, 8041
<code>\texname</code> .....	24	<code>\title</code> .....	73, 116
<code>\text</code> .....	20	<code>\title</code> .....	8191
<code>\textbf</code> 19, 6078, 7819, 8453, 9051, 9141, 9146		tl commands:	
<code>\textcolor</code> .....	9047	<code>\tl_clear:N</code> .....	400, 459, 510, 771, 1344, 1723, 2174, 2293, 2499, 2763, 3068, 3293, 3334, 3546, 3702, 3703, 3704, 3738, 3886, 3909, 3910, 3946, 3966, 4200, 4209, 4305, 4448, 4489, 4508, 4547, 4742, 4797, 4957, 4980, 4992, 5222, 5232, 5329, 5342, 5540, 5927, 5928, 5929, 6026, 6117, 6598, 6626, 6667, 6807, 6808, 6809, 7110, 7111, 7113, 7114, 7115, 7116, 7393, 7394, 8237, 8773, 8869, 8871, 8872, 9002, 9109, 9110, 9111, 9181, 9182, 9183, 9208, 9209, 9210, 9211
<code>\textsymdecl</code> .. 23, 24, 88, 129, 2985, 3906		<code>\tl_const:Nn</code> .....	5867, 5868
<code>\textwarning</code> .....	108	<code>\tl_count:N</code> .....	5484, 5490
<code>\textwidth</code> .....	8150, 8172, 9189	<code>\tl_count:n</code> .....	5615, 5626
<code>\the</code> 251, 252, 254, 256, 258, 2084, 2085, 2086		<code>\tl_gclear:N</code> .....	2319
<code>\theassignment</code> .....	9134, 9141	<code>\tl_gput_left:Nn</code> .....	372, 374, 375
<code>\thechapter</code> ... 1332, 1373, 1401, 1436, 7845, 7850, 7854, 7858, 7862, 7866		<code>\tl_gput_right:Nn</code> .....	252, 2120, 2124, 2143, 2296, 2560, 3301, 8223
<code>\theframenumber</code> .....	8049	<code>\tl_gset:Nn</code> .....	254, 367, 368, 1023, 1243, 1250, 2144, 2148, 4216, 5185, 5189, 8501, 8504, 9173, 9174
<code>\theparagraph</code> .....	7862, 7866	<code>\tl_gset_eq:NN</code> .....	46, 2795
<code>\thepart</code> ... 1328, 1370, 1397, 1430, 7840		<code>\tl_head:N</code> .. 886, 940, 4005, 6548, 6554	
<code>\theplainsproblem</code> .....	8361, 8362	<code>\tl_head:n</code> .....	.... 554, 595, 598, 4693, 5616, 5627
<code>\thesection</code> .....		<code>\tl_if_empty:NTF</code> .....	.... 524, 787, 1025, 1261, 1271, 1314, 1750, 2382, 2518, 2528, 3351, 3576, 3765, 3799, 3802, 3805, 3934, 4109, 4122, 4194, 4203, 4211, 4381, 4470, 4546, 4859, 4862, 4865, 4873, 4895, 4978, 5052, 5055, 5058, 5066, 5296, 5308, 5399, 5419, 5504, 5643, 5756, 5769, 5811, 5840, 6099, 7223, 7243, 7246, 7634, 8153, 8244, 8403, 8454, 8495, 8599, 8660, 8709, 8890, 8892, 8894, 8957, 8959, 8961, 9048, 9129, 9142, 9147, 9150, 9200, 9227
.. 7850, 7854, 7858, 7862, 7866, 8362			
<code>\thesproblem</code> . 8358, 8362, 8453, 8462, 9134			
<code>\thesubparagraph</code> .....	7866		
<code>\thesubsection</code> ... 7854, 7858, 7862, 7866			
<code>\thesubsubsection</code> .....	7858, 7862, 7866		
<code>\this</code> .....	57–59, 96, 97, 5224, 6097, 6526, 6603, 6604, 6608, 6633		
<code>\thisarchive</code> .....	3290, 3331		
<code>\thisargs</code> .....	3737, 4507		
<code>\thiscopyname</code> .....			
.. 3482, 3502, 3526, 3599, 3619, 3640			
<code>\thisdeclname</code> .... 3734, 3965, 4179, 4504			
<code>\thisdecluri</code> .....			
.. 3733, 3964, 4180, 4183, 4503, 4511			
<code>\thisdefiniens</code> .....	3736, 4506		
<code>\thisfor</code> .....	6893		
<code>\thismodulename</code> 133, 2536, 3067, 3292, 3333			
<code>\thismoduleuri</code> .....			
.... 133, 2535, 3066, 3291, 3332, 3483, 3503, 3527, 3600, 3620, 3641			
<code>\thisname</code> .....	6892		
<code>\thisnotation</code> 4181, 4491, 4510, 4799, 4994			
<code>\thisnotationvariant</code> .....			
.... 4178, 4490, 4509, 4798, 4993			
<code>\ThisStyle</code> .....	5895		
<code>\thisstyle</code> .....	133, 459, 462, 463, 464, 510, 513, 514, 515, 516, 524, 527, 528, 3068, 3293, 3334, 3738, 3966, 4489, 4508, 4797, 4992		

<code>\tl_if_empty:nTF</code> . . . . .	369, 553, 569, 585, 594, 751, 760, 882, 883, 945, 952, 1099, 1569, 1727, 1749, 1907, 1929, 1960, 2018, 2109, 2216, 2582, 2597, 2649, 2709, 2734, 2804, 2819, 2918, 2977, 3078, 3112, 3127, 3141, 3160, 3191, 3218, 3407, 4003, 4116, 4129, 4594, 4644, 4951, 5447, 5468, 5544, 5567, 5970, 6171, 6295, 6344, 6347, 6392, 6402, 6491, 6493, 6587, 6596, 6691, 6701, 6765, 6910, 7032, 7085, 7101, 7597, 8184, 8193	5062, 5146, 5179, 5184, 5186, 5193, 5223, 5229, 5230, 5231, 5343, 5364, 5484, 5490, 5503, 5509, 5512, 5515, 5558, 5644, 5676, 5678, 5728, 5871, 5872, 5916, 5917, 6156, 6168, 6237, 6310, 6322, 6358, 6359, 6363, 6366, 6373, 6409, 6448, 6522, 6528, 6538, 6601, 6603, 6628, 6629, 6635, 6644, 6692, 6694, 6736, 6741, 6753, 6756, 6760, 6763, 6768, 6771, 6775, 6778, 7216, 7222, 7840, 7841, 7845, 7846, 7850, 7851, 7854, 7855, 7858, 7859, 7862, 7863, 7866, 7867, 7882, 8339, 8340, 8452, 8477, 8478, 8538, 8827, 8881, 8888, 8913, 8924, 8955, 9013, 9014, 9161, 9167, 9168, 9224, 9230
<code>\tl_if_empty_p:N</code> . . . . .	777	
<code>\tl_if_eq:NNTF</code> . . . . .		
. . . . .	1141, 6189, 6548, 6554, 9019	
<code>\tl_if_eq:NnTF</code> . . . . .		
. . . . .	8270, 8416, 8419, 8436, 8441	
<code>\tl_if_eq:nnTF</code> . . . . .		
. . . . .	2451, 4718, 5112, 5627, 5672, 5710	
<code>\tl_if_exist:NNTF</code> . . . . .	210, 251, 266, 305, 463, 515, 527, 1287, 1296, 1531, 1691, 2552, 6256, 7871	
<code>\tl_if_in:NnTF</code> . . . . .	2231, 2249, 2253	
<code>\tl_item:nn</code> . . . . .	5618	
<code>\tl_map_inline:Nn</code> . . . . .	2184	
<code>\tl_map_inline:nn</code> . . . . .	217, 221	
<code>\tl_new:N</code> . . . . .		
. . . . .	965, 1035, 1241, 1566, 2116, 2117, 2140, 2151, 2469, 4763, 5178, 5182	
<code>\tl_put_left:Nn</code> . . . . .		
. . . . .	4633, 4634, 5373, 8384, 9122	
<code>\tl_put_right:Nn</code> . . . . .		
. . . . .	1217, 2370, 2745, 2749, 3888, 3889, 4311, 4317, 4327, 4554, 4559, 4562, 4665, 4673, 4678, 4686, 4749, 5210, 5234, 5244, 5354, 6597, 6693, 6740, 6751, 6766, 9023, 9185, 9186, 9187	
<code>\tl_range:nnn</code> . . . . .	571, 587	
<code>\tl_set:Nn</code> . . . . .	211, 248, 454, 473, 475, 494, 495, 500, 501, 503, 504, 750, 784, 864, 958, 1022, 1279, 1320, 1455, 1465, 1581, 1747, 2038, 2041, 2536, 2645, 2647, 2650, 2865, 3046, 3047, 3048, 3049, 3050, 3290, 3331, 3482, 3502, 3526, 3556, 3562, 3566, 3599, 3619, 3640, 3733, 3809, 3964, 4025, 4026, 4027, 4028, 4029, 4066, 4067, 4068, 4069, 4070, 4098, 4099, 4100, 4101, 4102, 4180, 4195, 4204, 4229, 4246, 4249, 4258, 4261, 4267, 4274, 4293, 4503, 4551, 4564, 4582, 4603, 4618, 4619, 4631, 4670, 4683, 4702, 4715, 4741, 4824, 4869, 4910, 4911, 4912, 4913, 4914, 5019,	
. . . . .	5062, 5146, 5179, 5184, 5186, 5193, 5223, 5229, 5230, 5231, 5343, 5364, 5484, 5490, 5503, 5509, 5512, 5515, 5558, 5644, 5676, 5678, 5728, 5871, 5872, 5916, 5917, 6156, 6168, 6237, 6310, 6322, 6358, 6359, 6363, 6366, 6373, 6409, 6448, 6522, 6528, 6538, 6601, 6603, 6628, 6629, 6635, 6644, 6692, 6694, 6736, 6741, 6753, 6756, 6760, 6763, 6768, 6771, 6775, 6778, 7216, 7222, 7840, 7841, 7845, 7846, 7850, 7851, 7854, 7855, 7858, 7859, 7862, 7863, 7866, 7867, 7882, 8339, 8340, 8452, 8477, 8478, 8538, 8827, 8881, 8888, 8913, 8924, 8955, 9013, 9014, 9161, 9167, 9168, 9224, 9230	
<code>\tl_set_eq:NN</code> . . . . .	350, 371, 373, 378, 1572, 2517, 3066, 3067, 3291, 3292, 3332, 3333, 3483, 3503, 3527, 3600, 3620, 3641, 3734, 3735, 3736, 3737, 3965, 4183, 4251, 4263, 4287, 4435, 4443, 4488, 4504, 4505, 4506, 4507, 4550, 4787, 4796, 4887, 4892, 4896, 4976, 4991, 5423, 5563, 5604, 5683, 6454, 6526, 6608, 6633, 6891, 7112, 8411, 8414, 8415, 9222, 9223	
<code>\tl_set_rescan:Nnn</code> . . . . .	139, 190	
<code>\tl_tail:n</code> . . . . .		
. . . . .	554, 555, 600, 2216, 5589, 6418	
<code>\tl_to_str:n</code> . . . . .	66, 69, 97, 100, 110, 122, 218, 222, 223, 247, 350, 572, 588, 684, 706, 713, 812, 830, 832, 835, 854, 860, 911, 981, 1073, 1177, 1249, 1264, 1502, 1789, 1814, 1815, 1843, 1844, 1850, 1899, 1947, 2107, 2128, 2144, 2147, 2148, 2185, 2188, 2230, 2232, 2248, 2250, 2254, 2272, 2281, 2425, 2427, 2428, 2431, 2432, 2462, 2463, 2521, 2589, 2590, 2591, 2613, 2846, 3000, 3075, 3350, 3359, 3548, 4290, 4602, 4616, 4695, 4740, 5529, 6166, 6199, 6249, 6316, 6336, 6340, 6499, 6627, 6682, 6712, 6726, 6750, 6788, 6824, 7741, 8321, 9092	
<code>\tl_trim_spaces:N</code> . . . . .	49	
<code>\l_tmpa_tl</code> . . . . .	45, 46, 48, 4321, 4742, 4749, 4756, 5173, 5471, 5477, 5482, 5484, 5488, 5490, 5493, 5502, 5504, 5509, 5512, 5515, 5730, 6406, 8888, 8904, 8907, 8955, 8971, 8974, 9013, 9019, 9181, 9185, 9193	
<code>\l_tmpb_tl</code> . . . . .		
. . . . .	5471, 5477, 5479, 5493, 5503, 5507, 9014, 9019, 9182, 9186, 9194	

tmpc commands:	
<code>\l_tmpc_tl</code> .....	9183, 9187, 9195
<code>\to</code> .....	7542, 7545, 7555, 7556
<code>\today</code> .....	8212
<code>\TODO</code> .....	5031, 5032, 6393, 8268
todo internal commands:	
<code>\l_todo_mmt_module_str</code> .....	7622, 7627, 7640, 7643, 7646, 7650, 7665, 7681
<code>\_todo_newlabel:n</code> .....	7740, 7747
<code>\l_todo_old_metagroup_cd</code> .....	7666, 7678
<code>\_todo_old_newlabel:</code> .....	7741, 7746
<code>\l_todo_stex_module_str</code> .....	7626, 7627, 7652, 7679
token commands:	
<code>\c_math_subscript_token</code> .....	49, 87, 4417, 4418, 4421, 4422, 4426, 6538, 7547, 7558, 7560
<code>\topsep</code> .....	7232, 8512, 8788, 8844, 8935
<code>\trueFfalseT</code> .....	8996
<code>\trueTfalseF</code> .....	8991
<code>\type</code> .....	73, 116
<b>U</b>	
<code>\undefined</code> .....	134, 41
<code>\univ</code> .....	63
<code>\unless</code> .....	7933
<code>\uppercase</code> .....	5964
<code>\uri</code> .....	138, 139
use commands:	
<code>\use:N</code> .....	249, 382, 460, 464, 466, 511, 516, 518, 525, 528, 530, 1315, 1317, 1726, 1749, 1847, 1852, 2185, 2188, 2663, 3936, 5088, 5298, 5324, 6456, 7872, 7878
<code>\use:n</code> .....	1835, 2028, 2036, 2050, 2063, 3806, 4866, 5059, 5254, 5344, 5376, 6508, 6615
<code>\use:nn</code> .....	55, 198, 201, 207, 547, 1317, 2460, 2484, 2815, 2860, 3016, 3021, 3760, 3930, 4184, 4335, 4374, 4493, 4512, 4801, 4996, 5143, 5165, 5170, 5386, 5589, 5636, 5721, 5896, 6137, 6169, 6574, 6641, 6683, 6845, 7103, 7431, 7668, 8248, 8540, 9163
<code>\use_i:nn</code> .....	6615, 9186
<code>\use_ii:nn</code> .....	1839, 2937, 2952, 6655
<code>\use_none:nnnnnn</code> .....	3035
<code>\usebox</code> .....	8104
<code>\usemodule</code> .....	15, 18, 22, 23, 26, 37, 49, 93, 94, 96, 212, 423, 430, 3054
<code>\usepackage</code> .....	7, 21, 2028, 7947
<code>\useSGvar</code> .....	109, 8261
<code>\usestructure</code> .....	96, 6784
<code>\usetHEME</code> .....	7947
<code>\usetikzlibrary</code> .....	80, 2106, 9260
<b>V</b>	
<code>\varbind</code> .....	47, 52, 95, 100, 6938, 6961, 7072, 7080
<code>\vardef</code> .....	35, 47, 52, 95, 124, 125, 4763, 7103, 7431
<code>\varempH</code> .....	103, 6036
<code>\Varname</code> .....	5924
<code>\varname</code> .....	5924
<code>\varnotation</code> .....	95, 4477
<code>\varref</code> .....	5924
<code>\varseq</code> .....	43, 95, 4965
<code>\vbox</code> .....	135, 7218, 8021, 8110, 8125, 8580, 8641, 8690, 8742, 8883
vbox commands:	
<code>\vbox_set:Nn</code> .....	2153
<code>\vdash</code> .....	7570
<code>\vfill</code> .....	7876, 9067
<code>\vM</code> .....	54
<code>\vMs</code> .....	55
<code>\vn</code> .....	43
<code>\vop</code> .....	45, 47
<code>\vrule</code> .....	7218, 8172, 8883
<code>\vskip</code> .....	7218, 8171, 8173, 8883
<code>\vspace</code> .....	9069
<b>W</b>	
<code>\wff</code> .....	20
<code>\withbrackets</code> .....	92, 5914, 5922
<b>X</b>	
<code>\xspace</code> .....	1287, 1296, 3980, 3981, 3987, 3988
<b>Y</b>	
<code>\yesFnoT</code> .....	8986
<code>\yesTnoF</code> .....	8981
<code>\yield</code> .....	7286, 7477, 7480