

# The $\text{\textcolor{blue}{S}\textcolor{red}{T}\textcolor{blue}{E}\textcolor{red}{X}3}$ Package Collection\*

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

2023-03-19

## Contents

<b>1</b>	<b>Introduction &amp; Setup</b>	<b>6</b>
1.1	What is $\text{\textcolor{blue}{S}\textcolor{red}{T}\textcolor{blue}{E}\textcolor{red}{X}}$ ? . . . .	6
1.2	The $\text{\textcolor{teal}{S}\textcolor{blue}{T}\textcolor{red}{E}\textcolor{blue}{X}}$ package . . . . .	7
1.3	What is MMT? . . . . .	7
1.4	Math archives and the MathHub Directory . . . . .	7
1.5	Setting Up the $\text{\textcolor{blue}{S}\textcolor{red}{T}\textcolor{blue}{E}\textcolor{red}{X}}$ IDE . . . . .	8
<b>I</b>	<b>Tutorial</b>	<b>10</b>
<b>2</b>	<b>The Basics</b>	<b>11</b>
2.1	Text symbols . . . . .	14
2.1.1	Using Modules & Search in the IDE . . . . .	14
2.2	Symbol References . . . . .	17
2.3	Modules and Simple Symbol Declarations . . . . .	20
2.4	Documenting Symbols . . . . .	23
2.5	Sectioning and Reusing Document Fragments . . . . .	25
2.6	Building and Exporting HTML . . . . .	27
<b>3</b>	<b>Mathematical Concepts</b>	<b>30</b>
3.1	Simple Symbol Declarations . . . . .	30
3.1.1	Semantic Macros and Notations . . . . .	30
3.1.2	Types and Variables . . . . .	33
3.1.3	Flexary Macros and Argument Modes . . . . .	38
3.1.4	Precedences . . . . .	40
3.1.5	Implicit Arguments . . . . .	40
3.1.6	Finishing Equality . . . . .	42
3.1.7	Variable Sequences . . . . .	42
3.2	Statements . . . . .	43
3.2.1	Definitions . . . . .	43
	Semantic Macros in Text Mode . . . . .	45

---

\*Version 3.3.0 (last revised 2023-03-19)

Definientia . . . . .	46
Using Symbols Without Semantic Macros and Exporting Code in Modules . . . . .	47
3.2.2 Assertions . . . . .	48
3.2.3 Proofs . . . . .	51
3.3 Mathematical Structures . . . . .	51
3.3.1 Declaring and Using Structures . . . . .	51
Instantiating Structures . . . . .	52
3.3.2 Extending Structures and Axioms . . . . .	54
Conservative Extensions . . . . .	54
3.3.3 Nesting Structures and <code>\this</code> . . . . .	56
3.4 Complex Inheritance and Theory Morphisms . . . . .	58
3.4.1 Glueing Structures Together . . . . .	60
3.4.2 Realizations . . . . .	62
<b>4 Extensions for Education</b> . . . . .	<b>64</b>
4.1 Slides and Course Notes . . . . .	64
4.2 Problems and Exercises . . . . .	64
4.3 Exams . . . . .	64
<b>II User Manual</b> . . . . .	<b>65</b>
<b>5 Basics</b> . . . . .	<b>66</b>
5.1 Package and Class Options . . . . .	66
5.2 Math Archives and the MathHub Directory . . . . .	67
5.2.1 The Structure of Math Archives . . . . .	68
5.2.2 MANIFEST.MF-Files . . . . .	68
5.3 The lib-Directory . . . . .	69
5.4 Basic Macros . . . . .	70
<b>6 Document Features</b> . . . . .	<b>71</b>
6.1 Document Fragments . . . . .	71
6.2 Using and Referencing Document Fragments . . . . .	72
6.3 Cross-Document References . . . . .	72
<b>7 Modules and Symbols</b> . . . . .	<b>75</b>
7.1 Modules . . . . .	75
7.1.1 Signature Modules, Languages, and Multilinguality . . . . .	76
7.2 Symbol Declarations . . . . .	76
7.2.1 Returns . . . . .	77
7.3 Referencing Symbols . . . . .	77
7.4 Notations and Semantic Macros . . . . .	79
7.4.1 Precedences and Bracketing . . . . .	80
7.4.2 Notations for Argument Sequences . . . . .	81
7.4.3 Semantic Macros . . . . .	81
7.5 Simple Inheritance . . . . .	82
7.6 Variables and Sequences . . . . .	84
7.7 Structures . . . . .	85
7.7.1 Semantic Macros for Structures . . . . .	85

<b>8 Statements</b>	<b>88</b>
8.1 More on Definitions . . . . .	89
8.2 More on Assertions . . . . .	90
<b>9 Customizing Typesetting</b>	<b>91</b>
9.1 Highlighting Symbol References . . . . .	91
9.2 Styling Environments and Macros . . . . .	92
9.3 Custom CSS for Environments . . . . .	93
<b>10 Additional Packages</b>	<b>94</b>
10.1 NotesSlides Manual . . . . .	94
10.1.1 Introduction . . . . .	94
10.1.2 Package Options . . . . .	94
10.1.3 Notes and Slides . . . . .	95
10.1.4 Customizing Header and Footer Lines . . . . .	96
10.1.5 Frame Images . . . . .	96
10.1.6 Ending Documents Prematurely . . . . .	97
10.1.7 Global Document Variables . . . . .	97
10.1.8 Excursions . . . . .	98
10.2 Problem Manual . . . . .	99
10.2.1 Introduction . . . . .	99
10.2.2 Problems and Solutions . . . . .	99
10.2.3 Markup for Added-Value Services . . . . .	100
Multiple Choice Blocks . . . . .	101
Filling-In Concrete Solutions . . . . .	102
10.2.4 Including Problems . . . . .	104
10.2.5 Testing and Spacing . . . . .	104
10.3 HWExam Manual . . . . .	104
10.3.1 Introduction . . . . .	104
10.3.2 Package Options . . . . .	105
10.3.3 Assignments . . . . .	105
10.3.4 Including Assignments . . . . .	105
10.3.5 Typesetting Exams . . . . .	105
10.4 Tikzinput Manual . . . . .	106
<b>III Documentation</b>	<b>108</b>
<b>11 S<sup>T</sup>E<sub>X</sub> Developer Manual</b>	<b>109</b>
11.1 Documents . . . . .	110
11.2 Modules . . . . .	110
11.3 Symbols . . . . .	112
11.4 Notations . . . . .	114
11.5 Structural Features . . . . .	117
11.6 Imports and Morphisms . . . . .	119
11.7 Expressions and Semantic Macros . . . . .	120
11.8 Optional (Key-Value) Argument Handling . . . . .	121
11.9 Stylable Commands and Environments . . . . .	122
11.10 Math Archives . . . . .	123
11.11 SMS-Mode . . . . .	124

11.11.1	Second Pass . . . . .	124
11.11.2	First Pass . . . . .	125
11.12	Strings, File Paths, URIs . . . . .	125
11.12.1	File Paths . . . . .	126
File Path Constants and Variables	. . . . .	127
11.12.2	URIs . . . . .	127
URI Constants and Variables	. . . . .	128
11.13	Language Handling . . . . .	128
11.14	Inserting Annotations . . . . .	129
11.14.1	Backend macros . . . . .	129
11.15	Persisting Content from Math Archives in sms-Files . . . . .	130
11.16	Utility Methods . . . . .	130
11.16.1	Group-like Behaviours . . . . .	131
<b>12</b>	<b>Additional Packages</b>	<b>133</b>
12.1	NotesSlides Documentation . . . . .	133
12.2	Problem Documentation . . . . .	133
12.3	HWExam Documentation . . . . .	133
12.4	Tikzinput Documentation . . . . .	133
<b>IV</b>	<b>Implementation</b>	<b>134</b>
<b>13</b>	<b>The <math>\text{\texttt{S}\text{\texttt{I}\text{\texttt{E}\text{\texttt{X}}}}</math> Implementation}</b>	<b>135</b>
13.1	Setting up . . . . .	135
13.2	Utilities . . . . .	136
13.2.1	Calling kpsewhich and Environment Variables . . . . .	136
13.2.2	Logging . . . . .	136
13.2.3	Languages . . . . .	137
13.2.4	Group-like Behaviours . . . . .	140
13.2.5	HTML Annotations . . . . .	141
13.2.6	Auxiliary Methods . . . . .	143
13.2.7	Persistence . . . . .	149
13.2.8	Files, Paths and URIs . . . . .	150
13.2.9	File Hooks . . . . .	157
13.3	Math Archives . . . . .	159
13.4	Documents . . . . .	163
13.4.1	Title . . . . .	163
13.4.2	Sectioning . . . . .	164
13.4.3	References . . . . .	168
13.4.4	Inputs . . . . .	175
13.5	SMS Mode . . . . .	180
13.6	Modules . . . . .	184
13.6.1	The smodule-environment . . . . .	184
13.6.2	Structural Features . . . . .	195
13.7	Inheritance . . . . .	201
13.7.1	\importmodule/\usemodule . . . . .	201
13.7.2	Theory Morphisms . . . . .	206
13.8	Symbols . . . . .	212
13.8.1	Declarations . . . . .	212

13.8.2	Notations . . . . .	221
	a/B-mode argument handling . . . . .	232
13.8.3	Variables . . . . .	234
13.8.4	Sequences . . . . .	238
13.8.5	Expressions . . . . .	242
	Invoking Semantic Macros . . . . .	243
	Argument Handling and Annotating . . . . .	249
	Term HTML Annotations . . . . .	254
	Automated Bracketing . . . . .	256
	Symname and Variants . . . . .	258
	Highlighting . . . . .	260
13.9	Mathematical Structures . . . . .	261
13.10	Statements . . . . .	274
13.11	Proofs . . . . .	280
13.12	Metatheory . . . . .	288
13.13	MMT Interfaces . . . . .	290
<b>14</b>	<b>Additional Packages</b>	<b>293</b>
14.1	Implementation: The notesslides Package . . . . .	293
	14.1.1 Class and Package Options . . . . .	293
	14.1.2 Notes and Slides . . . . .	297
	14.1.3 Environment and Macro Patches . . . . .	300
	14.1.4 Styling Across Notes/Slides . . . . .	302
	14.1.5 Beamer Compatibility . . . . .	302
	14.1.6 TODO Excursions . . . . .	303
14.2	Implementation: The problem Package . . . . .	304
	14.2.1 Package Options . . . . .	304
	14.2.2 Problems and Solutions . . . . .	305
14.3	Implementation: The hwexam Package . . . . .	314
	14.3.1 Package Options . . . . .	314
	14.3.2 Assignments . . . . .	315
	14.3.3 Leftovers . . . . .	318
14.4	Tikzinput Implementation . . . . .	319

## Index 321



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 sTeX, you can talk to us directly at <https://matrix.to/#/#stex:fau.de>.*

# Chapter 1

## Introduction & Setup

### 1.1 What is $\text{\textcolor{blue}{S}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}}$ ?

$\text{\textcolor{blue}{S}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}}$  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 [\textcolor{teal}{S}\textcolor{blue}{T}\textcolor{teal}{E}\textcolor{blue}{X} package](#) for [L\textcolor{blue}{A}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{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  $\text{\textcolor{blue}{S}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}}$ -annotated documents formats them into normal-looking [PDF](#).

But  $\text{\textcolor{blue}{S}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}}$  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\textcolor{blue}{A}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}](#) into a document format for (mathematical) knowledge management (MKM).

The  $\text{\textcolor{blue}{S}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}}$  system consists of the following components:

- The [\textcolor{teal}{S}\textcolor{blue}{T}\textcolor{teal}{E}\textcolor{blue}{X} package](#),
- the [Rus\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}](#) system for converting [L\textcolor{blue}{A}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{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  $\text{\textcolor{blue}{S}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}}$  document fragments and [symbols](#) for reuse, available of [mathhub.info](#), and
- the [\textcolor{teal}{S}\textcolor{blue}{T}\textcolor{teal}{E}\textcolor{blue}{X} IDE](#): A [VS Code](#) extension that, besides the usual [IDE](#) functionalities like syntax highlighting, integrates [Rus\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}](#) 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 [\textcolor{teal}{S}\textcolor{blue}{T}\textcolor{teal}{E}\textcolor{blue}{X} package](#) is all you *need*. Either way, however, we recommend using the [\textcolor{teal}{S}\textcolor{blue}{T}\textcolor{teal}{E}\textcolor{blue}{X} IDE](#) – it very much helps with semantically annotating your [L\textcolor{blue}{A}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{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](#).

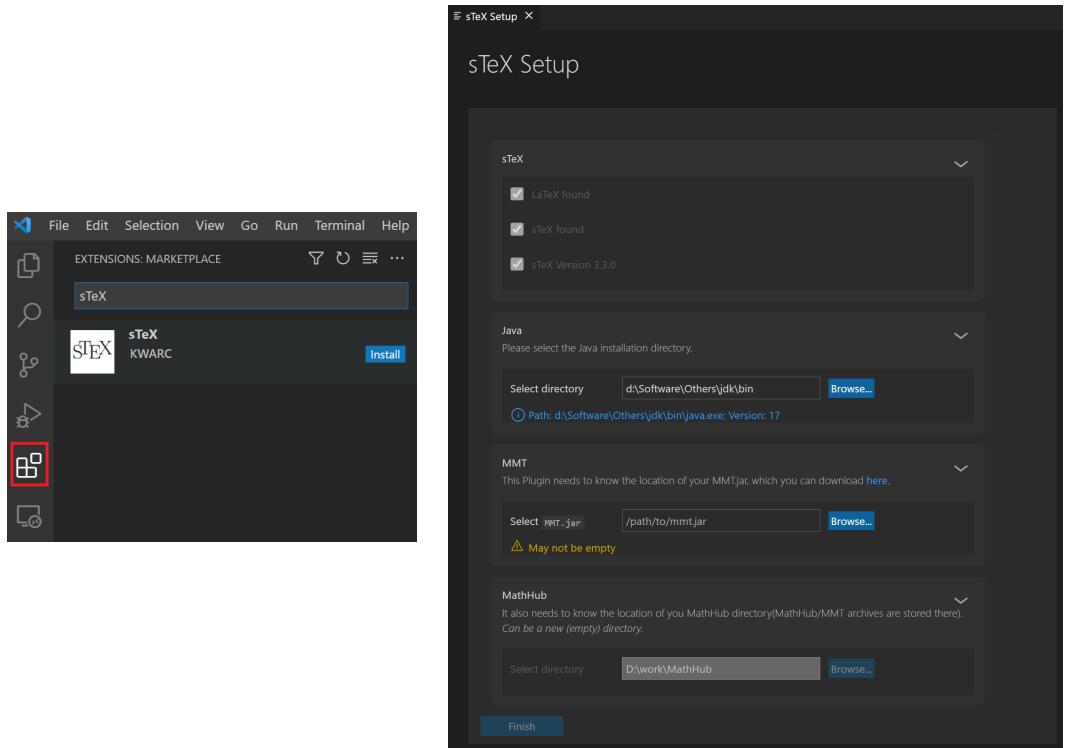


Figure 1: Installing the [sTeX IDE](#)

## 1.5 Setting Up the [sTeX IDE](#)

[sTeX](#) is based on [LaTeX](#), 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 with 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 **RuSTEX** 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://stexmmt.mathhub.info/:  
sTeX/fullhtml?archive=sTeX/Documentation&filepath=tutorial.en.xhtml](https://stexmmt.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 [sTeX](#) 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 [sTeX](#) 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 [sTeX](#) 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](#):

[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.

If you were to look at the generated [HTML](#) from this fragment, you could hover over the highlighted words ([sTeX](#), [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 [LaTeX](#)-version of the above document fragment, without any [sTeX](#) markup, could look like this:

### Example 1

Input:

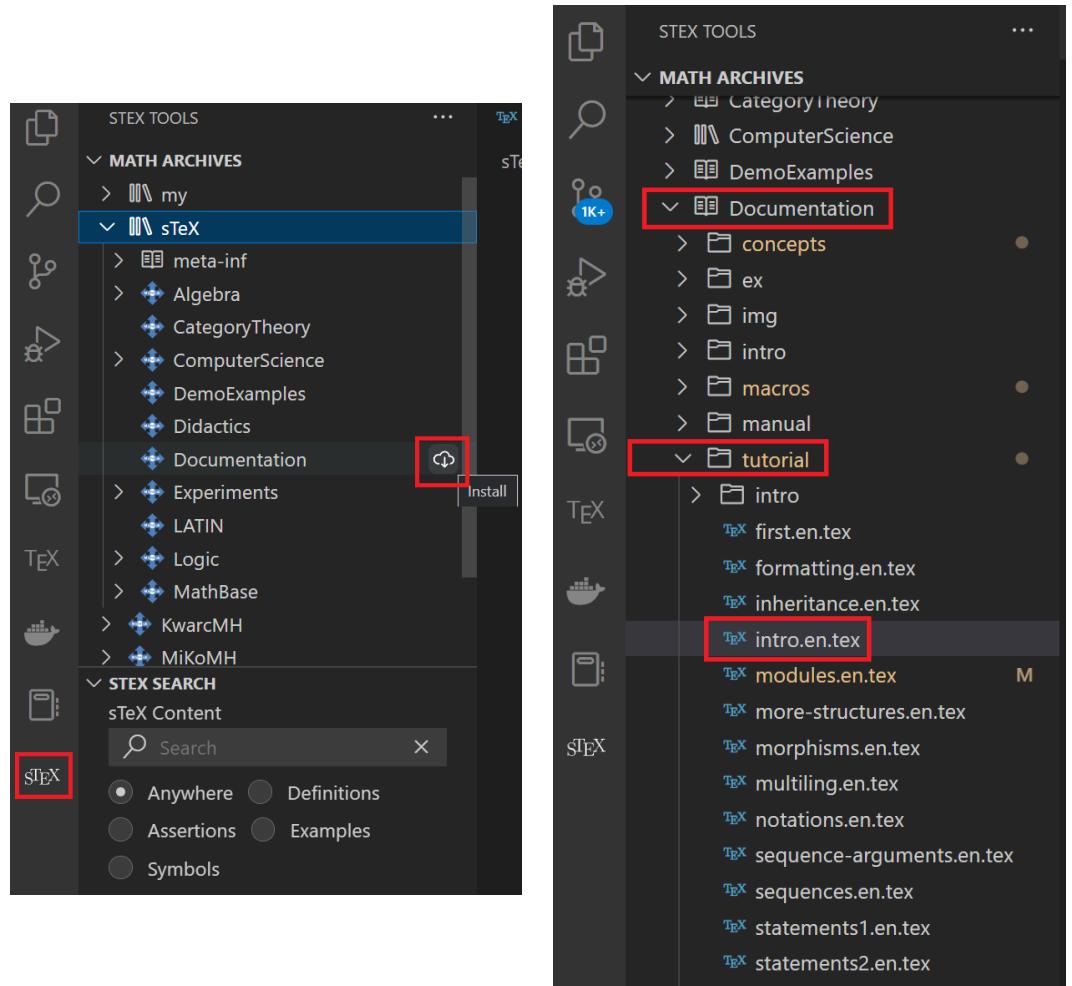


Figure 2: Installing Math Archives

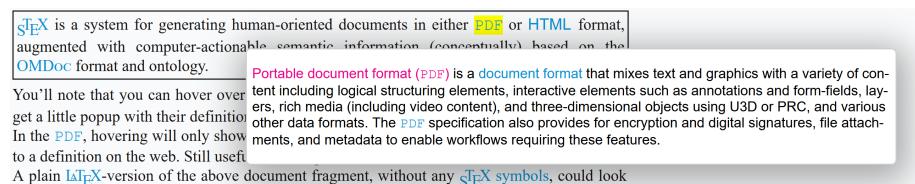


Figure 3: Definition on Hover

```

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).

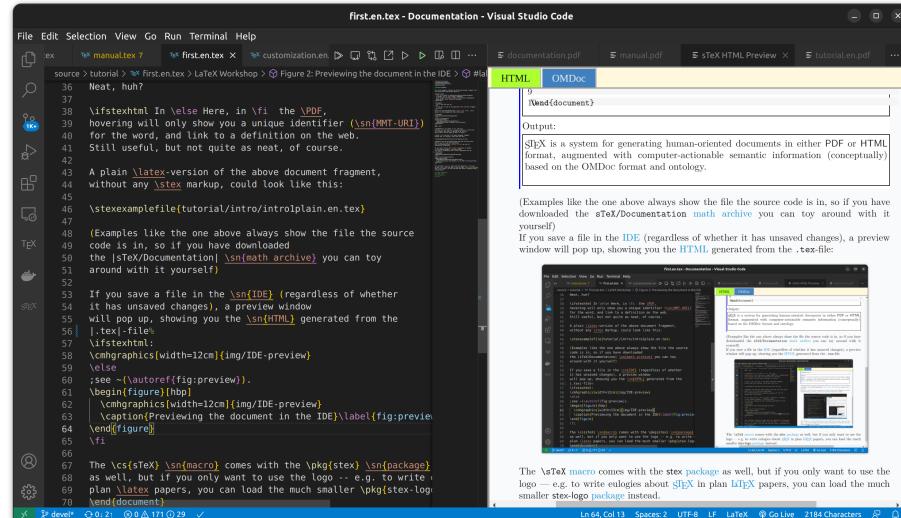


Figure 4: Previewing the document in the IDE

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 **sTeX** is that of a *symbol*:

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, **sTeX** is all about *reuse*, and naturally, there are **sTeX symbols** for all of the above already. Let's start with the one for **sTeX** itself:

### 2.1.1 Using Modules & Search in the IDE

In the **VS Code IDE**, navigate to the **sTeX**-tab on the left. In the search panel, select the “**Symbols**” radio button and search for “**sTeX**”. 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 “**sTeX**” with **semantic macro** `\stex` in the **module** `mod/systems/tex?sTeX` that is in the `\sTeX/ComputerScience/Software` archive. It produces the presentation “**sTeX**” as we want ([Figure 6](#)).

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 **sTeX 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 **\sTeX macro** – that is, of course, after we include the **stex package** first.

**sTeX** The `\usemodule` macro takes as *optional* argument the name of a **math archive**, and as a regular argument the path to an **sTeX 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:

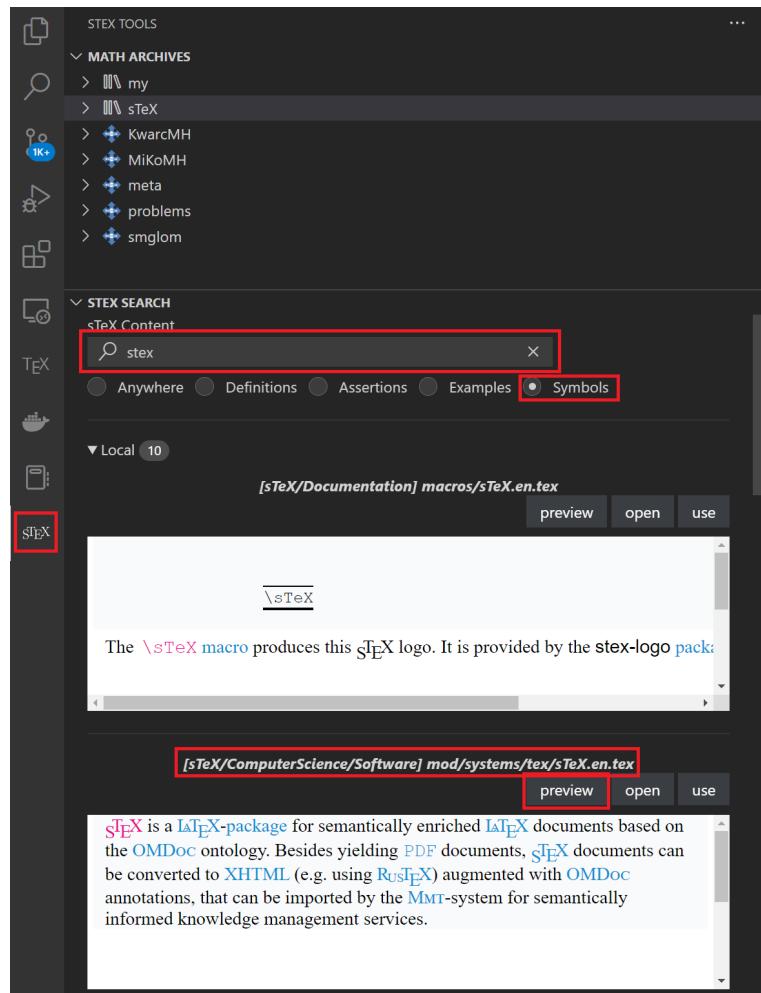


Figure 5: Search in the **STeX** IDE



Figure 6: OMDoc Preview

## Example 2

Input:

```
File [sTeX/Documentation]tutorial/intro/introstex.en.tex
1 \documentclass{article}
2 \usepackage{sstex}
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 – except for (optional/configurable) colors.

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 `sstex-highlighting` package – including that will

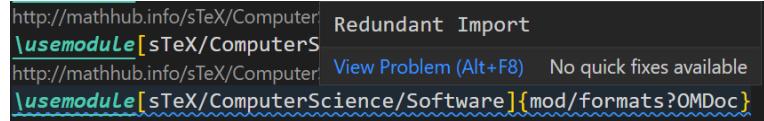


Figure 7: Redundant Imports

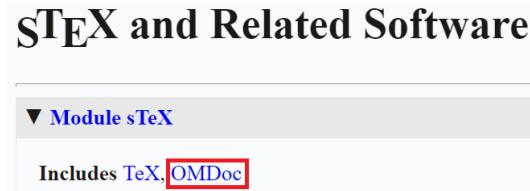


Figure 8: Includes in the OMDoc Preview

- 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 [stexmmmt.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 **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**.

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 “**L<sup>A</sup>T<sub>E</sub>X**”. But if we look at the documentation, we will note that *more* is highlighted:

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**.

The “**package**”-symbol can be found in the **L<sup>A</sup>T<sub>E</sub>X** 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 **L<sup>A</sup>T<sub>E</sub>X**, **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.

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](#):

[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 [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](#).

Firstly, note that the first paragraph would be perfectly suitable to serve as a pop-up definition on hover for the [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 [sTeX](#)-tab on the left and click the “New sTeX Archive” button ([Figure 9](#)). You will then be asked for the

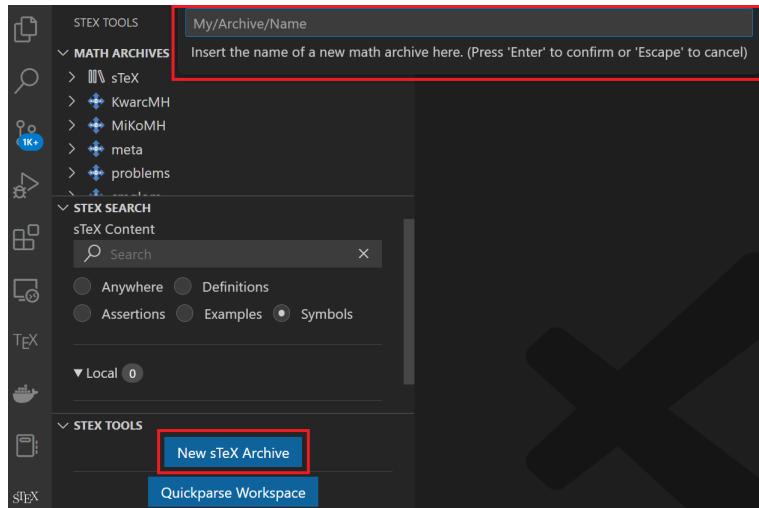


Figure 9: New Math Archive in the IDE

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 your 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
\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 `LTeX`, 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).

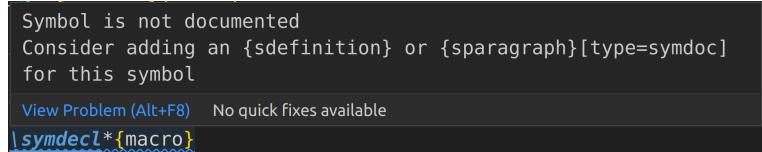


Figure 12: Undocumented Symbols

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
```

```

software developed by Donald Knuth, with a focus on
mathematical formulae. It is based on a powerful
and extensible \definename{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).

*Solution:* 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`, **STEX** 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, **STEX** 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`, **STEX** 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?

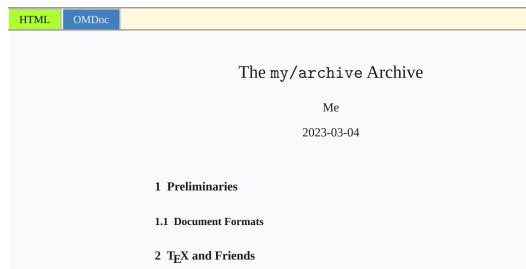


Figure 13: Missing Fragments in the **HTML** Preview

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 **\inputref**ed fragment *somewhere*. Meaning: we have to compile all of the fragments 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**.



Figure 14: The Build PDF/XHTML/OMDoc Button

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](#)).

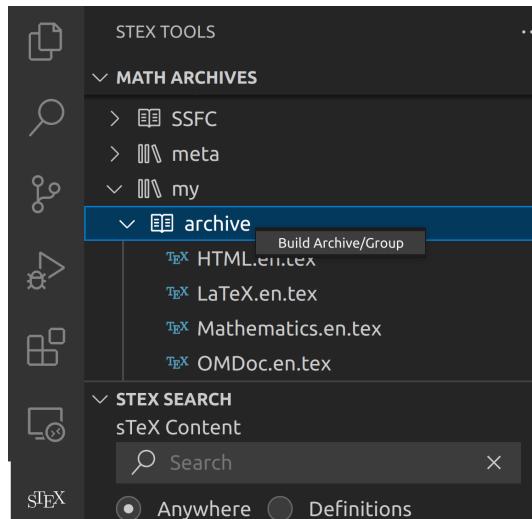


Figure 15: Building Archives in the IDE

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](#)).

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



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

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.

Also **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.

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 **STEX** 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**:  $\backslash eq{a}{b}$  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:

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/kerning 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>	( [ {
closing	<code>\mathclose</code>	) ] }
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 `$\leq{a}{b}$` 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.

---

*Solution:* 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 `$\eq{a}{b}$` and `$\eq[equiv]{a}{b}$` 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

```
$\eq{a}{b}$ or $\eq[equiv]{a}{b}$,
\eqab

invalid unit:
http://mathhub.info/sTeX/MathTutorial/mod/Equality1/Equality?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

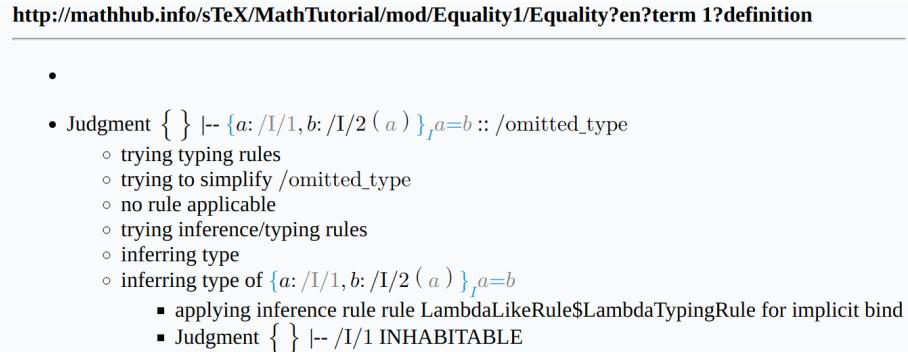


Figure 18: Type Checking Proof Tree

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 **T<sub>E</sub>X**-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 \defname{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]
    \Defname{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  $0,1,2,\dots$   
**Addition**  $a+b$  refers to the process of adding two **natural numbers**.

(like `\defname` 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 `\defname` 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}
  \Defname{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.  
Let's change that by giving the `variables` the `type`  $\mathbb{N}$ :

---

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

### Example 7

Input:

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

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, ..., a_n}{b})`

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

Figure 19: Syntax Preview

us that if we write `\funspace{a_1, ..., 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:

```
\syndef{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](#).

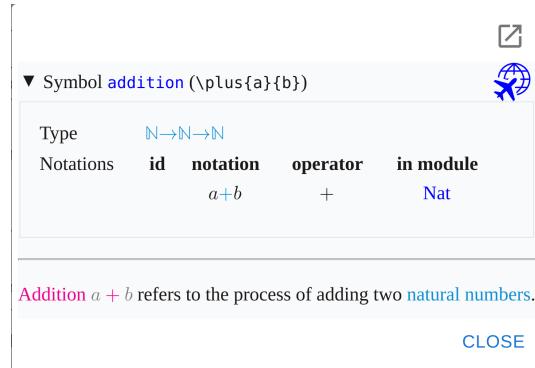


Figure 20: On-Click Popup in the `HTML`

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**

**▼ Document Elements**

- ▶ Variable `a` (`\va`) of type `N`
- ▶ Variable `b` (`\vb`) of type `N`
- ▼  $\{a: \mathbb{N}, b: \mathbb{N}\}_I^{a+b}$

Inferred Type:  $\{a: \mathbb{N}, b: \mathbb{N}\}_I^{\mathbb{N}}$

Figure 21: Inferred Type

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 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 N × N → N`<sup>2</sup>.
3. 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}{\plus{d}{e}}}}`?

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.

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,...}`,
- STEX** 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 expression, we hence do `\symdecl{add}[args=biii]`,

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

- in `\foral{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{foral}[args=Bii]`.

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

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 **sTeX** know what to do with the **sequence argument** in our **notation**. Using the `\argsep` macro, we can tell **sTeX** 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  $a \cdot b \cdot c$ .

---

*Solution:* 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 *STEX* 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}}`, *STEX* will automatically insert parentheses and yield  $a \cdot (b + c)$  – and conversely, if we do `\plus{a, \mult{b, c}}`, *STEX* 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 *STEX*, 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  $\mathbb{N}$ , `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

The screenshot shows the OMDoc tab in an HTML preview. It displays the definition of the `equal` symbol and its usage in a document element.

**Module Equality**

Includes Function

**Symbol `equal` (`\eq{a}{b}`)**

Type  $\{A: \text{SET}\}_I A \rightarrow A \rightarrow \text{Prop}$

Notations id notation operator in module

eq	$a = b$	=	Equality
equiv	$a \equiv b$	≡	Equality

**Document Elements**

Includes `Nat`

► Variable `n` (`\vn`) of type  $\mathbb{N}$

▼  $\{n: \mathbb{N}, m: \mathbb{N}\}_{\frac{n=m}{\mathbb{N}}}$

Inferred Type:  $\{n: \mathbb{N}, m: \mathbb{N}\}_{\mathbb{N}} \text{Prop}$

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  $\text{N}$  (by the  $\dots$ ),  
and
4. that it was enough to give  $\backslashvn$  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}}
```

The screenshot shows a software interface for defining mathematical structures. At the top, there's a section titled "Module Equality". Below it, under "Includes Function", is a symbol "equal" (\eq{a\_1, ..., a\_n}) with a small globe icon next to it. Under "Document Elements", there's a section for "Nat, Conjunction". It shows a conjunction of equalities: \{n: \mathbb{N}, m: \mathbb{N}, p: \mathbb{N}\} \frac{n=m \wedge m=p}{\mathbb{N}}. Below this, an "Inferred Type" is shown: \{n: \mathbb{N}, m: \mathbb{N}, p: \mathbb{N}\} \frac{}{\text{Prop}}.

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,\dots,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 `\defname`, `\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    \defname{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` `\foral` (as to not conflict with the `TeX` primitive `macro` `\forall`).

The naive approach would be to replace the “for all” by e.g. `\sr{foral}{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` `\foral` – that will take two arguments (of `modes` `B` and `i`) and produce the corresponding `notation`, so that

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

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

```
\foral{\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:

```
\foral{  
    $ \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 definientia 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 $ \mathit{fun}(\mathit{vop})\{ \mathit{vA}, \mathit{vA} \} \mathit{vA} $ is called
18  \definename{associative}, if
19  \definiens{$ \mathit{foral} \{ \mathit{arg}[2] \{ \mathit{eq}(
20    \mathit{vop} \{ (\mathit{vA}, \mathit{vb}) \}, \mathit{vc} \},
21    \mathit{vop} \{ \mathit{vA}, (\mathit{vop} \{ \mathit{vb}, \mathit{vc} \}) \}
22  } \} \mathit{comp} \{ \mathit{for all} \} \{ \mathit{arg}[1] \{ \mathit{inset} \{ \mathit{va}, \mathit{vb}, \mathit{vc} \} \mathit{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`.

*Solution:* 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:

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

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

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 `⊢` symbol after the name of the assertion? As an aside for those who are curious:

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{assertion}[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
      $\varphi(\textcolor{green}{m+1})$ also holds,
  \end{enumerate}
  then $\varphi(n)$ holds for all $\textcolor{green}{n} \in \mathbb{N}$.
\end{assertion}
```

**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 `\Rightarrow` (**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”.

---

*Solution:* 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 ⊢ ∀φ:N→Prop.φ(0)⇒( ∀m:N.φ(m)⇒φ(m+1) )⇒( ∀n:N.φ(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`. Its 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 \defname{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
$\$\\vM!$$	$M$
$\$\\vM{}$$	$\langle U_M, o_M \rangle$
$\$\\vM{univ}{}$$	$U_M$
$\$\\vM{op}!$$	$o_M$
$\$\\vM{op}{a,b,c}{}$$	$a o_M b o_M c$

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  $a o_M b$ .

We can change the way the `\maincomp` notation component is modified, by using the optional argument `copm=` 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  $M := \langle U, o \rangle$  and  $M' := \langle U', o' \rangle$  `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  $M := \langle A, o \rangle$  and  $M' := \langle A, o' \rangle$  `magmas`.

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

```
\vaseq{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  $(M_i)_1^n := \langle U_i, o_i \rangle_1^n$  a sequence of `magmas`...

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 `sTeX` 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 \definename{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*.

#### Conservative Extensions

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

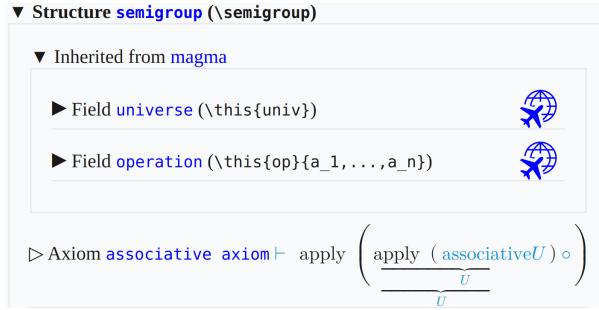


Figure 27: Axioms in OMDoc

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]{\#1^{\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., 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 `magmas` 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` distict 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 maningful `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 \defname{magmahom} is a function
21  $\fun{\f!}{\dom{univ}}{\cod{univ}}$ such that
22  \inlineass[name=homomorphism condition]{\conclusion{\forall{
23    $arg[2]{\leq[
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
$\$\\vh!$$	$H$
$\$\\vh{}$$	$\langle M_1, M_2, H \rangle$
$\$\\vh{f}!$$	$H$
$\$\\vh{f}{a}$$	$H(a)$
$\$\\vh{dom}!$$	$M_1$
$\$\\vh{cod}{}$$	$\langle U_2, o_2 \rangle$
$\$\\vh{cod}{univ}$$	$U_2$
$\$\\vh{dom}{op}!$$	$o_1$
$\$\\vh{cod}{op}{a,b,c}$$	$a o_2 b o_2 c$

Note how – as one would expect – we can treat  $\backslash vh\{dom\}$  and  $\backslash vh\{cod\}$  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 `\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 `{...}`, 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 [sTeX](#).

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 **V** absorbs **Λ** and vice versa; i.e.  $a \vee (a \wedge b) = a$  and  $a \wedge (a \vee b) = a$ .  
The operations **Λ** and **V** 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 **meetsl** and **joinsl**. 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 **meetsl**:

```
\copymod{meetsl}{semilattice}{
    op @ meet
}
\copymod{joinsl}{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 **joinsl** is now called **joinsl/universe**, and the one in **meetsl** is called **meetsl/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{meetsl}{semilattice}{
    univ @ univ,
    op @ [meet] meet
}
\copymod{joinsl}{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 `definientia` 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{meets1}[
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 $mysl{order}{a,b} \qquad \mysl{}[\univ,\op,order]$
```

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 individual somewhere in the body of the `environment`, interleaved with arbitrary text. Additionally, within the `environment`, all `STEX` features that introduce *definientia* (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}{meets1}
\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{meets1}
\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 “definientia” (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.

### 4.1 Slides and Course Notes

TODO<sup>6</sup>

### 4.2 Problems and Exercises

TODO<sup>7</sup>

### 4.3 Exams

TODO<sup>8</sup>

---

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

<sup>7</sup>TODO: problem.sty

<sup>8</sup>TODO: hwexam.sty

## Part II

# User Manual

*The dynamic [HTML](#) version of this part can be found at  
<https://stexmmt.mathhub.info/>:  
<https://stexmmt.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, **S<sub>T</sub>E<sub>X</sub>** 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 `STEX`3 features,
- `smgloM/*` – a vast collection of multilingual `modules` of concepts in mathematics and computer science. The SMGloM predates `STEX`3 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://stexmmt.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`/`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 the library is found.

A good practice is to have individual `sTeX` 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` The `\sTeX` macro produces this S<sup>I</sup>TeX logo. It is provided by the `sTEX-logo` package, `\stex` included with the `stex` package.

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

`\STEXinvisible` `\STEXinvisible{\langle code \rangle}`

Processes `\langle code \rangle`, but does not generate any output. In the `HTML`, `\langle code \rangle` 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, **STEX** 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** 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.

---

`\lstinputmhlisting` `\lstinputmhlisting` 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>9</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).

↳ An `STEX module` corresponds to an `MMT/OMDOC theory`. As such  
↳ it gets assigned an `MMT-URI` (*universal resource identifier*) of the form  
↳ `~T~> <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.

→ `\symdecl` introduces a new `MMT/OMDoc constant` in the current `module` (i.e. → `MMT/OMDoc theory`). Correspondingly, they get assigned the `MMT-URI` ↵ `<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` `TODO10`.

---

`\textsymdecl` `\textsymdecl{<symbol>}[<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` `\<symbol>name` is generated that produces `<code>` without any semantic markup.

---

`\symdef` `\symdef{<symbol>}[<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. `STEX` will replace `#1` and `#2` in `<code>` by `a` and `b`, yielding `<retcode>`.
2. `STEX` 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>10</sup>TODO: experimental

This is as good a place as any other to explain how **STEX** resolves a string `symbol` to an actual `symbol`.

If `\symbol` is a `semantic macro`, then **STEX** 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.... **STEX** attempts to handle this case thusly:



If `symbol` does *not* correspond to a `semantic macro` `\symbol` and does *not* contain a ?, then **STEX** checks all `symbols` currently in scope until it finds one, whose name is `symbol`. If `symbol` is of the form `pre?name`, **STEX** 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.

M → `\symref{\symbol}{\text}`    in    MMT/OMDoc    generates    the    term  
 M → `<OMS name="\symbol URI"/>`.  
 ~T →

---

`\symname`    `\symname[pre=\pre,post=\post]{\symbol}`

`\sn`    `\symname[pre=\pre,post=\post]{\symbol}`

`\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 `\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`    `\srefsymuri{\symbol}{\text}`

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 `\neginfpref` (negative infinity).
- `prec=nobrackets` is an abbreviation for `prec=\neginfpref;\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 conceptionally meaningful:

The underlying principle is that the arguments to a `semantic macro` represent *arguments to the mathematical operation* represented by a `symbol`. For example, a `semantic macro` 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 `STEX` 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 `LATEX` `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` and `\neginfprec` represent *infinitely large* and *infinitely small precedences*, respectively.



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

When `STEX` steps into an argument of a `semantic macro`, it sets  $p_d$  to the respective `argument precedence` of the `notation` used.

##### Example 16

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. `STEX` starts out with  $p_d = \infprec$ .
2. `STEX` encounters `\plus` with  $p_{op} = 100$ . Since  $100 \not> \infprec$ , it inserts no parentheses.
3. Next, `STEX` encounters the two arguments for `\plus`. Both have no specifically provided `argument precedence`, so `STEX` uses  $p_d = p_{op} = 100$  for both and recurses.
4. Next, `STEX` 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 `STEX` again inserts no parentheses.

6. Since the **notation** of `\mult` has no explicitly set **argument precedences**, **STEX** again uses the **operator precedence** for the arguments of `\mult`, hence sets  $p_d = p_{op} = 50$  and recurses.
7. Next, **STEX** encounters the inner `\plus{c, ...}` 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 **STEX** to insert parentheses, and we proceed as before.

---

**\dobracket** `\dobraackets{(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:

---

**\argssep** `\argssep{(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** `TODO11`

#### 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!{\langle arg\rangle}` marks up `\langle arg\rangle` similarly to how `\symref{\smacro}{\langle arg\rangle}` would.

Without the `!`, `\smacro` still only takes a single argument, but it is expected, that within `\langle arg\rangle`, the arguments for the `symbol` are explicitly marked up. The `\comp` macro is allowed in `\langle arg\rangle` to determine the components of `\langle arg\rangle` 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`.

↳ 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.  
~T~ 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{\langle module\rangle}` 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{\langle module\rangle}` 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{\langle module\rangle}` 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.

$\hookrightarrow M \rightarrow$  In **MMT**, every **document** and every **module** induces an **MMT theory**. `\usemodule`  
 $\dashv M \rightarrow$  induces and **MMT include** in the document **theory**, `\importmodule` and  
 $\rightsquigarrow T \rightsquigarrow \requiremodule$  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{\langle code \rangle}{\langle sequence \rangle}`

Maps the function  $\langle code \rangle$  (containing #1) over every element of the  $\langle sequence \rangle$ .  
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}{\langle mname \rangle}[\langle name \rangle, this=\langle code \rangle]` opens a new mathematical structure with name  $\langle mname \rangle$  (if provided) or  $\langle mname \rangle$  (otherwise), and semantic macro `\mname`. It subsequently behaves like the `smodule` environment.

\this

The optional `this=\langle code \rangle` 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.)`

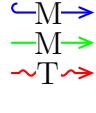
`\begin{extstructure}{\langle mname \rangle}[\langle name \rangle, this=\langle code \rangle]{\langle structs \rangle}` opens a new mathematical structure extending the structures given in  $\langle structs \rangle$  (a comma-separated list of names).

`extstructure* (env.)`

`\begin{extstructure*}{\langle struct \rangle}` opens a new mathematical structure conservatively extending the (single) structure  $\langle struct \rangle$ . Conservative meaning: Every symbol newly introduced in this structure needs to have a definiens. The new symbols are attached as fields directly to  $\langle struct \rangle$ .

\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]):  
  
`\begin{mathstructure}{\langle name \rangle}` creates a nested theory with name  $\langle name \rangle$ -module. The constant  $\langle name \rangle$  is defined as a dependent record type with manifest fields, the fields of which are generated from (and correspond to) the constants in  $\langle name \rangle$ -module.

### 7.7.1 Semantic Macros for Structures

Assume we have a mathematical structure with semantic macro `\struct`:

#### Example 17

```
\begin{mathstructure}{\struct}
\symdef{fielda}{a}
\symdef{fieldb}[args=2]{#1 \maincomp{b} #2}
\symdef{fieldc}[args=2,def=\sn{fieldb}]{#1 \maincomp{c} #2}
```

```

\inliness[name=axiom1]{\conclusion{some axiom}}
\end{mathstructure}
\notation{struct}{StRuCt}

```

- If `\struct` has no `notations`, then `\struct!` produces  $\langle a, b, c \rangle$ . Otherwise, it produces the notation, i.e. `StRuCt`. In both cases, `\struct{}{}` produces  $\langle a, b, c \rangle$  (for reasons that will become clearer in a moment).
- `\struct{}{}` (or `\struct!` in the case where no `notations` are around) can be modified in the following ways:
  - `\struct{}{{}}[\langle fieldname \rangle, ...]` lets you pick, which precise fields to show, so e.g. `\struct{}{{}}[fielda, fieldb]` produces  $\langle a, b \rangle$ . By default, `\struct{}{}` shows exactly the fields that have `semantic macros` (which are also used to access the fields in `\struct{{}}{\langle fieldname \rangle}`).
  - `\struct{}{{}}[\langle id \rangle]` lets you pick the `notation` of the “`mathematical structure`” symbol to use to typeset the `structure`; e.g. `\struct{}{{}}[parens]` yields  $\langle a, b, c \rangle$ , and (combining both) `\struct{}{{}}[fielda, fieldb][angle]` yields  $\langle a, b \rangle$ .
- The two arguments in `\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, `StTeX` will present all of them (that are not assertions). Hence, `\struct{S}{{}}` yields  $\langle a_S, b_S, c_S \rangle$ , `\struct{S}{fielda}` produces  $a_S$ , `\struct{}{fielda}` produces  $a$ , and `\struct{S}{fieldb}{x}{y}` produces  $x b_S y$ .
- For the sake of completion, `\struct{first}!` simply produces the given argument; e.g. `\struct{S}!` simply produces  $S$ .

More precisely: `\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 `StTeX` does when using `\struct` in the `return` of a `symbol` (or `variable`).

Continuing:

- `\struct[comp=\langle code \rangle]{...}{...}` 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_{\this}`. For example, `\struct[comp=\#1^{\langle Foo \rangle}]{S}{{}` produces  $\langle a^{F\!oo}, b^{F\!oo}, c^{F\!oo} \rangle$ , and `\struct[comp=\#1^{\langle \this \rangle}]{S}{fieldb}{x}{y}` yields  $x b_S y$ .
- `\struct[this=\langle code \rangle]{...}{...}` modifies the way `\this` is being typeset; i.e. the presentation of the first `{...}` argument. For example `\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, `\struct[\langle fieldname \rangle=\langle code \rangle]{...}{...}` assigns the field  $\langle fieldname \rangle$  to the term  $\langle code \rangle$ .  $\langle code \rangle$  is subsequently used when using `{...}{}`, but not in fields directly. For example, `\struct[fielda=A]{S}{{}}` produces  $\langle A!, b_S, c_S \rangle$ , but `\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 `\definename`) 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 (or counterexamples).

**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 to 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`, `sdefinition` and `sexample` environments respectively, but take the text to annotate as 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{\langle cls \rangle}`

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 highlighting for `definienda` and adds the referenced `symbol` to the `for=` list of the environment.  
\defname `\defname` is to `\definiendum` as `\symname` is to `\symref`. Analogously, `\Defname` 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 `sdefinition`.

If the `sdefinition` environment has several elements in its `for` list, an optional argument `\definiens[\langle symbol \rangle]{...}` 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).

Secondly, 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=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  
\conclusion

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*.

If the `sassertion` environment has several elements in its `for` list, an optional argument `\conclusion[⟨symbol⟩]{...}` can be used to tell `STEX` 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 `STEX` 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>12</sup>

---

<sup>12</sup>TODO: proofs

# Chapter 9

# Customizing Typesetting

There are two kinds of typesetting that can be customized in **STEX**: **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  
`\compemph@uri` (marked with `\comp` or `\maincomp`) in `notations`.

---

`\defemph` Like `\symrefemph` and `\symrefemph@uri`, but governs the highlighting of definienda  
`\defemph@uri` marked with `\definiendum` (or `\definename`).

---

`\varemph` Like `\compemph` and `\compemph@uri`, but governs the highlighting of components (marked  
`\varemph@uri` with `\comp` or `\maincomp`) in the `notations` of `variables`.  
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 *stylistable* using the `macros` `\stextstyle{name}[\langle style \rangle]{...}`. These *stylistable 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)

```
\stextstyledefinition{\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

```
\stextstyleassertion[theorem]{\begin{theorem}}{\end{theorem}}
\stextstyleassertion[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`:

```
\stextstyleassertion[theorem]{\ifx\thistitle\empty
  \begin{theorem}\else\begin{theorem}[\thistitle]\fi
  \end{theorem}}
```

---

```
\stexstylemodule           TODO13
\stexstylecopymodule
\stexstyleinterpretmodule
\stexstylerealization
\stexstylemathstructure
\stexstyleextstructure
\stexstyledefinition
\stexstyleassertion
\stexstyleexample
\stexstyleparagraph
\stexstyleproof
\stexstylesubproof
```

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
\stexstyleextsymdecl
\stexstylenotation
\stexstylevarnotation
\stexstylesymdef
\stexstylevardef
\stexstylevarseq
\stexstylepsfsketch
\stexstyleMMTinclude
```

### 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 `STEX` 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` 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` If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated `fiboxed` 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 **\notestrue** 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.



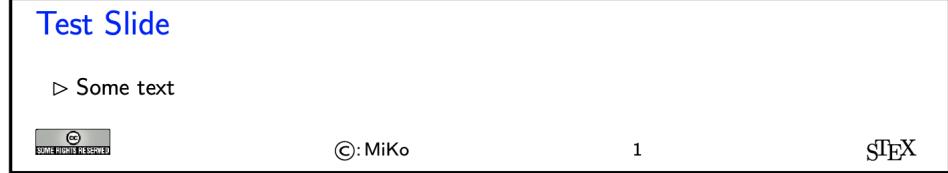
The **beamer** class recommends not to use the **allowframebreaks** option on frames (even though it is very convenient). This holds even more in the **notesslides** case: At least in conjunction with **\newpage**, frame numbering behaves funny (we have tried to fix this, but who knows).

`\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 `beamterthemesTeX`. It is assumed as the default theme for `sTeX`-based notes and slides. The result in `notes` mode (which is like the `slides` version except that the slide height is variable) is



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

`\setslidelogo` The default logo provided by the `notesslides` package is the `sTeX` logo it can be customized using `\setslidelogo{\langle logo name \rangle}`.

`\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{\langle name \rangle}` 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[\langle url \rangle]{\langle logo name \rangle}` is used for customization, where `\langle url \rangle` 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 `sTeX` notes.

---

```
\frameimage  
\mhframeimage
```

In this case we can use `\frameimage[⟨opt⟩]{⟨path⟩}`, where `⟨opt⟩` are the options of `\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 document management features in `MathHub`.

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

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

---

```
\textwarning
```

The `\textwarning` macro generates a warning sign: 

### 10.1.6 Ending Documents Prematurely

---

```
\prematurestop  
\afterprematurestop
```

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

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

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the S<sup>T</sup>E<sub>X</sub> 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** `\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 `\inputref{<path>}`. In this way, the `\printexcursions` macro (usually in the appendix) will collect up all excursions that are specified in the main text.

Sometimes, we want to reference – in an excursion – part of another. We can use `\excursionref{<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<sup>15</sup>. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

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

### 10.2.2 Problems and Solutions

---

`solutions` The `problem` package takes the options `solutions` (should solutions be output?), `notes` (should the problem notes be presented?), `hints` (do we give the hints?), `gnotes` (do we show grading notes?), `pts` (do we display the points awarded for solving the problem?), `min` (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.

`boxed` The `boxed` option specifies that problems should be formatted in framed boxes so that they are more visible in the text. Finally, the `test` option signifies that we are in a test situation, so this option does not show the solutions (of course), but leaves space for the students to solve them.

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

#### Example 18

Input:

---

<sup>15</sup>for the moment multiple choice problems are not supported, but may well be in a future version

```

1 \documentclass{article}
2 \usepackage[solutions,hints,pts,min]{problem}
3 \begin{document}
4   \begin{sproblem}[id=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}[for=elefants]
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?

*Solution for=elefants:* 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 referenced **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.

---

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

---

**\if solutions** 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 **\if solutions** 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` `\mcc[<keyvals>]{<text>}` takes an optional key/value argument `<keyvals>` for choice metadata and a required argument `<text>` for the proposed answer text. The following keys are supported

- `T` 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 19

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=Noooooooooooo, 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 – Correct  
*that is for C and C++*
- function – Wrong  
*that is for Standard ML*
- fun – Wrong  
*that is for Java*
- public static void – Nooooooooooooo  
*that is for Java*

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

### Example 20

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 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:

```
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>16</sup>

<sup>16</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.

### 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 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.

```
\testspace      \testspace takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: \testsmallspace, \testmediumspace, \testlargepage \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.  
\testemptypage Local Variables: mode: latex TeX-master: ../stex-manual End:  
          LocalWords: solutions,notes,hints,gnotes,pts,min,boxed,test gnotes elefants,pts gnote  
          LocalWords: 2,title exnote hint,exnote,gnote ifsolutions mcb keyvals Ttext Ftext Local-  
          Words: Functions,name F,feedback Noooooooooo,feedback 2,title includeproblem Local-  
          Words: elefants.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

---

**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).

**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 L<sup>A</sup>T<sub>E</sub>X 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 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 due 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 `min` of minutes, and `reqpts` the points that are required for a perfect grade.

```
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)

2023-03-19

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

Write the solutions to the sheet.

The estimated time for solving this exam is 10 minutes, leaving you 50 minutes for revising your exam.

You can reach 10 points if you solve all problems. You will only need 27 points for a perfect score, i.e. -17 points are bonus points.

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

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

	To be used for grading, do not write here									
prob.	4.1	1.1	1.2	2.1	2.2	2.3	2.4	2.5	Sur	
total	0	0	0	10	0	0	0	0	0	10
reached										

good luck

17

Local Variables: mode: latex TeX-master: .../stex-manual End:

LocalWords: hwexam solutions,notes,hints,gnotes,pts,min gnotes testemptytypepage reqpts 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`.

---

<sup>17</sup>MK: The first three “problems” come from the stex examples above, how do we get rid of this?

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

```
1 \documentclass{standalone}
2 \usepackage{tikz}
3 \usetikzlibrary{...}
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` `\cmhtikzinput` `\mhtikzinput` 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 `mhrepos=` key. Again, `\cmhtikzinput` 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 {⟨prefix⟩} {⟨msg⟩}`

Logs the debug message {⟨msg⟩} under the prefix {⟨prefix⟩}. A message is shown if its prefix is in a list of prefixes given either via the package option `debug=⟨prefixes⟩` or the environment variable `STEX_DEBUG=⟨prefixes⟩`, where the latter overrides the former.

---

`\_stex_do_deprecation:n` TODO `\_stex_do_deprecation:n`

## 11.1 Documents

`\l_stex_docheader_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_docheader_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:nnnn`).

---

`\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 TeX 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 **STEX** 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·`<arity>`),  
#5 : `<definiens>` (or empty),  
#6 : `<type>` (or empty),  
#7 : `<return code>` (or empty), and  
#8 : `<\invocation_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.

**TeXhackers note:** `\invocation_macro` must be `\protected`.

---

`\stex_iterate_symbols:n`  
`\stex_iterate_break:`

`\stex_iterate_symbols:n {<code>}`

`\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>} {<\invocation_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: * whether to typeset declaration components (notations, types, definientia etc.) in a throw-  
\stex_if_check_terms:TF *
```

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 `j{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 {{maybename}}`

checks whether `\l_stex_key_name_str` is empty, and if so, sets it to be `<maybename>`. 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_mroname_str`.

Variables work very similar to symbols, except that their declarations are local to the current 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 `<name>` and value `{{id}}{{name}}{{arity}}{{args}}{{definiens}}{{type}}{{return code}}{{\invokation_macro}}`.  
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`  
 `{{symboluri}}{{id}}{{arity}}{{code}}{{op code}}`

stores the arguments in the property list `\c_stex_module_{\l_stex_current_module}_notations_prop` under key `<symboluri>!<id>` 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 correponding 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  $\{⟨symboluri⟩\}{⟨id⟩}{⟨arity⟩}{⟨code⟩}{⟨op code⟩}$ .

---

```
\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 *precs*-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  $\backslash notation$ -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
\STEXInternalSymbolAfterInvocationTL
```

---

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 `\STEXInternalSymbolAfterInvocationTL`.

How this works is best explained by example.

### Example 23

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 und j the index), the precedence (20 or 40 respectively), and the parameter token.

Finally, a `\STEXInternalSymbolAfterInvocationTL` 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 }\STEXInternalSymbolAfterInvocationTL

```

---

`\stex_notation_top:nw` `\stex_notation_top:nw {<symboluri>}{<code>}`

calls `\stex_notation_parse_and_then:nw{<code>}` and, adds the notation for the symbol with URI `<symboluri>` 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:nn {<name>}{<typeid>}`  
`\stex_structural_feature_module_end:` `\g_stex_last_feature_str`

opens a new module-like structural feature of type `<typeid>` with name `<name>`, which needs to be closed with `\stex_structural_feature_module_end:`.

Its body behaves like a nested module with name `<modulename>/<name>`, 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_end: \stex_structural_feature_morphism:nnnnn
{\morphisname}{\typeid}{\archive}{\domain}{\annotations}
\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  $\langle morphisname \rangle$  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 `\renamedecl`.

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  $\langle morphisname \rangle / \langle name \rangle$  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 STEX 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_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

---

<code>\_stex_invoke_symbol:nnnnnnnN</code>	<code>\l_stex_current_symbol_str</code>	<code>\_stex_invoke_symbol:nnnnnnnN</code>
<code>\l_stex_current_arity_str</code>		<code>\{\langle module \rangle\}\{\langle name \rangle\}\{\langle arity \rangle\}\{\langle args \rangle\}\{\langle definiens \rangle\}</code>
<code>\l_stex_current_args_tl</code>		<code>\{\langle type \rangle\}</code>
<code>\l_stex_current_return_tl</code>		<code>\{\langle return code \rangle\}</code>
<code>\l_stex_current_type_tl</code>		<code>\{\langle invocation_macro \rangle\}</code>

---

is how a semantic macro is/should be defined. `\_stex_invoke_symbol:nnnnnnnN` 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 `\STEXInternalSymbolAfterInvocationTL`, defines the following macros for all of its arguments, and subsequently calls the `\invocation_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 `\invocation_macro` is `\stex_invoke_symbol`:

---

```
\_stex_invoke_variable:nnnnnnN
```

analogous to `\_stex_invoke_symbol:nnnnnnnN`, but for variables; sets the `\comp`-controlled highlighting to `\varemph@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  $\langle return \rangle\{\l_stex_invoke_symbol!\}$ .
- If text:

---

```
\stex_invoke_sequence_range:  
\stex_invoke_sequence_in:
```

TODO

## 11.8 Optional (Key-Value) Argument Handling

LATEX3 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_set:nn
```

---

```
\stex_keys_define:nnnn {\langle id\rangle}{\langle setup code\rangle}  
{\langle keyval setup code\rangle}{\langle parents\rangle}
```

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 24

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 Stylistable Commands and Environments

---

```
\stex_new_stylistable_cmd:nnnn \stex_new_stylistable_cmd:nnnn {<name>} {<arity>} {<code>}
{<default>}
```

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  $\text{\stexstyle}\langle name \rangle$ . On calling  $\text{\stex_style_apply}$ : executes the presentation code provided by a user.

$\langle code \rangle$  should:

- Call  $\text{\stex_keys_set:nn}\{style\} \dots$  (or a keyset inheriting from  $style$ ),
- set macros with prefix  $\backslash this\dots$  that a user might want to use for presentation (e.g.  $\backslash thistitle$ ),
- call  $\text{\stex_style_apply}$ : at some point.

---

```
\stex_new_stylistable_env:nnnnnnn \stex_new_stylistable_env:nnnnnnn {<name>} {<arity>} {<begincode>}
{<endcode>} {<default begin>} {<default end>} {<prefix>}
```

Like  $\text{\stex_new_stylistable_cmd:nnnn}$ , but defines a new environment  $\langle prefix \rangle \langle name \rangle$  stylistable via  $\text{\stexstyle}\langle name \rangle$ . Should call  $\text{\stex_style_apply}$ : twice; once in the  $\langle begincode \rangle$  and once in  $\langle endcode \rangle$ .

---

**\stex\_style\_apply:** Sets  $\backslash thisstyle$  to be the head of the CSL  $\text{\l_stex_key_style_clist}$  and checks whether a style for the current stylistable macro/environment has been defined; if not, executes the code for the default style.

### Example 25

$\text{\importmodule}$  is defined something like the following:

```
1 \stex_new_stylistable_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  $\text{\importmodule}$  (by default none) via  $\text{\stexstyleimportmodule}\{...\} \text{\thismodulename}\{...\}$ .

### Example 26

$\text{\smodule}$  does something like

```
1 \stex_new_stylistable_env:nnnnnnn{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 `\stextylemodule`.

## 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)}`

---

computes the absolute file path of `(pathstring)` relative to the source-folder of `(archive-id)` (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)}`

---

looks for a math archive `(id)` 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 `(code)` in the context of math archive `(opt-id)` (using `\stex_require_archive:n`), i.e. iff `(opt-id)` is non-empty, changes the current archive to the one with id `(opt-id)`, call `(code)` with `(opt-id)` as argument (in #1) and changes it back afterwards.

If `(opt-id)` is empty, `(code)` 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

**STEX** 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, STEX 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:on    \stex\_file\_in\_smsmode:nn {\<filestring>} {\<setup-code>}  
sets up sms-mode and internal grouping, calls *<setup-code>* and subsequently processes the file *<filestring>* in sms-mode as described above.

### 11.11.1 Second Pass

---

\stex\_sms\_allow:N    \stex\_sms\_allow:N {\<macro>}

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 {<\macro>}`

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 {<envname>}`

registers the environment `<envname>` to be allowed in sms mode.

As with `\stex_sms_allow_escape:N`, the `\begin{<envname>}` is escaped, hence the begin-code of the environment needs to call `\stex_smsmode_do:`. Since `\end{<envname>}` 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 *`

`\stex_str_if_starts_with:nn {<first>} {<second>}`

Checks whether the string `<first>` starts with the string `<second>` (i.e. `<second>` is a prefix of `<first>`).

---

`\stex_str_if_ends_with_p:nn *`

`\stex_str_if_ends_with:nn {<first>} {<second>}`

Checks whether the string `<first>` ends with the string `<second>` (i.e. `<second>` is a suffix of `<first>`).

### 11.12.1 File Paths

*File paths* are represented as L<sup>A</sup>T<sub>E</sub>X3 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>X3 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>X3 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:NNo` `\stex_uri_add_module:NNn {\langle target \rangle}{\langle uri \rangle}{\langle name \rangle}`

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_abrevs_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_abrevs_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 {<abbrev>}`

Sets `\l_stex_current_language_str`, and, if the `babel` package is loaded, calls `\selectlanguage` on the language corresponding to `{<abbrev>}`.

Note that the package option `lang=` automatically loads the `babel` package.  
 If `<abbrev>=tr`, additionally call `\selectlanguage` with the option `shorthands=:!`.  
 Throws `error/unknownlanguage` if no language with abbreviation `{<abbrev>}` 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 {<code>}`

turns annotations off temporarily in `<code>` (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**) or not (e.g. `pdflatex`).  
`\stex_if_html_backend:TF *`

`\ifstexhtml` is set accordingly.

---

`\stex_annotate:nnn`

`\stex_annotate:nnn {<attr>} {<value>} {<code>}`

In **HTML** mode, annotates the output of `<code>` with the **XML**-attribute `<attr>="<value>"`.  
 In **PDF** mode, just calls `<code>`.

---

\stex\_annotation\_invisible:n  
\stex\_annotation\_invisible:n

\stex\_annotation\_invisible:n {⟨code⟩}

Should annotate ⟨code⟩ with `shtml:visible="false" style="display:none;"`. In PDF mode, does nothing.

\stex\_annotation\_invisible:nnn combines \stex\_annotation\_invisible:n and \stex\_annotation:nnn.

`stex_annotation_env (env.)`    \begin{stex\_annotation\_env}{⟨attr⟩}{⟨value⟩} ⟨code⟩ \end{stex\_annotation\_env}  
should behave like \stex\_annotation:nnn{⟨attr⟩}{⟨value⟩}{⟨code⟩}

---

\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 {⟨code⟩}

writes ⟨code⟩ to the `\jobname.sms`-file, if `writesms` is active.

**TeXhackers note:** ⟨code⟩ is being read with `expl3` category codes (except for spaces having catcode 10), but not pretokenized; i.e. ⟨code⟩ 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}`.

**TeXhackers note:** Does not work with “higher” parameter tokens, i.e. `##1`, `####1` etc.

---

`\stex_deactivate_macro:Nn` `\stex_reactivate_macro:N`

`\stex_deactivate_macro:Nn {(\macro)} {(msg)}`

Makes `\macro` throw an error message `error/deactivated-macro{(msg)}`, 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 {(\macro)} {(args)}`

Calls “`kpsewhich args`” and stores the result in `\macro`,

**TeXhackers note:** Does not require `shell-escape`

---

`\stex_get_env:Nn`

`\stex_get_env:Nn {(\macro)} {(envvar)}`

Stores the value of the environment variable `(envvar)` in `\macro`.

---

`\stex_fatal_error:n` `\stex_fatal_error:nn` `\stex_fatal_error:nxx`

Mimic the `\msg_error:-macros`, but make sure that TeX stops processing.

**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:nn`

`\stex_pseudogroup_with:nn {(\macros)}{(\code)}`

Calls `(code)` and subsequently restores the values of the `(macros)` given.

**TeXhackers note:** Does *not* work recursively!

---

`\stex_pseudogroup:nn`

`\stex_pseudogroup:nn {(\code1)}{(\code2)}`

Expands `(code2)`, and inserts the result after `(code1)`. 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 {(\macro)}`

### Example 27

```
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:o
```

`\stex_metagroup_new:n {⟨id⟩}`

Opens a new metagroup at the current TeX group level with identifier `⟨id⟩`.

---

```
\stex_metagroup_do_in:nn  
\stex_metagroup_do_in:nx
```

`\stex_metagroup_do_in:nn {⟨id⟩}{⟨code⟩}`

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}{2023/03/19}{3.3.0}{sTeX document class}
6
7  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
8  \ProcessOptions
9
10 \RequirePackage{stex}
11
12 \LoadClass{article}
13 </cls>
```

Setup code for the package

```
14 <*package>
15 \RequirePackage{expl3, l3keys2e, ltxcmds}
16 \ProvidesExplPackage{stex}{2023/03/19}{3.3.0}{sTeX package}
17 \RequirePackage{stex-logo} % externalized for backwards-compatibility reasons
18 \RequirePackage{standalone}
19
20 \message{^^J*~This~is~sTeX~version~3.3.0~*^^J}
```

Package options:

```
21 \keys_define:nn { stex / package } {
22   debug      .str_set_x:N  = \c_stex_debug_clist ,
23   lang       .clist_set:N = \c_stex_languages_clist ,
24   mathhub    .tl_set_x:N  = \mathhub ,
25   usesms     .bool_set:N = \c_stex_persist_mode_bool ,
26   writesms   .bool_set:N = \c_stex_persist_write_mode_bool ,
27   checkterms .bool_set:N = \c_stex_check_terms_bool ,
28   image      .bool_set:N = \c_tikzinput_image_bool,
29   unknown    .code:n     = {}}
30 }
```

```

31 \exp_args:NNo \clist_set:Nn \c_stex_debug_list \c_stex_debug_list
32 \ProcessKeysOptions { stex / package }

    Error messages:

33 \input{stex-en.ldf}

```

## 13.2 Utilities

34 \cs\_set\_eq:NN \stex\_undefined: \undefined

### 13.2.1 Calling kpsewhich and Environment Variables

35 <@@=stex\_envs>

\stex\_kpsewhich:Nn

```

36 \cs_new_protected:Nn \stex_kpsewhich:Nn {\group_begin:
37   \catcode`\ =12
38   \sys_get_shell:nnN { kpsewhich ~ #2 } { } \l_tmpa_tl
39   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
40   \group_end:
41   \exp_args:NNo\str_set:Nn #1 {\l_tmpa_tl}
42   \tl_trim_spaces:N #1
43 }

```

(End definition for \stex\_kpsewhich:Nn. This function is documented on page 131.)

\stex\_get\_env:Nn

```

44 \sys_if_platform_windows:TF{
45   \cs_new_protected:Nn \stex_get_env:Nn {\group_begin:
46     \escapechar=-1\catcode`\\=12
47     \exp_args:NNe \stex_kpsewhich:Nn #1 {-expand-var~\c_percent_str#2\c_percent_str}
48     \exp_args:NNx\use:nn\group_end:{\str_set:Nn \exp_not:N #1 { #1 }
49   }
50 }
51 }
52 }{
53   \cs_new_protected:Nn \stex_get_env:Nn {
54     \stex_kpsewhich:Nn #1 {-var-value-#2}
55   }
56 }

```

(End definition for \stex\_get\_env:Nn. This function is documented on page 131.)

### 13.2.2 Logging

57 <@@=stex\_debug>

\stex\_debug:nn

```

58 \cs_new_protected:Nn \stex_debug:nn {
59   \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_list { \tl_to_str:n{all} }{
60     \__stex_debug_:nn{#1}{#2}
61   }{
62     \exp_args:NNo \clist_if_in:NnT \c_stex_debug_list { \tl_to_str:n{#1} }{
63       \__stex_debug_:nn{#1}{#2}
64     }

```

```

65     }
66 }
67
68 \cs_new_protected:Nn \__stex_debug_nn {
69     \msg_set:nnn{stex}{debug / #1}{
70         \\\Debug~#1:~#2\\
71     }
72     \msg_none:nn{stex}{debug / #1}
73 }

```

(End definition for `\stex_debug:nn`. This function is documented on page 109.)

```

\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:

```

74 \cs_new_protected:Nn \stex_fatal_error:n {
75     \msg_error:nn{stex}{#1}\input{Fatal-Error!!}
76 }
77 \cs_new_protected:Nn \stex_fatal_error:nnn {
78     \msg_error:nnn{stex}{#1}{#2}{#3}\input{Fatal-Error!!}
79 }
80 \cs_generate_variant:Nn \stex_fatal_error:nnn {nxx}

```

(End definition for `\stex_fatal_error:n` and `\stex_fatal_error:nnn`. These functions are documented on page 131.)

We check an environment variable for debugging and set things up:

```

81 \stex_get_env:Nn\__stex_debug_env_str{STEX_DEBUG}
82 \str_if_empty:NTF\__stex_debug_env_str {
83     \clist_set_eq:NN \l__stex_debug_cl \c_stex_debug_clist
84 }{
85     \clist_set:No \l__stex_debug_cl {\__stex_debug_env_str}
86 }
87 \clist_clear:N \c_stex_debug_clist
88 \clist_map_inline:Nn \l__stex_debug_cl {
89     \exp_args:NNo \clist_put_right:Nn \c_stex_debug_clist
90     { \tl_to_str:n{#1} }
91 }
92
93 \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist {\tl_to_str:n{all}} {
94     \msg_redirect_module:nnn{ stex }{ none }{ term }
95     \stex_debug:nn{all}{Logging~everything!}
96 }{
97     \clist_map_inline:Nn \c_stex_debug_clist {
98         \msg_redirect_name:nnn{ stex }{ debug / #1 }{ term }
99         \stex_debug:nn{#1}{Logging~#1}
100    }
101 }

```

### 13.2.3 Languages

102 `<@=stex_lang>`

```

\c_stex_languages_prop
\c_stex_language_abbrevs_prop

```

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

```

103 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
104     en = english ,
105     de = ngerman ,

```

```

106   ar = arabic ,
107   bg = bulgarian ,
108   ru = russian ,
109   fi = finnish ,
110   ro = romanian ,
111   tr = turkish ,
112   fr = french
113 }
114
115 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abrevs_prop { \tl_to_str:n {
116   english = en ,
117   ngerman = de ,
118   arabic = ar ,
119   bulgarian = bg ,
120   russian = ru ,
121   finnish = fi ,
122   romanian = ro ,
123   turkish = tr ,
124   french = fr
125 }
126 % todo: chinese simplified (zhs)
127 %         chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abrevs_prop`. These variables are documented on page 128.)

### `\l_stex_current_language_str`

```
128 \str_new:N \l_stex_current_language_str
```

(End definition for `\l_stex_current_language_str`. This variable is documented on page 128.)  
we use the lang-package option to load the corresponding babel languages:

```

\stex_set_language:n
\stex_set_language:x
\stex_set_language:o
129 \cs_new_protected:Nn \stex_set_language:n {
130   \str_set:Nn \l_stex_current_language_str { #1 }
131   \prop_if_in:NnTF \c_stex_languages_prop {#1} {
132     \tl_set_rescan:Nnx \l__stex_lang_lang_str {}{\prop_item:Nn \c_stex_languages_prop {#1}}
133     \cs_if_eq:NNTF\@onlypreamble\@notprerr{
134       \ltx@ifpackageloded{babel}{%
135         \exp_args:No\selectlanguage\l__stex_lang_lang_str
136       }{}}
137     }{
138       \ltx@ifpackageloded{babel}{}{%
139         \str_if_eq:nnTF {#1} {tr} {
140           \RequirePackage[turkish,shorthands=:!]{babel}
141         }{
142           \RequirePackage[\l__stex_lang_lang_str]{babel}
143         }
144       }
145     }{
146       \msg_error:nnx{stex}{error/unknownlanguage}{#1}
147     }
148   }
149 }
150 \cs_generate_variant:Nn \stex_set_language:n {x,o}

```

(End definition for `\stex_set_language:n`. This function is documented on page 129.)

`\stex_language_from_file:`

```
151 \cs_new_protected:Nn \stex_language_from_file: {
152   \seq_get_right:NN \g_stex_current_file \l_tmpa_str
153   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
154   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % = ".tex/.dtx/.sty"
155   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
156     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
157       \exp_args:No \str_if_eq:nnF \l_tmpa_str {ltx} {
158         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
159       }
160     }
161   }
162   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
163   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
164     \seq_pop_right:NN \l_tmpa_seq \l__stex_lang_str
165     \str_if_eq:NNF \l__stex_lang_str \l_stex_current_language_str {
166       \exp_args:NNo \prop_if_in:NnT \c_stex_languages_prop \l__stex_lang_str {
167         \stex_set_language:o \l__stex_lang_str
168       }
169     }
170     \stex_debug:nn{lang} {Language~\l_stex_current_language_str~
171       inferred~from~file~name}
172   }
173 }
```

(End definition for `\stex_language_from_file:`. This function is documented on page 129.)

Loading babel:

```
174 \clist_if_empty:NF \c_stex_languages_clist {
175   \bool_set_false:N \l__stex_lang_turkish_bool
176   \seq_clear:N \l_tmpa_seq
177   \clist_map_inline:Nn \c_stex_languages_clist {
178     \str_set:Nx \l_tmpa_str {\#1}
179     \str_if_eq:nnT {\#1}{tr} {
180       \bool_set_true:N \l__stex_lang_turkish_bool
181     }
182     \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
183       \tl_set_rescan:Nno \l_tmpa_str {} \l_tmpa_str
184       \seq_put_right:No \l_tmpa_seq \l_tmpa_str
185     }
186     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
187   }
188 }
189 \stex_debug:nn{lang} {Languages:~\seq_use:Nn \l_tmpa_seq {,~} }
190 \bool_if:NTF \l__stex_lang_turkish_bool {
191   \exp_args:NNe \use:nn \RequirePackage
192     {[main=\seq_use:Nn \l_tmpa_seq, ,shorthands=:!]}{babel}
193 }{
194   \exp_args:NNe \use:nn \RequirePackage
195     {[main=\seq_use:Nn \l_tmpa_seq, ]}{babel}
196 }
197 }
```

### 13.2.4 Group-like Behaviours

198 `<@=stex_groups>`

```
\stex_pseudogroup:nn
\stex_pseudogroup_restore:N
199 \cs_new_protected:Npn \stex_pseudogroup:nn {
200   \exp_args:Nne \use:n
201 }
202 \cs_new:Nn \stex_pseudogroup_restore:N {
203   \tl_if_exist:NTF #1 {
204     \tl_set:Nn \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
205   }{
206     \cs_undefine:N \exp_not:N #1
207   }
208 }
```

(End definition for `\stex_pseudogroup:nn` and `\stex_pseudogroup_restore:N`. These functions are documented on page 131.)

`\stex_pseudogroup_with:nn`

```
209 \cs_new_protected:Nn \stex_pseudogroup_with:nn {
210   \tl_map_inline:nn{#1} {
211     \cs_set_eq:cN{\_stex_groups_\tl_to_str:n{##1}}##1
212   }
213   #2
214   \tl_map_inline:nn{#1} {
215     \cs_set_eq:Nc{#1}{\_stex_groups_\tl_to_str:n{##1}}
216     \cs_undefine:c{\_stex_groups_\tl_to_str:n{##1}}
217   }
218 }
```

(End definition for `\stex_pseudogroup_with:nn`. This function is documented on page 131.)

`\stex_metagroup_new:n` List of all currently existing metagroup identifiers

`\stex_metagroup_new:o` `\seq_new:N \l__stex_groups_ids_seq`

start a new metagroup at the current group level with id #1

```
220 \cs_new_protected:Nn \stex_metagroup_new:n {
221   \str_set:cx{\l__stex_groups_#1_int} {\int_use:N\currentgrouplevel}
222   \seq_put_right:Nn \l__stex_groups_ids_seq {#1}
223 }
224 \cs_generate_variant:Nn \stex_metagroup_new:n {o}
```

(End definition for `\stex_metagroup_new:n`. This function is documented on page 132.)

`\stex_metagroup_do_in:nn`

`\stex_metagroup_do_in:nx`

```
225 \prg_new_conditional:Nnn \__stex_groups_exists:n {TF} {
226   \str_if_exist:cTF{\l__stex_groups_#1_int}
227     \prg_return_true: \prg_return_false:
228 }
229 \cs_new_protected:Nn \stex_metagroup_do_in:nn {
230   \__stex_groups_exists:nTF{#1} {
231     \__stex_groups_do_in:nn{#1}{#2}
232   }{
233     \msg_error:nnn{stex}{error/metagroup/missing}{#1}
```

```

235     }
236 }
237 \cs_generate_variant:Nn \stex_metagroup_do_in:nn {nx}
238
239 \cs_new_protected:Nn \__stex_groups_do_in:nn {
240   \exp_args:Nnx\stex_debug:nn{metagroup}{adding~to~\detokenize{\#1}:^~J\tl_to_str:n{\#2}}
241   \tl_set:Nn\__stex_groups_tmp{\#2}
242   \exp_args:Nx \int_compare:NnF {\use:c{l__stex_groups_#1_int}}
243     = \currentgrouplevel {
244       \tl_if_exist:cTF{g__stex_groups_#1_\the\currentgrouplevel _content} {
245         \exp_args:Nno \tl_gput_right:cn{g__stex_groups_#1_\the\currentgrouplevel _content}
246       }{
247         \exp_args:Nno \tl_gset:cn{g__stex_groups_#1_\the\currentgrouplevel _content}
248       }\__stex_groups_tmp
249       \bool_if_exist:cF {l__stex_groups_\the\currentgrouplevel _bool} {
250         \group_insert_after:N \__stex_groups_do:
251         \bool_set_true:c {l__stex_groups_\the\currentgrouplevel _bool}
252       }
253     }
254   \__stex_groups_tmp
255 }
256
257 \cs_new_protected:Nn \__stex_groups_do: {
258   \seq_map_inline:Nn \l__stex_groups_ids_seq {
259     \tl_if_exist:cT{g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content} {
260       \exp_args:NNno \exp_args:Nno \__stex_groups_do_in:nn{##1} {
261         \csname g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content\endcsname
262       }
263       \cs_undefine:c{g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content}
264     }
265     \bool_if_exist:cF {l__stex_groups_\int_eval:n\currentgrouplevel _bool} {
266       \group_insert_after:N \__stex_groups_do:
267       \bool_set_true:c {l__stex_groups_\int_eval:n\currentgrouplevel _bool}
268     }
269   }
270 }

```

(End definition for `\stex_metagroup_do_in:nn`. This function is documented on page 132.)

### 13.2.5 HTML Annotations

```
271 <@=stex_annotation>
```

`\stex_if_do_html:TF` Whether to (locally) produce HTML output

```

272 \bool_new:N \_stex_html_do_output_bool
273 \bool_set_true:N \_stex_html_do_output_bool
274
275 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
276   \bool_if:nTF \_stex_html_do_output_bool
277     \prg_return_true: \prg_return_false:
278 }
```

(End definition for `\stex_if_do_html:TF`. This function is documented on page 129.)

```

\stex_suppress_html:n Temporarily disable HTML output
279 \cs_new_protected:Nn \stex_suppress_html:n {
280   \stex_pseudogroup:nn{
281     \bool_set_false:N \stex_html_do_output_bool
282     #1
283   }{
284     \stex_if_do_html:T {
285       \bool_set_true:N \stex_html_do_output_bool
286     }
287   }
288 }

```

(End definition for `\stex_suppress_html:n`. This function is documented on page 129.)

We determine the backend:

```

\stex_if_html_backend_p:
\stex_if_html_backend:TF
  \ifstexhtml
    \stex@backend
  \fi
  \stex_get_env:Nn\__stex_annotate_env_str{STEX_FORCE_PDF}
  \exp_args:No \str_if_eq:nnTF \__stex_annotate_env_str {true} {
    \def\stex@backend{pdflatex}
  }{
    \tl_if_exist:NF\stex@backend{
      \if@rustex
        \def\stex@backend{rustex}
      \else
        \cs_if_exist:NTF\HCode{
          \def\stex@backend{tex4ht}
        }{
          \def\stex@backend{pdflatex}
        }
      \fi
    }
  }
  \input{stex-backend-\stex@backend.cfg}
  \newif\ifstexhtml
  \stex_if_html_backend:TF {
    \stexhtmltrue
    \bool_set_true:N \stex_html_do_output_bool
  }{
    \stexhtmlfalse
    \bool_set_false:N \stex_html_do_output_bool
  }

```

(End definition for `\stex_if_html_backend:TF`, `\ifstexhtml`, and `\stex@backend`. These functions are documented on page 129.)

```

\_stex_annotate_force_break:n
321 \stex_if_html_backend:TF {
322   \cs_new_protected:Nn \_stex_annotate_force_break:n {

```

```

323     \stex_annotation_invisible:n{~}
324     #1
325     \stex_annotation_invisible:n{~}
326   }
327 }{
328   \cs_new_protected:Nn \_stex_annotation_force_break:n { #1 }
329 }
```

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

```

\mmlintent
\mmlarg 330 \stex_if_html_backend:TF {
331   \cs_new_protected:Npn \mmlintent #1 #2 {
332     \stex_annotation:nn{mml:intent={#1}}{#2}
333   }
334   \cs_new_protected:Npn \mmlarg #1 #2 {
335     \stex_annotation:nn{mml:arg={#1}}{#2}
336   }
337 }{
338   \cs_new_protected:Npn \mmlintent #1 #2 { #2 }
339   \cs_new_protected:Npn \mmlarg #1 #2 { #2 }
340 }
```

(End definition for `\mmlintent` and `\mmlarg`. These functions are documented on page ??.)

### 13.2.6 Auxiliary Methods

```
341 <@=stex_aux>
```

```

\stex_deactivate_macro:Nn
\stex_reactivate_macro:N
342 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
343   \tl_set_eq:cN{\tl_to_str:n{#1}~~~orig}{#1}
344   \tl_set:Nn#1{
345     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
346   }
347 }
348 \cs_new_protected:Nn \stex_reactivate_macro:N {
349   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1}~~~orig\endcsname
350 }
```

(End definition for `\stex_deactivate_macro:Nn` and `\stex_reactivate_macro:N`. These functions are documented on page 131.)

`\stex_ignore_spaces_and_pars:`

```

351 \protected\def\stex_ignore_spaces_and_pars: {
352   \begingroup\catcode13=10\relax
353   \@ifnextchar\par{
354     \endgroup\expandafter\stex_ignore_spaces_and_pars:\@gobble
355   }{
356     \endgroup
357   }
358 }
```

(End definition for `\stex_ignore_spaces_and_pars:`. This function is documented on page 131.)

```

\stex_keys_define:nnnn
 359 \cs_new_nopar:Nn \stex_keys_define:nnnn {
 360   \tl_gset:cn {__stex_aux_keys_#1_pre_tl}{#2}
 361   \tl_gset:cn {__stex_aux_keys_#1_def_tl}{#3}
 362   \tl_if_empty:nF{#4} {
 363     \clist_map_inline:nn{#4} {
 364       \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_pre_tl}
 365       \tl_gput_left:co{__stex_aux_keys_#1_pre_tl} \l__stex_aux_tl
 366       \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_def_tl}
 367       \tl_gput_left:cn{__stex_aux_keys_#1_def_tl} ,
 368       \tl_gput_left:co{__stex_aux_keys_#1_def_tl} \l__stex_aux_tl
 369     }
 370   }
 371   \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_#1_def_tl}
 372   \exp_args:Nno \keys_define:nn {stex / #1} {\l__stex_aux_tl}
 373 }

```

(End definition for `\stex_keys_define:nnnn`. This function is documented on page 121.)

`\stex_keys_set:nn`

```

 374 \cs_new_nopar:Nn \stex_keys_set:nn {
 375   \use:c{__stex_aux_keys_#1_pre_tl}
 376   \keys_set:nn {stex / #1} { #2 }
 377 }

```

(End definition for `\stex_keys_set:nn`. This function is documented on page 121.)

Some ubiquitous key sets:

```

 378 \stex_keys_define:nnnn{archive file} {
 379   \str_clear:N \l_stex_key_archive_str
 380   \str_clear:N \l_stex_key_file_str
 381 } {
 382   archive .str_set_x:N = \l_stex_key_archive_str ,
 383   file .str_set_x:N = \l_stex_key_file_str
 384 } {}
 385
 386 \stex_keys_define:nnnn{id} {
 387   \str_clear:N \l_stex_key_id_str
 388 } {
 389   id .str_set_x:N = \l_stex_key_id_str
 390 } {}
 391
 392 \stex_keys_define:nnnn{title} {
 393   \tl_clear:N \l_stex_key_title_tl
 394 } {
 395   title .tl_set:N = \l_stex_key_title_tl
 396 } {}
 397
 398 \stex_keys_define:nnnn{style} {
 399   \clist_clear:N \l_stex_key_style_clist
 400 } {
 401   style .clist_set:N = \l_stex_key_style_clist
 402 } {}
 403
 404 \stex_keys_define:nnnn{deprecate} {

```

```

405   \str_clear:N \l_stex_key_deprecate_str
406 }{
407   deprecate     .str_set_x:N = \l_stex_key_deprecate_str
408 }{}
409
410 \stex_keys_define:nnnn{uses}{}
411   uses .code:n = {
412     \clist_map_inline:nn{#1}{%
413       \stex_if_starts_with:nTF{##1}[%
414         \__stex_aux_split_at_bracket:w ##1 \_stex_end:
415       }{%
416         \usemodule{##1}
417       }
418     }
419   }
420 }{}
421
422 \cs_new_protected:Npn \__stex_aux_split_at_bracket:w [ #1 ] #2 \_stex_end: {%
423   \usemodule[#1]{#2}
424 }

```

### \\_stex\_do\_deprecation:n

```

425 \cs_new:Nn \_stex_do_deprecation:n {
426   \str_if_empty:NF \l_stex_key_deprecate_str {
427     \msg_warning:nnxx{stex}{warning/deprecated}{#1}{\l_stex_key_deprecate_str}
428   }
429 }

```

(End definition for `\_stex_do_deprecation:n`. This function is documented on page 109.)

### \\_stex\_do\_id:

```

430 \cs_new_protected:Nn \_stex_do_id: {
431   \stex_if_smsmode:F {
432     \str_if_empty:NF \l_stex_key_id_str {
433       \exp_args:No \stex_ref_new_doc_target:n \l_stex_key_id_str
434     }
435   }
436 }

```

(End definition for `\_stex_do_id:`. This function is documented on page 121.)

### \stex\_new\_styable\_env:nnnnnn

### \stex\_new\_styable\_cmd:nnnn

### \stex\_style\_apply:

```

437 \cs_new_protected:Nn \stex_new_styable_cmd:nnnn {
438   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[2][]{%
439     \__stex_aux_patch:nnn{#1}{##1}{##2}
440   }
441   \exp_after:wN \NewDocumentCommand\cs:w #1\cs_end:{#2}{%
442     \cs_set:Npn \stex_style_apply: {
443       \__stex_aux_apply_patch:n{#1}
444     }
445     #3
446   }
447   \tl_set:cn {\__stex_aux_style_#1:} { #4 }
448 }

```

```

449 \cs_new_protected:Nn \__stex_aux_apply_patch:n {
450   \clist_if_empty:NTF \l_stex_key_style_clist {
451     \tl_clear:N \thisstyle
452     \use:c{\__stex_aux_style_#1:}
453   }{
454     \clist_get:NN \l_stex_key_style_clist \thisstyle
455     \tl_if_exist:cTF{\__stex_aux_style_#1_\thisstyle :} {
456       \use:c{\__stex_aux_style_#1_\thisstyle :}
457     }{
458       \use:c{\__stex_aux_style_#1:}
459     }
460   }
461 }
462 }
463
464 \cs_new_protected:Nn \__stex_aux_patch:nnn {
465   \str_if_empty:nTF {#2}{%
466     \tl_set:cn{\__stex_aux_style_#1:}{#3}
467   }{
468     \tl_set:cn{\__stex_aux_style_#1_#2:}{#3}
469   }
470 }
471
472 \cs_new_protected:Nn \stex_new_stylable_env:nnnnnnn {
473   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[3][]{
474     \__stex_aux_patch:nnnn{#1}{##1}{##2}{##3}
475   }
476   \NewDocumentEnvironment{#7#1}{#2}{%
477     \cs_set:Npn \stex_style_apply: {
478       \__stex_aux_apply_patch_begin:n{#1}
479     }
480     #3
481   }{
482     \cs_set:Npn \stex_style_apply: {
483       \__stex_aux_apply_patch_end:n{#1}
484     }
485     #4
486   }
487   \tl_set:cn {\__stex_aux_style_#1_start:} { #5 }
488   \tl_set:cn {\__stex_aux_style_#1_end:} { #6 }
489 }
490
491 \cs_new_protected:Nn \__stex_aux_patch:nnnn {
492   \str_if_empty:nTF {#2}{%
493     \tl_set:cn{\__stex_aux_style_#1_start:}{#3}
494     \tl_set:cn{\__stex_aux_style_#1_end:}{#4}
495   }{
496     \tl_set:cn{\__stex_aux_style_#1_#2_start:}{#3}
497     \tl_set:cn{\__stex_aux_style_#1_#2_end:}{#4}
498   }
499 }
500
501 \cs_new_protected:Nn \__stex_aux_apply_patch_begin:n {
502   \clist_if_empty:NTF \l_stex_key_style_clist {

```

```

503   \tl_clear:N \thisstyle
504   \use:c{__stex_aux_style_#1_start:}
505 }{
506   \clist_get:NN \l_stex_key_style_clist \thisstyle
507   \stex_debug:nnf{styling}{dominant-style:~\thisstyle}
508   \tl_if_exist:cTF{__stex_aux_style_#1_\thisstyle_start:}{
509     \use:c{__stex_aux_style_#1_\thisstyle_start:}
510   }{
511     \use:c{__stex_aux_style_#1_start:}
512   }
513 }
514 }
515
516 \cs_new_protected:Nn \__stex_aux_apply_patch_end:n {
517   \tl_if_empty:NTF \thisstyle {
518     \use:c{__stex_aux_style_#1_end:}
519   }{
520     \tl_if_exist:cTF{__stex_aux_style_#1_\thisstyle_end:}{
521       \use:c{__stex_aux_style_#1_\thisstyle_end:}
522     }{
523       \use:c{__stex_aux_style_#1_end:}
524     }
525   }
526 }

```

(End definition for `\stex_new_stylable_env:nnnnnnn`, `\stex_new_stylable_cmd:nnnn`, and `\stex_style_apply:`. These functions are documented on page 122.)

`\stex_str_if_ends_with_p:nn`

`\stex_str_if_ends_with:nnTF`

```

527 \prg_new_conditional:Nnn \stex_str_if_ends_with:nn {p,T,F,TF} {
528   \exp_args:Ne \str_if_eq:nnTF {
529     \str_range:nnn{#1}{- \str_count:n{#2}}{-1}
530   }{#2}\prg_return_true: \prg_return_false:
531 }

```

(End definition for `\stex_str_if_ends_with:nnTF`. This function is documented on page 125.)

`\stex_str_if_starts_with_p:nn`

`\stex_str_if_starts_with:nnTF`

```

532 \prg_new_conditional:Nnn \stex_str_if_starts_with:nn {p,T,F,TF} {
533   \exp_args:Ne \str_if_eq:nnTF {
534     \str_range:nnn{#1}{1}{\str_count:n{#2}}
535   }{#2}\prg_return_true: \prg_return_false:
536 }

```

(End definition for `\stex_str_if_starts_with:nnTF`. This function is documented on page 125.)

`\stex_macro_body:N`

```

537 \cs_new:Npn \__stex_aux_start:#1\__stex_aux_end: {\exp_not:n{#1}}
538 \cs_new_protected:Nn \__stex_aux_end: {}
539 \cs_new:Nn \stex_macro_body:N {
540   \exp_args:Nne\use:nn{\exp_after:wN \__stex_aux_start: #1} {
541     \__stex_aux_args:e {\cs_parameter_spec:N #1}\__stex_aux_end:
542   }
543 }
544

```

```

545 \cs_new:Nn \__stex_aux_args:n {
546   \tl_if_empty:nF{#1} {
547     {##\exp_args:Ne \tl_head:n {\tl_tail:n {#1}}}
548     \__stex_aux_args:e {\exp_args:Ne\tl_tail:n{\tl_tail:n{#1}}}
549   }
550 }
551 \cs_generate_variant:Nn \__stex_aux_args:n {e}

```

(End definition for `\stex_macro_body:N`. This function is documented on page 130.)

### `\stex_macro_definition:N`

```

552 \cs_new:Nn \stex_macro_definition:N {
553   \__stex_aux_prefix:e {\cs_prefix_spec:N #1}
554   \def\exp_not:N #1
555   \__stex_aux_params:e {\cs_parameter_spec:N #1}
556   {
557     \stex_macro_body:N #1
558   }
559 }
560
561 \cs_new:Nn \__stex_aux_prefix:n {
562   \tl_if_empty:nF{#1} {
563     \str_if_eq:eeTF {
564       \tl_range:nnn{#1}{1}{10}~
565     }{\tl_to_str:n{\protected}}{
566       \protected
567       \__stex_aux_prefix_long:e {
568         \str_range:nnn{#1}{11}{-1}
569       }
570     }{
571       \__stex_aux_prefix_long:n {#1}
572     }
573   }
574 }
575 \cs_generate_variant:Nn \__stex_aux_prefix:n {e}
576
577 \cs_new:Nn \__stex_aux_prefix_long:n {
578   \tl_if_empty:nF{#1} {
579     \str_if_eq:eeT {
580       \tl_range:nnn{#1}{1}{10}~
581     }{\tl_to_str:n{\long}}{\long}
582   }
583 }
584 \cs_generate_variant:Nn \__stex_aux_prefix_long:n {e}
585
586 \cs_new:Nn \__stex_aux_params:n {
587   \tl_if_empty:nF{#1} {
588     \exp_args:NNN \str_if_eq:VnTF \c_hash_str {\tl_head:n{#1}}{
589       #####
590     }{
591       \tl_head:n{#1}
592     }
593     \__stex_aux_params:e {\tl_tail:n{#1}}
594   }

```

```

595 }
596 \cs_generate_variant:Nn \__stex_aux_params:n {e}

```

(End definition for `\stex_macro_definition:N`. This function is documented on page 130.)

### 13.2.7 Persistence

```

597 <@=stex_persist>

```

We check the environment variables:

```

598 \stex_get_env:Nn\__stex_persist_env_str{STEX_USESMS}
599 \str_if_empty:NF\__stex_persist_env_str{
600     \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false} {
601         \bool_set_true:N \c_stex_persist_mode_bool
602     }
603 }
604 \stex_get_env:Nn\__stex_persist_env_str{STEX_WRITESMS}
605 \str_if_empty:NF\__stex_persist_env_str{
606     \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false} {
607         \bool_set_true:N \c_stex_persist_write_mode_bool
608     }
609 }

```

```

\stex_persist:n
\stex_persist:e
610 \iow_new:N \c__stex_persist_sms_iow
611
612 \bool_if:NTF \c_stex_persist_write_mode_bool {
613     \stex_if_html_backend:TF{
614         \cs_new:Npn \stex_persist:n #1 {}
615         \cs_new:Npn \stex_persist:e #1 {}
616     }{
617         \cs_new_protected:Nn \stex_persist:n {
618             \iow_now:Nn \c__stex_persist_sms_iow {#1}
619         }
620         \cs_generate_variant:Nn \stex_persist:n {e}
621     }
622 }{
623     \cs_new:Npn \stex_persist:n #1 {}
624     \cs_new:Npn \stex_persist:e #1 {}
625 }

```

(End definition for `\stex_persist:n`. This function is documented on page 130.)

Is called at the end of the .sty-file:

```

626 \cs_new_protected:Nn \__stex_persist_load_file:n {
627     \file_if_exist:nT{#1} {
628         \group_begin:
629         \cs_set:Npn \stex_persist:n ##1 {}
630         \cs_set:Npn \stex_persist:e ##1 {}
631         \stex_debug:nn{persist}{restoring~from~sms~file}
632         \catcode`\ =10\relax
633         \cs:w @ @ input \cs_end:#1\relax
634         \group_end:
635     }
636 }
637 }

```

```

638 \cs_new_protected:Nn \__stex_persist_write_only: {
639   \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
640   \AtEndDocument{ \iow_close:N \c__stex_persist_sms_iow }
641 }
642 }
643
644 \cs_new_protected:Nn \__stex_persist_read_and_write: {
645   \file_if_exist:nTF{\jobname sms}{\ior_open:Nn \g_tmpa_ior {\jobname sms}}
646   \ior_open:Nn \g_tmpa_iow {\jobname sms2}
647   \ior_str_map_inline:Nn \g_tmpa_ior {
648     \iow_now:Nn \g_tmpa_iow {##1}
649   }
650   \iow_close:N \g_tmpa_iow
651   \ior_close:N \g_tmpa_ior
652   \__stex_persist_write_only:
653   \ior_open:Nn \g_tmpa_ior {\jobname sms2}
654   \ior_str_map_inline:Nn \g_tmpa_ior {
655     \iow_now:Nn \c__stex_persist_sms_iow {##1}
656   }
657   \ior_close:N \g_tmpa_ior
658   \__stex_persist_load_file:n{\jobname sms2}
659 }\__stex_persist_write_only:
660 }
661 }
662
663 \cs_new_protected:Nn \__stex_persist_read_now: {
664   \bool_if:NTF \c_stex_persist_mode_bool {
665     \bool_if:NTF \c_stex_persist_write_mode_bool
666       \__stex_persist_read_and_write:
667     {
668       \__stex_persist_load_file:n{\jobname sms}
669     }
670   }{
671     \bool_if:NT \c_stex_persist_write_mode_bool \__stex_persist_write_only:
672   }
673 }

```

### 13.2.8 Files, Paths and URIs

```

674 <@=stex_path>

\stex_file_set:Nn
\stex_file_set:No
\stex_file_set:Nx
675 \cs_new_protected:Nn \stex_file_set:Nn {
676   \str_if_empty:nTF {#2} { \seq_clear:N #1 }{
677     \exp_args:NNno \seq_set_split:Nnn #1 / { \tl_to_str:n{#2} }
678   }
679 }
680 \cs_generate_variant:Nn \stex_file_set:Nn {No, Nx}

```

(End definition for \stex\_file\_set:Nn. This function is documented on page 126.)

```

\stex_file_resolve:Nn
\stex_file_resolve:No
\stex_file_resolve:Nx
681 \sys_if_platform_windows:TF{
682   \cs_new_protected:Npn \__stex_path_win_take:w #1#2#3 \__stex_path_:

```

```

683   \str_set:Nn \l__stex_path_win_drive {#1#2}
684   \str_set:Nn \l__stex_path_str{#3}
685 }
686 \cs_new_protected:Nn \stex_file_resolve:Nn {
687   \str_set:Nn \l__stex_path_str {#2}
688   \str_clear:N \l__stex_path_win_drive
689   \exp_args:NNo \str_replace_all:Nnn \l__stex_path_str \c_backsplash_str /
690   \exp_args:Nx \str_if_eq:nnT {\str_item:Nn \l__stex_path_str 2} : {
691     \exp_after:wN \__stex_path_win_take:w \l__stex_path_str \__stex_path_:
692   }
693   \stex_file_set:No #1 \l__stex_path_str
694   \__stex_path_canonicalize:N #1
695   \str_if_empty:NF \l__stex_path_win_drive {
696     \seq_pop_left:NN #1 \l__stex_path_str
697     \seq_put_left:No #1 \l__stex_path_win_drive
698   }
699   \%stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
700 }
701 }{
702   \cs_new_protected:Nn \stex_file_resolve:Nn {
703     \str_set:Nn \l__stex_path_str {#2}
704     \stex_file_set:No #1 \l__stex_path_str
705     \__stex_path_canonicalize:N #1
706     \%stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
707   }
708 }
709 \cs_generate_variant:Nn \stex_file_resolve:Nn {No, Nx}
710
711 \cs_new_protected:Nn \__stex_path_canonicalize:N {
712   \seq_if_empty:NF #1 {
713     \seq_pop>NN #1 \l__stex_path_str
714     \seq_clear:N \l__stex_path_seq
715     \str_if_empty:NTF \l__stex_path_str {
716       \seq_map_function:NN #1 \__stex_path_dodots:n
717       \seq_put_left:Nn \l__stex_path_seq {}
718     }{
719       \seq_push:No #1 \l__stex_path_str
720       \seq_map_function:NN #1 \__stex_path_dodots:n
721     }
722     \seq_set_eq:NN #1 \l__stex_path_seq
723   }
724 }
725
726 \cs_new_protected:Nn \__stex_path_dodots:n {
727   \str_if_empty:nF{#1} {
728     \str_if_eq:nnF {#1} {.} {
729       \str_if_eq:nnTF {#1} {..} {
730         \seq_if_empty:NF \l__stex_path_seq {
731           \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
732         }
733       }{
734         \seq_put_right:Nn \l__stex_path_seq {#1}
735       }
736     }

```

```
737     }
738 }
```

(End definition for `\stex_file_resolve:Nn`. This function is documented on page 126.)

```
\stex_if_file_absolute_p:N
\stex_if_file_absolute:NTF
739
740 \sys_if_platform_windows:TF {
741   \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
742     \seq_if_empty:NTF #1 \prg_return_false: {
743       \tl_set:Nx \l__stex_path_maybein_str {\seq_item:Nn #1 1}
744       \exp_args:No \tl_if_empty:nTF \l__stex_path_maybein_str \prg_return_true: {
745         \exp_args:Nx \str_if_eq:nnTF {\str_item:Nn \l__stex_path_maybein_str 2} :
746           \prg_return_true: \prg_return_false:
747         }
748       }
749     }
750   }{
751     \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
752       \seq_if_empty:NTF #1 \prg_return_false: {
753         \exp_args:Nx \tl_if_empty:nTF {\seq_item:Nn #1 1}
754           \prg_return_true: \prg_return_false:
755         }
756       }
757     }
758 }
```

(End definition for `\stex_if_file_absolute:NTF`. This function is documented on page 126.)

```
\stex_file_use:N
758 \cs_new:Nn \stex_file_use:N {
759   \seq_use:Nn #1 /
760 }
```

(End definition for `\stex_file_use:N`. This function is documented on page 126.)

```
stex_if_file_starts_with>NNTF
761 \prg_new_protected_conditional:Nnn \stex_if_file_starts_with>NN {T,F,TF} {
762   \seq_set_eq:NN \l__stex_path_a_seq #1
763   \seq_set_eq:NN \l__stex_path_b_seq #2
764   \tl_clear:N \l__stex_path_return_tl
765   \bool_while_do:nn{
766     \bool_not_p:n{
767       \bool_lazy_any_p:n{
768         {\seq_if_empty_p:N \l__stex_path_a_seq}
769         {\seq_if_empty_p:N \l__stex_path_b_seq}
770         {\bool_not_p:n{\tl_if_empty_p:N \l__stex_path_return_tl}}
771       }
772     }
773   }{
774     \seq_pop_left:NN \l__stex_path_a_seq \l__stex_path_a_tl
775     \seq_pop_left:NN \l__stex_path_b_seq \l__stex_path_b_tl
776     \str_if_eq:NNF \l__stex_path_a_tl \l__stex_path_b_tl {
777       \tl_set:Nn \l__stex_path_return_tl {\prg_return_false:}
778     }
779   }
```

```

780   \tl_if_empty:NTF \l__stex_path_return_tl {
781     \seq_if_empty:NTF \l__stex_path_b_seq \prg_return_true: \prg_return_false:
782   } \l__stex_path_return_tl
783 }

```

(End definition for `\stex_if_file_starts_with:NNTF`. This function is documented on page 126.)

`\stex_file_split_off_ext:NN`

```

\stex_file_split_off_lang:NN
784 \cs_new_protected:Nn \stex_file_split_off_ext:NN {
785   \seq_set_eq:NN #1 #2
786   \seq_pop_right:NN #1 \l__stex_path_str
787   \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
788   \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
789   \seq_put_right:Nx #1 {\seq_use:Nn \l__stex_path_seq .}
790 }
791 \cs_new_protected:Nn \stex_file_split_off_lang: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
797   \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
798   \exp_args:NNo \prop_if_in:NnF \c_stex_languages_prop \l__stex_path_str {
799     \seq_put_right:No \l__stex_path_seq \l__stex_path_str
800   }
801
802   \seq_put_right:Nx #1 {\seq_use:Nn \l__stex_path_seq .}
803 }

```

(End definition for `\stex_file_split_off_ext:NN` and `\stex_file_split_off_lang:NN`. These functions are documented on page 126.)

URIs:

`\stex_map_uri:Nnnnn`

```

804 \cs_set_protected:Nn \__stex_path_auth:n {
805   \msg_error:nnx{stex}{error/misused-uri}{\tl_to_str:n{#1}}
806 }
807 \cs_set_eq:NN \__stex_path_path:n \__stex_path_auth:n
808 \cs_set_eq:NN \__stex_path_module:n \__stex_path_auth:n
809 \cs_set_eq:NN \__stex_path_name:n \__stex_path_auth:n
810
811 \cs_set_protected:Nn \stex_map_uri:Nnnnn{
812   \stex_pseudogroup_with:nn{\__stex_path_auth:n\__stex_path_path:n\__stex_path_module:n\__stex_path_name:n}
813   \cs_set:Npn \__stex_path_auth:n ##1 {#2}
814   \cs_set:Npn \__stex_path_path:n ##1 {#3}
815   \cs_set:Npn \__stex_path_module:n ##1 {#4}
816   \cs_set:Npn \__stex_path_name:n ##1 {#5}
817   #1
818 }
819 }

```

(End definition for `\stex_map_uri:Nnnnn`. This function is documented on page 127.)

```

\stex_uri_set:Nn
\stex_uri_set:Nn
\stex_uri_set:Nx
820 \str_set:Nx\__stex_path_colonslash{\cColonStr}
821 \cs_new_protected:Nn \__stex_path_uri_set:NnN {
822   \str_if_empty:nTF {#2} {
823     \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{empty}
824   }{
825     \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn #1 \__stex_path_colonslash { \tl_to_str
826     \seq_pop_left:NN #1 \l__stex_path_auth_str
827     \seq_if_empty:NTF #1 {
828       \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{missing-authority}
829     }{
830       \exp_args:NNnx \seq_set_split:Nnn #1 ? {\exp_args:NNo \seq_use:Nn #1 \__stex_path_colon
831       \seq_pop_left:NN #1 \l__stex_path_path
832       #3 \l__stex_path_path \l__stex_path_path
833       \seq_if_empty:NTF #1 {
834         \exp_args:NNo \__stex_path_uri_set:Nnxnn #1 \l__stex_path_auth_str
835         {\stex_file_use:N \l__stex_path_path} {} {}
836       }{
837         \seq_pop_left:NN #1 \l__stex_path_mod
838         \seq_if_empty:NTF #1 {
839           \exp_args:NNo \__stex_path_uri_set:Nnxon #1 \l__stex_path_auth_str
840           {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod {}
841         }{
842           \seq_pop_left:NN #1 \l__stex_path_name
843           \seq_if_empty:NTF #1 {
844             \exp_args:NNo \__stex_path_uri_set:Nnxon #2 \l__stex_path_auth_str
845             {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod \l__stex_path_name
846           }{
847             \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{too-many~?s}
848           }
849         }
850       }
851     }
852   }
853   \stex_debug:nn{uris}{Set~\tl_to_str:n{#1}~to~\stex_uri_use:N #1}
854 }
855
856 \cs_new_protected:Nn \__stex_path_uri_set:Nnnnn{
857   \tl_set:Nn #1 {
858     \__stex_path_auth:n{ #2 }
859     \__stex_path_path:n{ #3 }
860     \__stex_path_module:n{ #4 }
861     \__stex_path_name:n{ #5 }
862   }
863 }
864 \cs_generate_variant:Nn\__stex_path_uri_set:Nnnnn {Nnxnn,Nnxon,Nnxoo}
865
866 \cs_new_protected:Nn \stex_uri_set:Nn {
867   \__stex_path_uri_set:NnN #1 {#2} \stex_file_set:Nn
868 }
869 \cs_generate_variant:Nn \stex_uri_set:Nn {No, Nx}

```

(End definition for `\stex_uri_set:Nn`. This function is documented on page 127.)

```

\stex_uri_resolve:Nn
\stex_uri_resolve:Nn
\stex_uri_resolve:Nx
870 \cs_new_protected:Nn \stex_uri_resolve:Nn {
871   \__stex_path_uri_set:NnN #1 {#2} \stex_file_resolve:No
872 }
873 \cs_generate_variant:Nn \stex_uri_resolve:Nn {No, Nx}

(End definition for \stex_uri_resolve:Nn. This function is documented on page 127.)

```

```

\stex_uri_use:N
874 \cs_new:Npn \__stex_path_uri_use:w \__stex_path_auth:n #1 \__stex_path_path:n #2 \__stex_path_
875   #1\c_colon_str/ #2 \tl_if_empty:nF { #3 }{ ? #3
876     \tl_if_empty:nF { #4 }{ ? #4 } }
877 }
878 \cs_new:Nn \stex_uri_use:N {
879   \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N #1 } \__stex_path_auth:n {
880     \exp_after:wn \__stex_path_uri_use:w #1
881   }{
882     \msg_error:nnnn{stex}{error/invalid-uri}{#1}{Not~a~URI}
883   }
884 }

(End definition for \stex_uri_use:N. This function is documented on page 127.)

```

```

\stex_uri_from_repo_file>NNNn
\stex_uri_from_repo_file_nolang:NNNn
885 \cs_new_protected:Npn \stex_uri_from_repo_file_nolang:NNNn {
886   \__stex_path_from_repo_file:NNNNn \stex_file_split_off_lang:NN
887 }
888 \cs_new_protected:Npn \stex_uri_from_repo_file:NNNn {
889   \__stex_path_from_repo_file:NNNNn \stex_file_split_off_ext:NN
890 }
891
892 \cs_new_protected:Nn \__stex_path_from_repo_file:NNNNn {
893   #1 \l__stex_path_file #4
894   \prop_if_exist:NTF #3 {
895     \str_clear:N \l__stex_path_uri
896     \prop_get:NnNF #3 {#5} \l__stex_path_uri {
897       \prop_get:NnNF #3 {ns} \l__stex_path_uri {
898         \__stex_path_uri_set:Nnxnn #2 {file}
899         {\stex_file_use:N \l__stex_path_file} {} {}
900       }
901     }
902     \str_if_empty:NF \l__stex_path_uri {\__stex_path_relativize:N #2}
903   }{
904     \exp_args:NNx \__stex_path_uri_set:Nnxnn #2 {\tl_to_str:n{file}}
905     {\stex_file_use:N \l__stex_path_file} {} {}
906   }
907 }
908
909 \cs_new_protected:Nn \__stex_path_relativize:N {
910   \seq_set_eq:NN \l__stex_path_seq \l__stex_path_file
911   \seq_map_inline:Nn \c_stex_mathhub_file { % mathhub path
912     \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
913   }
914   \stex_file_set:Nx \l__stex_path_path {\prop_item:Nn \l_stex_current_archive_prop {id} }
```

```

915   \seq_map_inline:Nn \l__stex_path_path { % id
916     \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
917   }
918   \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl % source
919
920   \stex_uri_set:Nx #1 { \l__stex_path_uri / \stex_file_use:N \l__stex_path_seq }
921 }

```

(End definition for `\stex_uri_from_repo_file:Nnn` and `\stex_uri_from_repo_file_nolang:Nnn`. These functions are documented on page 127.)

```

\stex_uri_from_current_file:Nn
\stex_uri_from_current_file_nolang:Nn
922 \cs_new_protected:Nn \stex_uri_from_current_file:Nn {
923   \stex_debug:nn{docuri}{Here:~\stex_file_use:N \g_stex_current_file}
924   \stex_uri_from_repo_file:Nnn #1 \l_stex_current_archive_prop
925     \g_stex_current_file {#2}
926   \stex_debug:nn{docuri}{resolved:~\stex_uri_use:N #1}
927 }
928 \cs_new_protected:Nn \stex_uri_from_current_file_nolang:Nn {
929   \stex_uri_from_repo_file_nolang:Nnn #1 \l_stex_current_archive_prop
930     \g_stex_current_file {#2}
931 }

```

(End definition for `\stex_uri_from_current_file:Nn` and `\stex_uri_from_current_file_nolang:Nn`. These functions are documented on page 128.)

```

\stex_uri_add_module:NNn
\stex_uri_add_module:NNo
932 \cs_new_protected:Nn \stex_uri_add_module:NNn {
933   \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N #2 } \__stex_path_auth:n {
934     \stex_pseudogroup_with:nn
935       {\__stex_path_auth:n\__stex_path_path:n\__stex_path_module:n\__stex_path_name:n}
936     {
937       \cs_set:Npn \__stex_path_module:n ##1 {
938         \tl_if_empty:nTF{##1} {
939           \exp_not:N \__stex_path_module:n {#3}
940         }{
941           \msg_error:nnn{stex}{error/invalid-dpath}{#2}
942         }
943       }
944       \cs_set:Npn \__stex_path_name:n ##1 {
945         \tl_if_empty:nTF{##1} {
946           \exp_not:N \__stex_path_name:n {}
947         }{
948           \msg_error:nnn{stex}{error/invalid-dpath}{#2}
949         }
950       }
951       \tl_set:Nx #1 {#2}
952     }
953   }{
954     \msg_error:nnnn{stex}{error/invalid-uri}{#2}{Not~a~URI}
955   }
956 }
957 \cs_generate_variant:Nn \stex_uri_add_module:NNn {NNo}

```

(End definition for `\stex_uri_add_module:NNn`. This function is documented on page 128.)

```
\l_stex_current_doc_uri
```

```
958 \tl_new:N \l_stex_current_doc_uri
```

(End definition for `\l_stex_current_doc_uri`. This variable is documented on page 128.)

```
\stex_set_document_uri:
```

```
959 \cs_new_protected:Nn \stex_set_document_uri: {  
960   \stex_uri_from_current_file:Nn \l_stex_current_doc_uri {narr}  
961   \%stex_debug:nn{sref}{Document~URI:~\stex_uri_use:N \l_stex_current_doc_uri}  
962 }
```

(End definition for `\stex_set_document_uri`. This function is documented on page 128.)

```
\stex_set_current_namespace:
```

```
963 \cs_new_protected:Nn \stex_set_current_namespace: {  
964   \stex_uri_from_current_file_nolang:Nn \l_stex_current_ns_uri {source-base}  
965   \%stex_debug:nn{modules}{Namespace~URI:~\stex_uri_use:N \l_stex_current_ns_uri}  
966 }
```

(End definition for `\stex_set_current_namespace`. This function is documented on page 128.)

We determine the PWD

```
\c_stex_pwd_file
```

```
\c_stex_main_file
```

```
967 \sys_if_platform_windows:TF{  
968   \stex_get_env:Nn\l_stex_path_str{CD}  
969 }{  
970   \stex_get_env:Nn\l_stex_path_str{PWD}  
971 }  
972 \stex_file_resolve:No \c_stex_pwd_file \l_stex_path_str  
973 \seq_set_eq:NN \c_stex_main_file \c_stex_pwd_file  
974 \seq_put_right:Nx \c_stex_main_file {\jobname\tl_to_str:n{.tex}}  
975  
976 \stex_debug:nn {files} {PWD:~\stex_file_use:N \c_stex_pwd_file}
```

(End definition for `\c_stex_pwd_file` and `\c_stex_main_file`. These variables are documented on page 127.)

### 13.2.9 File Hooks

keeps track of file changes:

```
977 \seq_gclear_new:N\g__stex_path_stack  
978 \seq_gclear_new:N\g_stex_current_file
```

```
\stex_filestack_push:n
```

```
979 \cs_new_protected:Nn \stex_filestack_push:n {  
980   \stex_str_if_ends_with:nnTF {\#1}{.tex}{  
981     \stex_file_resolve:No \g_stex_current_file {\#1}  
982   }{  
983     \stex_file_resolve:No \g_stex_current_file {\#1.tex}  
984   }  
985   \stex_if_file_absolute:NF \g_stex_current_file {  
986     \stex_file_resolve:Nx \g_stex_current_file {  
987       \stex_file_use:N \c_stex_pwd_file / \stex_file_use:N \g_stex_current_file  
988     }
```

```

989 }
990 \seq_gset_eq:NN \g_stex_current_file \g_stex_current_file
991 \exp_args:NNx \seq_gpush:Nn \g__stex_path_stack {\stex_file_use:N \g_stex_current_file}
992 \_stex_every_file:
993 }
994
995 \cs_new_protected:Nn \_stex_every_file: {
996   \stex_set_document_uri:
997   \stex_language_from_file:
998   \stex_set_current_namespace:
999 }
1000 \%AtBeginDocument{\_stex_every_file:}

(End definition for \stex_filestack_push:n. This function is documented on page 126.)

```

### \stex\_filestack\_pop:

```

1001 \cs_new_protected:Nn \stex_filestack_pop: {
1002   \seq_if_empty:NF \g__stex_path_stack {
1003     \seq_gpop:NN \g__stex_path_stack \l__stex_path_str
1004   }
1005   \seq_if_empty:NTF \g__stex_path_stack {
1006     \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
1007   }{
1008     \seq_get:NN \g__stex_path_stack \l__stex_path_str
1009     \exp_args:NNo \stex_file_set:Nn \g_stex_current_file \l__stex_path_str
1010     \seq_gset_eq:NN \g_stex_current_file \g_stex_current_file
1011   }
1012   \_stex_every_file:
1013 }

(End definition for \stex_filestack_pop:. This function is documented on page 126.)

```

Hooks for the current file:

```

1014 \cs_new_protected:Nn \stex_input_with_hooks:n {
1015   \tl_set:Nn \l__stex_path_do_hooks_pre_tl {
1016     \tl_gset:Nn \l__stex_path_do_hooks_pre_tl {}
1017     \stex_debug:nn{HERE}{Hook~for~#1^~J\meaning\CurrentFilePath^~J\CurrentFile}
1018     \tl_if_empty:NTF\CurrentFilePath{
1019       \exp_args:No \stex_filestack_push:n {\CurrentFile}
1020     }{
1021       \exp_args:Ne \stex_filestack_push:n { \CurrentFilePath / \CurrentFile }
1022     }
1023   }
1024   \input{#1}
1025   \stex_debug:nn{HERE}{Hook~end~for~#1}
1026   \stex_filestack_pop:
1027 }
1028 \tl_new:N \l__stex_path_do_hooks_pre_tl {}
1029
1030 \AddToHook{file/before}{
1031   \l__stex_path_do_hooks_pre_tl
1032 }
1033 \%AddToHook{file/after}{ \stex_filestack_pop: }


```

### 13.3 Math Archives

```

1034 <@=stex_mathhub>
\mathhub
\c_stex_home_file
\c_stex_mathhub_file
1035
1036 \sys_if_platform_windows:TF{
1037   \stex_get_env:Nn \l_stex_mathhub_str {homedrive\c_percent_str\c_percent_str homepath}
1038 }{
1039   \stex_get_env:Nn \l_stex_mathhub_str {HOME}
1040 }
1041 \stex_file_resolve:No \c_stex_home_file \l_stex_mathhub_str
1042
1043 \str_if_empty:NTF\mathhub{
1044   \stex_get_env:Nn \l_stex_mathhub_str {MATHHUB}
1045   \str_if_empty:NTF \l_stex_mathhub_str {
1046     \ior_open:NnTF \g_tmpa_ior{\stex_file_use:N \c_stex_home_file/.stex/mathhub.path}{

1047       \group_begin:
1048         \escapechar=-1\catcode`\=\12
1049         \ior_str_get:NN \g_tmpa_ior \l_stex_mathhub_str
1050         \str_gset_eq:NN \l_stex_mathhub_str \l_stex_mathhub_str
1051       \group_end:
1052         \ior_close:N \g_tmpa_ior
1053         \stex_debug:nn{mathhub}{MathHub-directory-determined-from-home-directory}
1054   }{
1055     \str_clear:N \l_stex_mathhub_str
1056   }
1057 }{
1058   \stex_debug:nn{mathhub}{MathHub-directory-determined-from-environment-variable}
1059 }
1060 }{
1061   \str_set_eq:NN \l_stex_mathhub_str \mathhub
1062 }
1063
1064 \str_if_empty:NTF \l_stex_mathhub_str {
1065   \msg_warning:nn{stex}{warning/nomathhub}
1066   \exp_args:NN \stex_file_set:Nn \c_stex_mathhub_file {\stex_file_use:N \c_stex_home_file \
1067   \seq_clear:N \c_stex_mathhub_file
1068 }{
1069   \stex_file_resolve:No \c_stex_mathhub_file \l_stex_mathhub_str
1070   \stex_if_file_absolute:NF \c_stex_mathhub_file {
1071     \exp_args:NN \stex_file_resolve:Nn \c_stex_mathhub_file {
1072       \stex_file_use:N \c_stex_main_file / .. / \l_stex_mathhub_str
1073     }
1074   }
1075 }
1076
1077 \exp_args:NN \str_set:Nn \mathhub {\stex_file_use:N \c_stex_mathhub_file}
1078 \stex_debug:nn{mathhub}{MATHHUB:~\mathhub}

(End 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
```

```

1079 \cs_new_protected:Nn \stex_set_current_archive:n {
1080   \stex_require_archive:n { #1 }
1081   \stex_debug:nn{mathhub}{switching-to-archive-#1}
1082   \prop_set_eq:Nc \l_stex_current_archive_prop {
1083     c_stex_mathhub_#1_manifest_prop
1084   }
1085 }
```

(End definition for `\l_stex_current_archive` and `\stex_set_current_archive:n`. These variables are documented on page ??.)

### `\stex_in_archive:nn`

```

1086 \cs_new_protected:Nn \stex_in_archive:nn {
1087   \cs_if_exist:NF \l_stex_mathhub_cs {
1088     \cs_set:Npn \l_stex_mathhub_cs ##1 {}
1089   }
1090   \stex_pseudogroup:nn{
1091     \cs_set:Npn \l_stex_mathhub_cs ##1 {#2}
1092     \tl_if_empty:nTF{#1} {
1093       \prop_if_exist:NTF \l_stex_current_archive_prop {
1094         \exp_args:Ne \l_stex_mathhub_cs {\prop_item:Nn \l_stex_current_archive_prop { id }
1095       }
1096       \l_stex_mathhub_cs {}
1097     }
1098   }{
1099     \stex_set_current_archive:n{#1}
1100     \l_stex_mathhub_cs {#1}
1101   }
1102 }{
1103   \stex_pseudogroup_restore:N \l_stex_current_archive_prop
1104   \cs_set:Npn \exp_not:N \l_stex_mathhub_cs ##1 {
1105     \exp_args:No \exp_not:n {\l_stex_mathhub_cs {##1}}
1106   }
1107 }
1108 }
```

(End definition for `\stex_in_archive:nn`. This function is documented on page 123.)

### `\stex_require_archive:n`

```

\stex_require_archive:o
1109 \cs_new_protected:Nn \stex_require_archive:n {
1110   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
1111     \seq_if_empty:NTF \c_stex_mathhub_file {
1112       \stex_fatal_error:n{warning/nomathhub}
1113     }{
1114       \stex_debug:nn{mathhub}{Opening-archive:-#1}
1115       \__stex_mathhub_do_manifest:n { #1 }
1116     }
1117   }
1118 }
```

`\cs_generate_variant:Nn \stex_require_archive:n {o}`

(End definition for `\stex_require_archive:n`. This function is documented on page 123.)

Code for finding and parsing manifest files:

```

1120 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
```

```

1121 \exp_args:Nn \__stex_mathhub_find_manifest:n {\stex_file_use:N \c_stex_mathhub_file / #1}
1122 \str_if_empty:NT \l__stex_mathhub_manifest_str {
1123     \stex_fatal_error:nxx{error/noarchive}
1124     {#1}{\stex_file_use:N \c_stex_mathhub_file}
1125 }
1126 \__stex_mathhub_parse_manifest:n {#1}
1127 }
1128
1129 \cs_new_protected:Nn \__stex_mathhub_find_manifest:n {
1130     \str_clear:N \l__stex_mathhub_manifest_str
1131     \seq_set_split:Nnn \l__stex_mathhub_seq / {#1}
1132     \bool_set_true:N \l__stex_mathhub_bool
1133     \bool_while_do:Nn \l__stex_mathhub_bool {
1134         \tl_if_eq:NNTF \l__stex_mathhub_seq \c_stex_mathhub_file {
1135             \bool_set_false:N \l__stex_mathhub_bool
1136         }{
1137             \__stex_mathhub_check_manifest:
1138             \bool_if:NT \l__stex_mathhub_bool {
1139                 \seq_pop_right:NN \l__stex_mathhub_seq \l__stex_mathhub_tl
1140             }
1141         }
1142     }
1143 }
1144 \cs_generate_variant:Nn \__stex_mathhub_find_manifest:n {x}
1145
1146 \cs_new_protected:Nn \__stex_mathhub_check_manifest: {
1147     \__stex_mathhub_check_manifest:n {MANIFEST.MF}
1148     \bool_if:NT \l__stex_mathhub_bool {
1149         \__stex_mathhub_check_manifest:n {META-INF/MANIFEST.MF}
1150         \bool_if:NT \l__stex_mathhub_bool {
1151             \__stex_mathhub_check_manifest:n {meta-inf/MANIFEST.MF}
1152         }
1153     }
1154 }
1155
1156 \cs_new_protected:Nn \__stex_mathhub_check_manifest:n {
1157     \stex_debug:nn{mathhub}{Checking~\stex_file_use:N \l__stex_mathhub_seq / #1}
1158     \file_if_exist:nT {\stex_file_use:N \l__stex_mathhub_seq / #1} {
1159         \bool_set_false:N \l__stex_mathhub_bool
1160         \str_set:Nx \l__stex_mathhub_manifest_str {\stex_file_use:N \l__stex_mathhub_seq / #1}
1161     }
1162 }
1163
1164 \ior_new:N \c__stex_mathhub_manifest_ior
1165 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
1166     \ior_open:Nn \c__stex_mathhub_manifest_ior \l__stex_mathhub_manifest_str
1167     \prop_clear:N \l__stex_mathhub_prop
1168     \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
1169         \exp_args:NNNo \exp_args:NNN
1170         \seq_set_split:Nnn \l__stex_mathhub_seq \c_colon_str {\tl_to_str:n{##1}}
1171         \seq_pop_left:NNT \l__stex_mathhub_seq \l__stex_mathhub_key {
1172             \exp_args:NNo \str_set:Nn \l__stex_mathhub_key \l__stex_mathhub_key
1173             \str_set:Nx \l__stex_mathhub_val {\seq_use:Nn \l__stex_mathhub_seq :}
1174             \str_case:Nn \l__stex_mathhub_key {

```

```

1175     {id}          {\prop_put:Nno \l__stex_mathhub_prop { id }      \l__stex_mathhub_
1176     {narration-base} {\prop_put:Nno \l__stex_mathhub_prop { narr }    \l__stex_mathhub_
1177     {url-base}      {\prop_put:Nno \l__stex_mathhub_prop { docurl }   \l__stex_mathhub_
1178     {source-base}   {\prop_put:Nno \l__stex_mathhub_prop { ns }       \l__stex_mathhub_
1179     {ns}            {\prop_put:Nno \l__stex_mathhub_prop { ns }       \l__stex_mathhub_
1180   }
1181 }
1182 }
1183 \ior_close:N \c__stex_mathhub_manifest_ior
1184 \prop_gset_eq:cN { c_stex_mathhub_#1_manifest_prop } \l__stex_mathhub_prop
1185 \stex_debug:nn{mathhub}{Result:~\prop_to_keyval:N \l__stex_mathhub_prop}
1186 \stex_persist:e {
1187   \prop_gset_from_keyval:cn {c_stex_mathhub_#1_manifest_prop} {
1188     \prop_to_keyval:N \l__stex_mathhub_prop
1189   }
1190 }
1191 }

```

Current MathHub archive

```

\c_stex_main_archive_prop
\l_stex_current_archive_prop
1192 \cs_new_protected:Nn \stex_main_archive: {
1193   \stex_if_file_starts_with:NNTF \c_stex_pwd_file \c_stex_mathhub_file {
1194     \__stex_mathhub_find_manifest:x { \stex_file_use:N \c_stex_pwd_file }
1195     \str_if_empty:NTF \l__stex_mathhub_manifest_str {
1196       \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~archive}
1197     }{
1198       \__stex_mathhub_parse_manifest:n { main }
1199       \prop_set_eq:NN \c_stex_main_archive_prop \c_stex_mathhub_main_manifest_prop
1200       \cs_undefine:N \c_stex_mathhub_main_manifest_prop
1201       \prop_get:NnN \c_stex_main_archive_prop {id}
1202         \l__stex_mathhub_str
1203       \prop_set_eq:cN { c_stex_mathhub_\l__stex_mathhub_str _manifest_prop }
1204         \c_stex_main_archive_prop
1205       \exp_args:No \stex_set_current_archive:n { \l__stex_mathhub_str }
1206       \stex_debug:nn{mathhub}{Current~archive:~
1207         \prop_item:Nn \l_stex_current_archive_prop {id}
1208       }
1209       \bool_if:NT \c_stex_persist_write_mode_bool {
1210         \tl_put_right:Nx \stex_persist_read_now: {
1211           \stex_persist:n {
1212             \prop_gset_from_keyval:cn {c_stex_mathhub_\l__stex_mathhub_str _manifest_prop} {
1213               \prop_to_keyval:N \c_stex_main_archive_prop
1214             }
1215             \prop_gset_eq:Nc \exp_not:N \l_stex_current_archive_prop {
1216               c_stex_mathhub_\l__stex_mathhub_str _manifest_prop
1217             }
1218             \prop_gset_eq:Nc \exp_not:N \c_stex_main_archive_prop {
1219               c_stex_mathhub_\l__stex_mathhub_str _manifest_prop
1220             }
1221           }
1222         }
1223       }
1224     }{
1225   }

```

```

1226     \stex_debug:nn{mathhub}{Not~currently~in~the~MathHub~directory}
1227 }
1228 }
1229 \%bool_if:NF \c_stex_persist_mode_bool {
1230   \_stex_main_archive:
1231 }
1232 }

(End definition for \c_stex_main_archive_prop and \l_stex_current_archive_prop. These variables
are documented on page 123.)
```

## 13.4 Documents

### 13.4.1 Title

Stores the title, if it exists:

```

1233 <@=stex_doc>
1234 \tl_new:N \g__stex_doc_title_tl
```

\stexdoctitle Initial definition, will be changed at begin document:

```

1235 \cs_new_protected:Npn \stexdoctitle #1 {
1236   \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1237   \global\def\stexdoctitle##1{}
1238 }
```

At begin document, we switch to:

```

1239 \cs_new_protected:Nn \__stex_doc_set_title:n {
1240   \stex_if_smsmode:F{
1241     \global\def\stexdoctitle##1{}
1242     \stex_debug:nn{title}{Setting~title~to:\tl_to_str:n##1}
1243     \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1244     \__stex_doc_title_html:
1245   }
1246 }
```

Hooks, changes and HTML:

```

1247 \cs_new_protected:Nn \__stex_doc_title_html: {
1248   \stex_if_do_html:T{\stex_if_html_backend:T{
1249     \stex_annotation_invisible:nn{shtml:doctitle={}}{ \hbox{\g__stex_doc_title_tl} }
1250   }}
1251 }

1252 \AtBeginDocument {
1253   \tl_if_empty:NTF \g__stex_doc_title_tl {
1254     \cs_set_eq:NN \stexdoctitle \__stex_doc_set_title:n
1255   }{
1256     \stex_debug:nn{title}{Setting~title~to:\exp_args:No\tl_to_str:n\g__stex_doc_title_tl}
1257     \global\def\stexdoctitle#1{}
1258     \__stex_doc_title_html:
1259   }

1260 \cs_set_eq:NN \__stex_doc_maketitle: \maketitle
1261 \global\protected\def\maketitle{
1262   \tl_if_empty:NF \@title {
```

```

1265     \exp_args:No \stexdoctitle \@title
1266   }
1267   \__stex_doc_maketitle:
1268 }
1269 }

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

### 13.4.2 Sectioning

```

1270 \int_new:N \l_stex_docheader_sect
1271
1272 \tl_set:Nn \stex_current_section_level {document}
1273
1274 \cs_set_protected:Npn \currentsectionlevel {
1275   \stex_if_do_html:TF{
1276     \stex_annotation:nn{shtml:currentsectionlevel={},shtml:capitalize=false}{}
1277   }{
1278     \stex_current_section_level
1279   }
1280   \tl_if_exist:NT\xspace\xspace
1281 }
1282
1283 \cs_set_protected:Npn \Currentsectionlevel {
1284   \stex_if_do_html:TF{
1285     \stex_annotation:nn{shtml:currentsectionlevel={},shtml:capitalize=true}{}
1286   }{
1287     \exp_args:No \stex_capitalize:n \stex_current_section_level
1288   }
1289   \tl_if_exist:NT\xspace\xspace
1290 }
1291
1292 \stex_if_html_backend:TF {
1293   \cs_new_protected:Nn \_sfragment_do_level:nn {
1294     \stexdoctitle{#2}
1295     \par
1296     \begin{stex_annotation_env}{shtml:section={\int_use:N \l_stex_docheader_sect}}
1297       \noindent\stex_annotation:nn{shtml:sectiontitle={}}{
1298         \stex_annotation_force_break:n{#2}
1299       }\par
1300     }
1301   \cs_new_protected:Nn \_sfragment_end: {
1302     \end{stex_annotation_env}
1303   }
1304 }
1305 \cs_new_protected:Nn \_sfragment_do_level:nn {
1306   \stexdoctitle{#2}
1307   \tl_if_empty:NTF \l_stex_key_short_tl {
1308     \use:c{#1}
1309   }{
1310     \exp_args:Nnx \use:nn{\use:c{#1}}{[\exp_args:No \exp_not:n \l_stex_key_short_tl]}
1311   }{#2}
1312   \int_incr:N \l_stex_docheader_sect
1313   \tl_set:Nn \stex_current_section_level{#1}
```

```

1314   }
1315   \cs_new_protected:Nn \_sfragment_end: {}
1316 }
1317
1318
1319 \cs_new_protected:Npn \__stex_doc_do_section:n {
1320   \int_case:nnF \l_stex_docheader_sect {
1321     {0}{\cs_if_exist:NTF \thepart {\_sfragment_do_level:nn{part}}{
1322       \int_incr:N \l_stex_docheader_sect
1323       \__stex_doc_do_section:n
1324     }}
1325     {1}{\cs_if_exist:NTF \thechapter {\_sfragment_do_level:nn{chapter}}{
1326       \int_incr:N \l_stex_docheader_sect
1327       \__stex_doc_do_section:n
1328     }}
1329     {2}{\_sfragment_do_level:nn{section}}
1330     {3}{\_sfragment_do_level:nn{subsection}}
1331     {4}{\_sfragment_do_level:nn{subsubsection}}
1332     {5}{\_sfragment_do_level:nn{paragraph}}
1333   }{\_sfragment_do_level:nn{subparagraph}}
1334 }
1335
1336 \stex_keys_define:nnnn{ sfragment }{
1337   \tl_clear:N \l_stex_key_short_tl
1338 }{
1339   short .tl_set:N = \l_stex_key_short_tl
1340 }{id}
1341
1342 \NewDocumentEnvironment{sfragment}{ O{} m}){
1343   \stex_keys_set:nn{ sfragment }{#1}
1344   \__stex_doc_do_section:n{#2}
1345   \stex_do_id:
1346 }{
1347   \_sfragment_end:
1348 }
1349
1350 %\int_incr:N \l_stex_docheader_sect
1351 \NewDocumentEnvironment{blindfragment}{}{
1352   \__stex_doc_skip_section:
1353 }{
1354   \stex_if_html_backend:T{
1355     \stex_annotation_invisible:n{~}
1356     \end{stex_annotation_env}
1357   }
1358 }
1359
1360
1361 \cs_new_protected:Nn \__stex_doc_skip_section_i: {
1362   \int_case:nn \l_stex_docheader_sect {
1363     {0}{\cs_if_exist:NF \thepart {
1364       \int_incr:N \l_stex_docheader_sect \__stex_doc_skip_section_i:
1365     }}
1366     {1}{\cs_if_exist:NF \thechapter {
1367       \int_incr:N \l_stex_docheader_sect \__stex_doc_skip_section_i:
1368     }}
1369   }
1370 }

```

```

1368     }}
1369   }
1370   \int_incr:N \l_stex_docheader_sect
1371 }
1372
1373 \stex_if_html_backend:TF {
1374   \cs_new_protected:Nn \__stex_doc_skip_section: {
1375     \__stex_doc_skip_section_i:
1376     \begin{stex_annotation_env}{\shtml:skipsection=\{\int_use:N \l_stex_docheader_sect\}}
1377     \stex_annotation_invisible:n{~}
1378   }
1379 }{
1380   \cs_set_eq:NN \__stex_doc_skip_section: \__stex_doc_skip_section_i:
1381 }
1382
1383
1384 \cs_new_protected:Nn \__stex_doc_skip_fragment:n {
1385   \stepcounter{#1}
1386 }
1387
1388 \cs_new_protected:Npn \skipfragment {
1389   \int_case:nnF \l_stex_docheader_sect {
1390     {0}{\cs_if_exist:NTF \thepart {\__stex_doc_skip_fragment:n{part}}{
1391       \int_incr:N \l_stex_docheader_sect
1392       \skipfragment
1393     }}
1394     {1}{\cs_if_exist:NTF \thechapter {\__stex_doc_skip_fragment:n{chapter}}{
1395       \int_incr:N \l_stex_docheader_sect
1396       \skipfragment
1397     }}
1398     {2}{\__stex_doc_skip_fragment:n{section}}
1399     {3}{\__stex_doc_skip_fragment:n{subsection}}
1400     {4}{\__stex_doc_skip_fragment:n{subsubsection}}
1401     {5}{\__stex_doc_skip_fragment:n{paragraph}}
1402   }{\__stex_doc_skip_fragment:n{ subparagraph}}
1403 }
1404
\setsectionlevel
1405 \cs_new_protected:Npn \setsectionlevel #1 {
1406   \str_case:nnF{#1}{
1407     {part}{\int_set:Nn \l_stex_docheader_sect 0}
1408     {chapter}{\int_set:Nn \l_stex_docheader_sect 1}
1409     {section}{\int_set:Nn \l_stex_docheader_sect 2}
1410     {subsection}{\int_set:Nn \l_stex_docheader_sect 3}
1411     {subsubsection}{\int_set:Nn \l_stex_docheader_sect 4}
1412     {paragraph}{\int_set:Nn \l_stex_docheader_sect 5}
1413   }{
1414     \int_set:Nn \l_stex_docheader_sect 6
1415   }
1416   \cs_if_eq:NNTF \onlypreamble \notprerr {
1417     \stex_annotation_invisible:nn{\shtml:sectionlevel=\{\int_use:N \l_stex_docheader_sect\}}{}}
1418 }
1419

```

```

1420 \stex_if_html_backend:T{
1421   \cs_new_protected:Nn \__stex_doc_check_topsect: {
1422     \int_case:nnF \l_stex_docheader_sect {
1423       {0}{\cs_if_exist:NTF \thechapter {
1424         \stex_annotation_invisible:nn{shtml:sectionlevel=0}{}}
1425       }{
1426         \int_incr:N \l_stex_docheader_sect
1427         \__stex_doc_check_topsect:
1428       }
1429     {1}{\cs_if_exist:NTF \thechapter {
1430       \stex_annotation_invisible:nn{shtml:sectionlevel=1}{}}
1431     }{
1432       \int_incr:N \l_stex_docheader_sect
1433       \__stex_doc_check_topsect:
1434     }
1435   }{
1436     \stex_annotation_invisible:nn{shtml:sectionlevel={\int_use:N \l_stex_docheader_sect}}{}
1437   }
1438 }
1439 \AtBeginDocument{\__stex_doc_check_topsect:}
1440 }
1441 \AtBeginDocument{
1442   \cs_if_exist:NTF\frontmatter{
1443     \let\__stex_doc_orig_frontmatter\frontmatter
1444     \let\frontmatter\relax
1445   }{
1446     \tl_set:Nn\__stex_doc_orig_frontmatter{
1447       \clearpage
1448       \%@\mainmatterfalse
1449       \pagenumbering{roman}
1450     }
1451   }
1452 }
1453 \cs_if_exist:NTF\backmatter{
1454   \let\__stex_doc_orig_backmatter\backmatter
1455   \let\backmatter\relax
1456 }{
1457   \tl_set:Nn\__stex_doc_orig_backmatter{
1458     \clearpage
1459     \%@\mainmatterfalse
1460     \pagenumbering{roman}
1461   }
1462 }
1463 \newenvironment{frontmatter} {
1464   \__stex_doc_orig_frontmatter
1465 }{
1466   \cs_if_exist:NTF\mainmatter{
1467     \mainmatter
1468   }{
1469     \clearpage
1470     \%@\mainmattertrue
1471     \pagenumbering{arabic}
1472   }
1473 }

```

```

1474 \newenvironment{backmatter}{%
1475     \__stex_doc_orig_backmatter
1476 }{%
1477     \cs_if_exist:N\mainmatter{%
1478         \mainmatter
1479     }{%
1480         \clearpage
1481         \%@\mainmattertrue
1482         \pagenumbering{arabic}
1483     }
1484 }
1485 }
```

(End definition for `\setsectionlevel`. This function is documented on page 71.)

### 13.4.3 References

```
1486 <@=stex_refs>
```

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

```

1487 \iow_new:N \c__stex_refs_iow
1488 \AtBeginDocument{\iow_open:Nn \c__stex_refs_iow {\jobname.sref}}
1489 \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>X2e character code scheme:

```

1490 \cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}
1491
1492 \cs_new_protected:Npn \STEXInternalSetSrefSymURL #1 #2 {
1493     \str_gset:cn{g_stex_sref_sym_\tl_to_str:n{#1}_target}{#2}
1494 }
1495
```

```

\stex_ref_new_doc_target:n
    \sreflabel
        1496 \seq_new:N \g__stex_refs_files_seq
        1497 \int_new:N \l__stex_refs_unnamed_counter_int
        1498
        1499 \cs_new_protected:Nn \stex_ref_new_id:n {
            1500     \str_if_empty:nTF {#1}{%
                1501         \int_gincr:N \l__stex_refs_unnamed_counter_int
                1502         \str_set:Nx \l__stex_refs_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
            }{%
                1504         \str_set:Nn \l__stex_refs_str {#1}
            }
            1506 \str_set:Nx \l_stex_ref_url_str {\stex_uri_use:N \l_stex_current_doc_uri ? \l__stex_refs_s
        1507 }
        1508
        1509 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
            1510     \stex_ref_new_id:n{#1}
            1511     \%stex_uri_add_module:NNo \l__stex_refs_uri \l_stex_current_doc_uri \l__stex_refs_str
            1512     \%stex_debug:nn{sref}{New-document-target:~\stex_uri_use:N \l__stex_refs_uri}
            1513     \__stex_refs_add_doc_ref:xo {\stex_uri_use:N \l_stex_current_doc_uri} \l__stex_refs_str
            1514     \stex_if_smemode:F {
                1515         \iow_now:Nx \c__stex_refs_iow {
```

```

1516     \STEXInternalSrefRestoreTarget
1517     {\l_stex_use:N \l_stex_current_doc_uri}
1518     {\l__stex_refs_str}
1519     {\@currentcounter}
1520     {\@currentlabel}
1521     {
1522         \tl_if_exist:NT\@currentlabelname{
1523             \exp_args:No\exp_not:n\@currentlabelname
1524         }
1525     }
1526 }
1527 \exp_args:Nx \label {sref@\l_stex_ref_url_str}
1528 \stex_if_do_html:T {
1529     \pdfdest name "sref@\l_stex_ref_url_str" xyz\relax
1530 }
1531 }
1532 }
1533 \NewDocumentCommand \sreflabel {m} {\stex_ref_new_doc_target:n {#1}}
1534
1535 \cs_new_protected:Nn \__stex_refs_add_doc_ref:nn {
1536     \seq_if_in:NnTF \g__stex_refs_files_seq {#1} {
1537         \seq_if_in:cnF {g__stex_refs_#1_seq}{#2} {
1538             \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1539         }
1540     }{
1541         \seq_gput_right:Nn \g__stex_refs_files_seq {#1}
1542         \seq_new:c{g__stex_refs_#1_seq}
1543         \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1544     }
1545 }
1546 \cs_generate_variant:Nn \__stex_refs_add_doc_ref:nn {xo,xx}

(End definition for \stex_ref_new_doc_target:n and \sreflabel. These functions are documented on page 110.)

```

\sref Optional arguments:

```

\extref
1547 \stex_keys_define:nnnn{sref / 1}{}{
1548     % TODO get rid of this
1549     fallback .code:n = {},
1550     pre .code:n = {},
1551     post .code:n = {}
1552 }{archive file}
1553 \stex_keys_define:nnnn{sref / 2}{}{}{archive file, title}
1554
1555 \str_new:N \l__stex_refs_default_archive_str
1556 \str_new:N \l__stex_refs_default_file_str
1557 \tl_new:N \l__stex_refs_default_title_tl
1558
1559 \cs_set_protected:Nn \__stex_refs_set_keys_b:n {
1560     \tl_if_empty:nTF{#1} {
1561         \str_set_eq:NN \l_stex_key_archive_str \l__stex_refs_default_archive_str
1562         \str_set_eq:NN \l_stex_key_file_str \l__stex_refs_default_file_str
1563         \tl_set_eq:NN \l_stex_key_title_tl \l__stex_refs_default_title_tl
1564     }

```

```

1565     \stex_keys_set:nn{ sref / 2 }{ #1 }
1566   }
1567 }
1568
1569 \newcommand\srefsetin[3][]{%
1570   \str_set:Nx \l__stex_refs_default_archive_str {#1}
1571   \str_set:Nx \l__stex_refs_default_file_str {#2}
1572   \tl_set:Nn \l__stex_refs_default_title_tl {#3}
1573 }
1574

Auxiliary methods:

1575 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1576   \str_clear:N \l__stex_refs_uri_str
1577   \stex_debug:nn{sref}{%
1578     File:~\l_stex_key_file_str^~J
1579     Repo:\l_stex_key_archive_str
1580   }
1581   \str_if_empty:NTF \l_stex_key_file_str {%
1582     \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1583     \seq_if_exist:cT{g__stex_refs_\stex_uri_use:N \l_stex_current_doc_uri _seq}{%
1584       \exp_args:Nnx \__stex_refs_find_uri_in_file:nnn{#1}%
1585         {\l_stex_uri_use:N \l_stex_current_doc_uri}\seq_map_break:
1586     }
1587     \str_if_empty:NT \l__stex_refs_uri_str {%
1588       \seq_map_inline:Nn \g__stex_refs_files_seq {%
1589         \__stex_refs_find_uri_in_file:nnn{#1}{##1}{\seq_map_break:n{\seq_map_break:}}%
1590       }
1591     }
1592   }{%
1593     \str_if_empty:NTF \l_stex_key_archive_str {%
1594       \prop_if_exist:NTF \l_stex_current_archive_prop {%
1595         \__stex_refs_find_uri_in_prop_file:N \l_stex_current_archive_prop
1596       }{%
1597         \stex_file_resolve:Nx \l__stex_refs_file
1598           { \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str }
1599         \str_set:Nx \l__stex_refs_uri_str { file:/ \stex_file_use:N \l__stex_refs_file }
1600       }
1601     }{%
1602       \stex_require_archive:o \l_stex_key_archive_str
1603       \prop_set_eq:Nc \l__stex_refs_prop { c_stex_mathhub_\l_stex_key_archive_str _manifest_ }
1604       \__stex_refs_find_uri_in_prop_file:N \l__stex_refs_prop
1605     }
1606   }
1607 }

1608 \cs_new_protected:Nn \__stex_refs_find_uri_in_prop_file:N {
1609   \str_set:Nx \l__stex_refs_uri_str {%
1610     \stex_file_use:N \c_stex_mathhub_file /
1611     \prop_item:Nn #1 {id} /
1612       source / \l_stex_key_file_str .sref
1613   }
1614 \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_uri_str
1615 \stex_uri_from_repo_file>NNNn \l__stex_refs_uri #1

```

```

1617     \l_stex_refs_file {narr}
1618     \str_set:Nx \l_stex_refs_uri_str {\stex_uri_use:N \l_stex_refs_uri}
1619 }
1620
1621 \cs_new_protected:Nn \__stex_refs_find_uri_in_file:nnn {
1622     \stex_debug:nn{sref}{Checking~file~#2}
1623     \seq_map_inline:cn{g_stex_refs_#2_seq}{
1624         \str_if_eq:nnT{#1}{##1}{
1625             \str_set:Nx \l_stex_refs_uri_str {\stex_uri_use:N \l_stex_current_doc_uri}
1626             \stex_debug:nn{sref}{Found.}
1627             #3
1628         }
1629     }
1630 }

```

Doing the actual referencing:

```

1631 \cs_new_protected:Nn \__stex_refs_do_autoref:n {
1632     \cs_if_exist:cTF{autoref} {
1633         \exp_args:Nx\autoref{sref@#1}
1634     }{
1635         \exp_args:Nx\ref{sref@#1}
1636     }
1637 }
1638
1639 \cs_new_protected:Nn \__stex_refs_do_sref:nn {
1640     \str_if_empty:NTF \l_stex_refs_uri_str {
1641         \str_if_empty:NTF \l_stex_key_file_str {
1642             \stex_debug:nn{sref}{autoref~on~\stex_uri_use:N \l_stex_current_doc_uri?#1}
1643             \exp_args:Ne \__stex_refs_do_autoref:n{\stex_uri_use:N \l_stex_current_doc_uri ? #1}
1644         }{
1645             \stex_debug:nn{sref}{srefin~on~#1}
1646             \__stex_refs_set_keys_b:n{ #2 }
1647             \__stex_refs_do_sref_in:n{#1}
1648         }
1649     }{
1650         \exp_args:NNo \seq_if_in:NnTF \g_stex_refs_files_seq \l_stex_refs_uri_str {
1651             \stex_debug:nn{sref}{Using~ref~file~\l_stex_refs_uri_str}
1652             \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref}\l_stex_refs_uri_str _seq{\detokenize{#1}}{
1653                 \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref~on~\l_stex_refs_uri_str?#1}
1654                 \__stex_refs_do_autoref:n{\l_stex_refs_uri_str?#1}
1655             }{
1656                 \str_if_empty:NTF \l_stex_key_file_str {
1657                     \stex_debug:nn{sref}{in~empty;~autoref~on~\l_stex_refs_uri_str?#1}
1658                     \__stex_refs_do_autoref:n{\l_stex_refs_uri_str?#1}
1659                 }{
1660                     \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l_stex_refs_uri_str?#1}
1661                     \__stex_refs_set_keys_b:n{ #2 }
1662                     \__stex_refs_do_sref_in:n{#1}
1663                 }
1664             }
1665         }{
1666             \stex_debug:nn{sref}{No~ref~file~found~for~\l_stex_refs_uri_str}
1667             \str_if_empty:NTF \l_stex_key_file_str {
1668                 \stex_debug:nn{sref}{in~empty;~autoref~on~\l_stex_refs_uri_str?#1}

```

```

1669     \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1670     }{
1671         \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l__stex_refs_uri_str?#1}
1672         \__stex_refs_set_keys_b:n{ #2 }
1673         \__stex_refs_do_sref_in:n{#1}
1674     }
1675 }
1676 }
1677 }
1678
1679 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1680     \stex_debug:nn{sref}{In: \l_stex_key_file_str^JRepo:\l_stex_key_archive_str}
1681     \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1682     \tl_if_exist:cTF{r@sref@}\l__stex_refs_uri_str?#1{
1683         \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1684     }{
1685         \%msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1686         \group_begin:\catcode13=9\relax\catcode10=9\relax
1687             \str_if_empty:NTF \l_stex_key_archive_str {
1688                 \prop_if_exist:NTF \l_stex_current_archive_prop {
1689                     \str_set:Nx \l__stex_refs_file_str {
1690                         \stex_file_use:N \c_stex_mathhub_file /
1691                         \prop_item:Nn \l_stex_current_archive_prop { id }
1692                         / source / \l_stex_key_file_str .sref
1693                     }
1694                 }{
1695                     \str_set:Nx \l__stex_refs_file_str {
1696                         \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str . sref
1697                     }
1698                 }
1699             }{
1700                 \str_set:Nx \l__stex_refs_file_str {
1701                     \stex_file_use:N \c_stex_mathhub_file / \l_stex_key_archive_str
1702                     / source / \l_stex_key_file_str . sref
1703                 }
1704             }
1705             \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_file_str
1706             \str_set:Nx \l__stex_refs_file_str {\stex_file_use:N \l__stex_refs_file }
1707             \stex_debug:nn{sref}{File: \l__stex_refs_file_str }
1708             \exp_args:NNNx \exp_args:No \str_if_eq:nnTF \l__stex_refs_file_str {\stex_file_use:N\c
1709                 \__stex_refs_do_autoref:n{
1710                     \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1711                 }
1712             }{
1713                 \exp_args:No \IfFileExists \l__stex_refs_file_str {
1714                     \tl_clear:N \l__stex_refs_return_tl
1715                     \str_set:Nn \l__stex_refs_id_str {#1}
1716                     \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1717                     \use:c{@ @ input}{\l__stex_refs_file_str}
1718                     \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1719                         \exp_args:Nnno \%msg_warning:nnnn{stex}{warning/smslabelmissing}\l__stex_refs_file
1720                         \__stex_refs_do_autoref:n{
1721                             \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1722                         }
1723             }

```

```

1723     }{
1724         \l__stex_refs_return_tl
1725     }
1726 }
1727 \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l__stex_refs_file_str
1728 \__stex_refs_do_autoref:n{
1729     \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1730 }
1731 }
1732 }
1733 \group_end:
1734 }
1735 }

1736 \cs_new_protected:Nn \__stex_refs_do_return:nnnn {
1737     \tl_set:Nn \l__stex_refs_return_tl {
1738         \stex_annotation:nn{shtml:sref={#4},shtml:srefin={\l__stex_refs_file_str}}{
1739             \use:c{#3autorefname}~#1\tl_if_empty:nF{#2}{~(#2)}
1740             \tl_if_empty:NF\l_stex_key_title_tl{
1741                 {}~in~\l_stex_key_title_tl
1742             }
1743         }
1744     }
1745 }
1746 }

1747 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1748     \str_if_empty:NTF \l__stex_refs_uri_str {
1749         \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2} {
1750             \__stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
1751         }
1752     }
1753 }
1754 \stex_debug:nn{sref}{\l__stex_refs_uri_str{}~ == ~ #1 ~ ?}
1755 \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1} {
1756     \stex_debug:nn{sref}{\l__stex_refs_id_str{}~ == ~ #2 ~ ?}
1757     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2} {
1758         \stex_debug:nn{sref}{success!}
1759         \__stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
1760         \endinput
1761     }
1762 }
1763 }
1764 }

```

The actual macros:

```

1765 \NewDocumentCommand \sref { O{} m O{} }{
1766     \stex_keys_set:nn { sref / 1 }{ #1 }
1767     \__stex_refs_find_uri:n { #2 }
1768     \__stex_refs_do_sref:nn{#2}{#3}
1769 }
1770 \NewDocumentCommand \extref { O{} m m }{
1771     \stex_keys_set:nn { sref / 1 }{ #1 }
1772     \__stex_refs_find_uri:n { #2 }
1773     \__stex_refs_set_keys_b:n{ #3 }
1774     \str_if_empty:NT \l_stex_key_file_str {
1775         \msg_error:nn{stex}{error/extrefmissing}

```

```

1776   }
1777   \__stex_refs_do_sref_in:n{#2}
1778 }
```

(End definition for `\sref` and `\extref`. These functions are documented on page 73.)

### `\stex_ref_new_sym_target:n`

```

1779 \cs_new_protected:Nn \stex_ref_new_symbol:n {
1780   \cs_if_exist:cF{r@sref@sym@\tl_to_str:n{#1}}{
1781     \__stex_refs_new_symbol:n{#1}
1782   }
1783 }
1784
1785 \cs_new_protected:Nn \stex_sref_do_aux:n {
1786   #1 \iow_now:Nn \auxout {#1}
1787 }
1788
1789 \cs_new_protected:Nn \__stex_refs_new_symbol:n {
1790   \prop_if_exist:NTF \l_stex_current_archive_prop {
1791     \prop_get:NnNTF \l_stex_current_archive_prop {docurl} \l__stex_refs_str {
1792       \exp_args:Ne \stex_sref_do_aux:n {
1793         \STEXInternalSetSrefSymURL{#1}{\l__stex_refs_str / symbol? #1}
1794       }
1795     }{
1796       \stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
1797     }
1798   }{
1799     \stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
1800   }
1801 }
1802
1803 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1804   \exp_args:Ne \label{\tl_to_str:n{sref@sym@ #1}}
1805 }
1806
1807 \cs_new_protected:Nn \stex_ref_new_sym_target:nn {
1808   \str_if_eq:nnTF{#1}{#2} {
1809     \stex_ref_new_sym_target:n{#1}
1810   }{
1811     \str_gset:cn{g_stex_sref_sym_ #1 _label}{#2}
1812   }
1813 }
```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 110.)

### `\srefsym`

```

1814 \NewDocumentCommand \srefsym { m m }{
1815   \stex_get_symbol:n { #1 }
1816   \exp_args:Ne
1817   \__stex_refs_sym_aux:nn{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_uri_str}
1818 }
1819
1820 \cs_new_protected:Npn \__stex_refs_do_internal_link:nn #1 {
1821   \cs_if_exist:NTF \hyperref {
1822     \hyperref[{#1}]}
```

```

1823     }\use:n
1824 }
1825
1826 \cs_new_protected:Npn \__stex_refs_do_url_link:nn {
1827     \cs_if_exist:NTF \href \href \use_i:nn
1828 }
1829
1830 \cs_new_protected:Npn \__stex_refs_sym_aux:nn #1 {
1831     \cs_if_exist:cTF{r@sref@sym@\tl_to_str:n{#1}}{
1832         \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1833     }{
1834         \str_if_exist:cTF{g_stex_sref_sym_#1_label}{
1835             \exp_args:Ne \__stex_refs_sym_aux:nn{\use:c{g_stex_sref_sym_#1_label}}
1836         }{
1837             \str_if_empty:cTF{g_stex_sref_sym_#1_target}{
1838                 \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1839             }{
1840                 \exp_args:Ne \__stex_refs_do_url_link:nn{\use:c{g_stex_sref_sym_#1_target}}
1841             }
1842         }
1843     }
1844 }
```

(End definition for `\srefsym`. This function is documented on page 78.)

#### \srefsymuri

```

1845 \cs_new_protected:Npn \srefsymuri #1 {
1846     \__stex_refs_sym_aux:nn{#1}
1847 }
```

(End definition for `\srefsymuri`. This function is documented on page 78.)

### 13.4.4 Inputs

`\stex_resolve_path_pair:Nnn`

```

\stex_resolve_path_pair:Nxx
1849 \cs_new_protected:Nn \stex_resolve_path_pair:Nnn {
1850     \stex_debug:nn{resolving-path}{#3~in~[#2]}
1851     \str_if_empty:nTF{#2} {
1852         \prop_if_exist:NTF \l_stex_current_archive_prop {
1853             \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1854             \prop_item:Nn \l_stex_current_archive_prop { id } / source /
1855             #3}
1856             \stex_debug:nn{resolving-path}{In-current-archive-
1857             \prop_item:Nn \l_stex_current_archive_prop { id }
1858             ;~result:~#1}
1859         }{
1860             \str_set:Nx #1 {\stex_file_use:N \c_stex_pwd_file / .. / #3 }
1861             \stex_debug:nn{resolving-path}{No-current-archive;~result:~#1}
1862         }
1863     }{
1864         \stex_require_archive:n { #2 }
1865         \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1866             \prop_item:cn {c_stex_mathhub_#2_manifest_prop} { id } / source /
```

```

1867     #3}
1868     \stex_debug:nn{resolving-path}{result:~#1}
1869   }
1870 }
1871 \cs_generate_variant:Nn \stex_resolve_path_pair:Nnn {Nxx}

```

(End definition for `\stex_resolve_path_pair:Nnn`. This function is documented on page 123.)

```

\inputref
\mhinput
\ifinputref
1872 \newif \ifinputref \inputreffalse
1873
1874 \cs_new_protected:Nn \__stex_inputs_mhinput:nn {
1875   \stex_in_archive:nn {#1} {
1876     \ifinputref
1877       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1878     \else
1879       \inputreftrue
1880       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1881       \inputreffalse
1882     \fi
1883   }
1884 }
1885
1886 \NewDocumentCommand \mhinput { O{} m }{
1887   \exp_args:NNx \exp_args:Nnx \__stex_inputs_mhinput:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2} }
1888 }
1889
1890 \cs_new_protected:Nn \__stex_inputs_inputref_html:nn {
1891   \str_clear:N \l_tmpa_str
1892   \prop_get:NnNF \l_stex_current_archive_prop { narr } \l_tmpa_str {
1893     \prop_get:NnNF \l_stex_current_archive_prop { ns } \l_tmpa_str {}
1894   }
1895   \tl_if_empty:nTF{ #1 }{
1896     \IfFileExists{#2} {
1897       \ifvmode\noindent\fi \stex_annotation_invisible:nn{ shtml:inputref={ \l_tmpa_str / #2 } }
1898     }{ }
1899   }{
1900     \stex_input_with_hooks:n{#2}
1901   }
1902 }{
1903   \IfFileExists{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }{
1904     \par \stex_annotation_invisible:nn{ shtml:inputref={ \l_tmpa_str / #2 } }
1905   }{ ~ }
1906   \input{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1907 }
1908 }{
1909   \input{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1910 }
1911 }
1912 }
1913
1914 \cs_new_protected:Nn \__stex_inputs_inputref_pdf:nn {
1915   \begingroup
1916     \inputreftrue

```

```

1917   \tl_if_empty:nTF{ #1 }{
1918     \stex_input_with_hooks:n{#2}
1919   }{
1920     \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1921   }
1922 \endgroup
1923 }
1924
1925 \cs_new_protected:Nn \__stex_inputs_inputref:nn {
1926   \stex_in_archive:nn {#1} {
1927     \stex_if_html_backend:TF
1928       \__stex_inputs_inputref_html:nn
1929       \__stex_inputs_inputref_pdf:nn
1930     {##1}{#2}
1931   }
1932 }
1933
1934 \NewDocumentCommand \inputref { O{} m }{
1935   \exp_args:NNx \exp_args:Nnx \__stex_inputs_inputref:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2} }
1936 }

```

(End definition for `\inputref`, `\mhinput`, and `\ifinputref`. These functions are documented on page 72.)

### \addmhbibresource

```

1937 \cs_new_protected:Nn \__stex_inputs_bibresource:n {
1938   \__stex_inputs_up_archive:nn{#1}{bib}
1939   \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
1940     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N \addmhbibresource}{#1.bib}
1941   }{
1942     \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
1943       \addbibrssource{ ##1 }
1944     }
1945   }
1946 }
1947 \newcommand\addmhbibresource[2][]{%
1948   \tl_if_empty:nTF{#1}{%
1949     \__stex_inputs_bibresource:n{#2}
1950   }{
1951     \stex_in_archive:nn{#1}{\__stex_inputs_bibresource:n{#2}}
1952   }
1953 }

```

(End definition for `\addmhbibresource`. This function is documented on page 69.)

### \IfInputref

```

1954 \stex_if_html_backend:TF{
1955   \newcommand \IfInputref[2]{%
1956     \stex_annotation:n{ shtml:ifinputref=true }{#1}
1957     \stex_annotation:n{ shtml:ifinputref=false }{#2}
1958   }
1959 }{
1960   \newcommand \IfInputref[2]{%
1961     \ifinputref #1 \else #2 \fi
1962   }

```

1963 }

(End definition for `\IfInputref`. This function is documented on page 72.)

### \libinput

```
1964 \cs_new_protected:Nn \__stex_inputs_up_archive:nn {
1965   \prop_if_exist:NF \l_stex_current_archive_prop {
1966     \msg_error:nnn{stex}{error/notinarchive}\libinput
1967   }
1968   \prop_get:NnNF \l_stex_current_archive_prop {id} \l__stex_inputs_id_str {
1969     \msg_error:nnn{stex}{error/notinarchive}\libinput
1970   }
1971   \seq_clear:N \l__stex_inputs_libinput_files_seq
1972   \seq_set_eq:NN \l__stex_inputs_path_seq \c_stex_mathhub_file
1973   \seq_set_split:NnV \l__stex_inputs_id_seq / \l__stex_inputs_id_str
1974
1975   \bool_while_do:nn { ! \seq_if_empty_p:N \l__stex_inputs_id_seq }{
1976     \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / meta-i
1977     \IfFileExists{ \l__stex_inputs_path_str }{
1978       \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
1979     }{}
1980     \seq_pop_left:NN \l__stex_inputs_id_seq \l__stex_inputs_path_str
1981     \seq_put_right:No \l__stex_inputs_path_seq \l__stex_inputs_path_str
1982   }
1983
1984   \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / lib / #1
1985   \IfFileExists{ \l__stex_inputs_path_str }{
1986     \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
1987   }{}
1988 }
1989
1990 \cs_new_protected:Nn \__stex_inputs_libinput:n {
1991   \__stex_inputs_up_archive:nn{#1}{tex}
1992   \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
1993     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
1994   }{
1995     \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
1996       \input{ ##1 }
1997     }
1998   }
1999 }
2000
2001 \newcommand \libinput [2] [] {
2002   \tl_if_empty:nTF{#1}{
2003     \__stex_inputs_libinput:n{#2}
2004   }{
2005     \stex_in_archive:nn{#1}{\__stex_inputs_libinput:n{#2}}
2006   }
2007 }
```

(End definition for `\libinput`. This function is documented on page 69.)

### \libusepackage

```
2008 \newcommand\libusepackage[2] [] {
2009   \__stex_inputs_up_archive:nn{#2}{sty}
```

```

2010 \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2011   \str_set:Nx \l__stex_inputs_tmp_str {\seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
2012   \exp_args:Nne \use:n {\usepackage[#1]} {
2013     \str_range:Nnn\l__stex_inputs_tmp_str 1 {-5}
2014   }
2015 }{
2016   \stex_fatal_error:nnn{error/nofile}{\libusepackage}{#1.sty}
2017 }
2018 }

```

(End definition for `\libusepackage`. This function is documented on page 69.)

```

\mhgraphics
\cmhgraphics
\lstinputmhlisting
\clstinputmhlisting
\mhtikzinput
\cmhtikzinput
2019 \str_new:N \l__stex_inputs_gin_repo_str
2020 \ltx@ifpackageloaded{graphicx}{\use:n}{\AtEndOfPackageFile{graphicx}}{
2021   \define@key{Gin}{archive}{
2022     \tl_set:Nx\Gin@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2023   }
2024 \providecommand\mhgraphics[2][]{%
2025   \tl_set:Nx\Gin@mhrepos{%
2026     \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop fid
2027   }
2028   \setkeys{Gin}{#1}
2029   \includegraphics[#1]{ \Gin@mhrepos #2 }
2030 }
2031 \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
2032 }
2033
2034 \ltx@ifpackageloaded{listings}{\use:n}{\AtEndOfPackageFile{listings}}{
2035   \define@key{lst}{archive}{
2036     \def\lst@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2037   }
2038 \newcommand\lstinputmhlisting[2][]{%
2039   \def\lst@mhrepos{%
2040     \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop fid
2041   }
2042   \setkeys{lst}{#1}%
2043   \lstinputlisting[#1]{\lst@mhrepos #2}
2044 \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
2045 }
2046
2047 \ltx@ifpackageloaded{tikzinput}{\use:n}{\AtEndOfPackageFile{tikzinput}}{
2048   \define@key{Gin}{archive}{
2049     \str_set:Nn \l__stex_inputs_gin_repo_str {#1}
2050     \def\Gin@mhrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2051   }
2052 \newcommand\mhtikzinput[2][]{%
2053   \str_clear:N \l__stex_inputs_gin_repo_str
2054   \def\Gin@mhrepos{%
2055     \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop fid
2056   }
2057   \setkeys{Gin}{#1}%
2058   \exp_args:No \stex_in_archive:nn \l__stex_inputs_gin_repo_str {
2059     \tikzinput[#1]{\Gin@mhrepos #2}

```

```

2060     }
2061   }
2062 \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
2063 }

```

(End definition for `\mhgraphics` and others. These functions are documented on page 72.)

#### `\libusetikzlibrary`

```

2064 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary:nn {
2065   \pgfkeys@spdef\pgf@temp{#1}
2066   \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
2067   \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfutil
2068   \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode`@}
2069   \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode`\|}%
2070   \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode`\$}
2071   \catcode`\@=11
2072   \catcode`\|=12
2073   \catcode`\$=3
2074   \pgfutil@InputIfFileExists{#2}{}{%
2075     \catcode`\@=\csname tikz@library@#1@atcode\endcsname
2076     \catcode`\|= \csname tikz@library@#1@barcode\endcsname
2077     \catcode`\$= \csname tikz@library@#1@dollarcode\endcsname
2078   }
2079
2080 \cs_new_protected:Npn \libusetikzlibrary #1 {
2081   \cs_if_exist:NF \usetikzlibrary {
2082     \msg_error:nnx{stex}{error/notikz}{\tl_to_str:n{\libusetikzlibrary}}
2083   }
2084   \__stex_inputs_up_archive:nn{tikzlibrary#1}{code.tex}
2085   \int_compare:nNnTF {\seq_count:N \__stex_inputs_libinput_files_seq} = 1 {
2086     \exp_args:Nne \__stex_inputs_usetikzlibrary:nn{#1}{ \seq_item:Nn \__stex_inputs_libinpu
2087   }{
2088     \stex_fatal_error:nnn{error/nofile}{\libusetikzlibrary}{tikzlibrary#1.code.tex}
2089   }
2090 }

```

(End definition for `\libusetikzlibrary`. This function is documented on page 107.)

## 13.5 SMS Mode

2091 `<@@=stex_smsmode>`

Macros and environments allowed in sms mode:

```

2092 \tl_new:N \g__stex_smsmode_allowed_tl
2093 \tl_new:N \g__stex_smsmode_allowed_escape_tl
2094 \seq_new:N \g__stex_smsmode_allowedenvs_seq

\stex_sms_allow:N
\stex_sms_allow_escape:N
\stex_sms_allow_env:n
2095 \cs_new_protected:Nn \stex_sms_allow:N {
2096   \tl_gput_right:Nn \g__stex_smsmode_allowed_tl {#1}
2097 }
2098
2099 \cs_new_protected:Nn \stex_sms_allow_escape:N {
2100   \tl_gput_right:Nn \g__stex_smsmode_allowed_escape_tl {#1}
2101 }

```

```

2102 \cs_new_protected:Nn \stex_sms_allow_env:n {
2103   \exp_args:NNx \seq_gput_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{#1}}
2104 }
2105 }
```

(End definition for `\stex_sms_allow:N`, `\stex_sms_allow_escape:N`, and `\stex_sms_allow_env:n`. These functions are documented on page 124.)

Some initial allowed macros:

```

2106 \stex_sms_allow:N \makeatletter
2107 \stex_sms_allow:N \makeatother
2108 \stex_sms_allow:N \ExplSyntaxOn
2109 \stex_sms_allow:N \ExplSyntaxOff
2110 \stex_sms_allow:N \rustexBREAK
```

#### `\stex_if_smsmode_p:`

#### `\stex_if_smsmode:TF`

```

2111 \bool_new:N \g_stex_smsmode_bool
2112 \bool_set_false:N \g_stex_smsmode_bool
2113 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2114   \bool_if:NTF \g_stex_smsmode_bool \prg_return_true: \prg_return_false:
2115 }
```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 124.)

#### `\stex_sms_allow_import:Nn`

#### `\stex_sms_allow_import_env:nn`

```

2116 \tl_new:N \g_stex_smsmode_allowed_import_tl
2117 \seq_new:N \g_stex_smsmode_allowed_import_env_seq
2118 \cs_new_protected:Nn \stex_sms_allow_import:Nn {
2119   \tl_gput_right:Nn \g_stex_smsmode_allowed_import_tl {#1}
2120   \tl_gset:cn{\tl_to_str:n{#1}---smsmode} {#2}
2121 }
2122 \cs_new_protected:Nn \stex_sms_allow_import_env:nn {
2123   \exp_args:NNx \seq_gput_right:Nn \g_stex_smsmode_allowed_import_env_seq {\tl_to_str:n{#1}}
2124   \tl_gset:cn{\tl_to_str:n{#1}---env---smsmode} {#2}
2125 }
2126
2127 \tl_new:N \g_stex_sms_import_code
```

(End definition for `\stex_sms_allow_import:Nn` and `\stex_sms_allow_import_env:nn`. These functions are documented on page 125.)

#### `\stex_file_in_smsmode:nn`

#### `\stex_file_in_smsmode:on`

```

2128 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:n { \stex_suppress_html:n {
2129   \vbox_set:Nn \l_tmpa_box {
2130     \bool_set_true:N \g_stex_smsmode_bool
2131     \bool_set_false:N \stex_html_do_output_bool
2132     #1
2133   }
2134   \%box_clear:N \l_tmpa_box
2135 } }
2136
2137 \quark_new:N \q_stex_smsmode_break
2138
2139 \cs_new_protected:Nn \__stex_smsmode_start_smsmode:n {
2140   \everyeof{\q_stex_smsmode_break\exp_not:N}
2141   \let\stex_smsmode_do:\__stex_smsmode_smsmode_do:
```

```

2142 \exp_after:wN \exp_after:wN \exp_after:wN
2143 \stex_smsmode_do:
2144 \cs:w @ @ input\cs_end: "#1" \relax
2145 }
2146
2147 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2148   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2149   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2150   \tl_clear:N \g_stex_sms_import_code
2151   \group_begin:
2152     \let \l_stex_metatheory_uri \c_stex_default_metalanguage
2153     \cs_set:Npn \stex_check_term:n #1 {}
2154     \seq_clear:N \l_stex_all_modules_seq
2155     \str_clear:N \l_stex_current_module_str
2156     #2
2157     \stex_filestack_push:n{#1}
2158     \__stex_smsmode_in_smsmode:n {
2159       \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_imports:N
2160       \tl_map_inline:Nn \g_stex_smsmode_allowed_import_tl {
2161         \use:c{\tl_to_str:n{##1}~~~smsmode}
2162       }
2163       \seq_map_inline:Nn \g_stex_smsmode_allowed_import_env_seq {
2164         \use:c{\tl_to_str:n{##1}~~~env~~~smsmode}
2165       }
2166       \__stex_smsmode_start_smsmode:n{#1}
2167     }
2168     \__stex_smsmode_in_smsmode:n \g_stex_sms_import_code
2169     \__stex_smsmode_in_smsmode:n {
2170       \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_normal:N
2171       \__stex_smsmode_start_smsmode:n{#1}
2172     }
2173     \stex_filestack_pop:
2174   \group_end:
2175 }
2176 \cs_generate_variant:Nn \stex_file_in_smsmode:nn {on}

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 124.)

```

\stex_smsmode_do:
2177 \cs_new_protected:Nn \__stex_smsmode_smsmode_do: {
2178   \%stex_if_smsmode:T {
2179     \__stex_smsmode_do:w
2180   \%}
2181 }
2182 \let\stex_smsmode_do:\relax
2183
2184
2185 \cs_new:Nn \__stex_smsmode_check_cs:NNn {
2186   \exp_after:wN\if\exp_after:wN\relax\exp_not:N#3
2187   \exp_after:wN#1\exp_after:wN#3\else
2188   \exp_after:wN#2\fi
2189 }
2190
2191 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {

```

```

2192 \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2193     \__stex_smsemode_check_cs:NNn \__stex_smsemode_do_aux:N \__stex_smsemode_do:w { #1 }
2194 }{
2195     \__stex_smsemode_do:w
2196 }
2197 }
2198
2199 \cs_new_protected:Nn \__stex_smsemode_do_aux:N {
2200     \cs_if_eq:NNF #1 \q__stex_smsemode_break {
2201         \__stex_smsemode_do_aux_curr:N #1
2202     }
2203 }
2204
2205 \cs_new_protected:Nn \__stex_smsemode_do_aux_imports:N {
2206     % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~import}
2207     \tl_if_in:NnTF \g__stex_smsemode_allowed_import_tl {#1} {
2208         \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}~in~import}
2209         #1
2210     }{
2211         \cs_if_eq:NNTF \begin{ #1 {
2212             \__stex_smsemode_check_begin:Nn \g__stex_smsemode_allowed_import_env_seq
2213         }{
2214             \cs_if_eq:NNTF \end #1 {
2215                 \__stex_smsemode_check_end:Nn \g__stex_smsemode_allowed_import_env_seq
2216             }{
2217                 \__stex_smsemode_do:w
2218             }
2219         }
2220     }
2221 }
2222
2223 \cs_new_protected:Nn \__stex_smsemode_do_aux_normal:N {
2224     % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~sms~mode}
2225     \tl_if_in:NnTF \g__stex_smsemode_allowed_tl {#1} {
2226         \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}}
2227         #1\__stex_smsemode_do:w
2228     }{
2229         \tl_if_in:NnTF \g__stex_smsemode_allowed_escape_tl {#1} {
2230             \stex_debug:nn{sms}{Executing~escaped~\tl_to_str:n{#1}}
2231             #1
2232         }{
2233             \cs_if_eq:NNTF \begin #1 {
2234                 \__stex_smsemode_check_begin:Nn \g__stex_smsemode_allowedenvs_seq
2235             }{
2236                 \cs_if_eq:NNTF \end #1 {
2237                     \__stex_smsemode_check_end:Nn \g__stex_smsemode_allowedenvs_seq
2238                 }{
2239                     \__stex_smsemode_do:w
2240                 }
2241             }
2242         }
2243     }
2244 }
2245

```

```

2246 \cs_new_protected:Nn \__stex_smsmode_check_begin:Nn {
2247   % \stex_debug:nn{sms}{Checking~environment~#2}
2248   \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2249     \stex_debug:nn{sms}{Environment~#2}
2250     \begin{#2}
2251   }{
2252     \__stex_smsmode_do:w
2253   }
2254 }
2255 \cs_new_protected:Nn \__stex_smsmode_check_end:Nn {
2256 % \stex_debug:nn{sms}{Checking~end~environment~#2}
2257 \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2258   \stex_debug:nn{sms}{End~Environment~#2}
2259   \end{#2}\__stex_smsmode_do:w
2260 }{
2261   \%str_if_eq:nnTF{#2}{document} \endinput
2262   \__stex_smsmode_do:w
2263 }
2264 }

```

(End definition for `\stex_smsmode_do:`. This function is documented on page 125.)

## 13.6 Modules

### 13.6.1 The smodule-environment

```
2265 <@=stex_modules>
```

`\l_stex_current_module_str` The current module:

```
2266 \str_new:N \l_stex_current_module_str
```

(End definition for `\l_stex_current_module_str`. This variable is documented on page 111.)

`\l_stex_all_modules_seq` Stores all modules currently in scope

```
2267 \seq_new:N \l_stex_all_modules_seq
```

(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 111.)

`\stex_every_module:n`

```

2268 <@=stex_module_setup>
2269 \tl_clear:N \g_stex_every_module_tl {
2270 }
2271 \cs_new_protected:Nn \stex_every_module:n {
2272   \tl_gput_right:Nn \g_stex_every_module_tl { #1 }
2273 }
```

(End definition for `\stex_every_module:n`. This function is documented on page 111.)

`\stex_module_setup:n` Sets up a new module:

```

2274 \cs_new_protected:Npn \stex_module_setup:n {
2275   \stex_if_in_module:TF \__stex_module_setup_setup_nested:n \__stex_module_setup_setup_top:n
2276 }
2277 \cs_new_protected:Nn \__stex_module_setup_setup_top:n {
2278   \__stex_module_setup_get_uri_str:n{#1}
```

```

2280 \stex_debug:nn{module}{Module~URI:~\l__stex_module_setup_ns_str?#1}
2281 \str_if_empty:NTF \l_stex_key_sig_str
2282 \_stex_module_setup_top_nosig:n \_stex_module_setup_setup_top_sig:n {\l__stex_module_setu
2283 \stex_metagroup_new:o \l_stex_current_module_str
2284 \g_stex_every_module_tl
2285 \stex_execute_in_module:x {
2286     \_stex_do_deprecation:n{#1}
2287 }
2288 \_\_stex_module_setup_load_meta:
2289 }

2290
2291 \cs_new_protected:Nn \_stex_module_setup_top_nosig:n {
2292     \stex_if_module_exists:nTF{#1}{
2293         \stex_debug:nn{modules}{(already exists)}
2294     }{
2295         \tl_gclear:c{c_stex_module_ #1 _code}
2296         \prop_gclear:c{c_stex_module_ #1 _morphisms_prop }
2297         \prop_gclear:c{c_stex_module_ #1 _symbols_prop }
2298         \prop_gclear:c{c_stex_module_ #1 _notations_prop }
2299     }
2300     \str_set:Nx \l_stex_current_module_str {#1}
2301     \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
2302 }

2303
2304 \cs_new_protected:Nn \_\_stex_module_setup_setup_top_sig:n {
2305     \stex_if_module_exists:nTF{#1}{
2306         \stex_debug:nn{modules}{(already exists)}
2307     }{
2308         \stex_debug:nn{modules}{(needs loading)}
2309         \_\_stex_module_setup_load_sig:
2310     }
2311 \% \stex_if_smsmode:F { % WHY?
2312     \stex_activate_module:x {
2313         #1
2314     }
2315 \%}
2316     \str_set:Nx\l_stex_current_module_str{#1}
2317 }

2318
2319 \cs_new_protected:Nn \_\_stex_module_setup_load_sig: {
2320     \stex_file_split_off_ext:NN \l__stex_module_setup_sigfile \g_stex_current_file
2321     \stex_file_split_off_lang:NN \l__stex_module_setup_sigfile \l__stex_module_setup_sigfile
2322     \exp_args:Ne \stex_file_in_smsmode:nn {
2323         \stex_file_use:N \l__stex_module_setup_sigfile . \l_stex_key_sig_str . tex
2324     }{}
2325 }

2326
2327 \cs_new_protected:Nn \_\_stex_module_setup_setup_nested:n {
2328     \exp_after:wN
2329         \_\_stex_module_setup_split_module:n \l_stex_current_module_str \_\_stex_module_setup_end:
2330     \stex_debug:nn{module}{Nested~Module~URI:~\l_stex_current_module_str}
2331     \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
2332     \stex_metagroup_new:o \l_stex_current_module_str
2333 }

```

```

2334
2335
2336 \cs_new_protected:Nn \__stex_module_setup_get_uri_str:n {
2337   \str_clear:N \l__stex_module_setup_ns_str
2338   \stex_map_uri:Nnnnn \l__stex_current_ns_uri {
2339     \str_set:Nx \l__stex_module_setup_ns_str{##1\cColonStr/}
2340   }{
2341     \seq_set_split:Nnn \l__stex_module_setup_seq / {##1}
2342     \seq_pop_right:NN \l__stex_module_setup_seq \l__stex_module_setup_seg
2343     \exp_args:No \str_if_eq:nnF \l__stex_module_setup_seg {#1} {
2344       \seq_put_right:No \l__stex_module_setup_seq \l__stex_module_setup_seg
2345     }
2346     \tl_put_right:Nx \l__stex_module_setup_ns_str {\seq_use:Nn \l__stex_module_setup_seq /}
2347   }{}{}
2348 }
2349
2350 \cs_new_protected:Npn \__stex_module_setup_split_module:n #1?#2 \__stex_module_setup_end: #3
2351   \stex_module_setup_top_nosig:n { #1 ? #2 / #3}
2352 }
2353
2354 \bool_new:N \l__stex_in_meta_bool
2355 \bool_set_false:N \l__stex_in_meta_bool
2356
2357 \cs_new_protected:Nn \__stex_module_setup_load_meta: {
2358   \tl_if_empty:NF \l__stex_metatheory_uri {
2359     \stex_execute_in_module:x{
2360       \stex_pseudogroup_with:nn{\l__stex_in_meta_bool}{%
2361         \stex_activate_module:n {\stex_uri_use:N \l__stex_metatheory_uri }%
2362       }
2363     }
2364   }
2365 }
2366
2367 <@@=stex_modules>

```

(End definition for `\stex_module_setup:n`. This function is documented on page 111.)

### `\stex_close_module:`

```

2368 \cs_new:Nn \stex_close_module: {
2369   \bool_if:NT \c_stex_persist_write_mode_bool \__stex_modules_persist_module:
2370   \stex_debug:nn{module}{
2371     Closing~module~\l__stex_current_module_str^~J
2372     Code:~\expandafter\meaning\csname c_stex_module_\l__stex_current_module_str _code\endcsna
2373     Imports:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l__stex_current_module_str
2374     Declarations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l__stex_current_module_
2375     Notations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l__stex_current_module_
2376   }
2377 }
2378
2379 \cs_new_protected:Nn \__stex_modules_persist_module: {
2380   \stex_persist:e {
2381     \__stex_modules_restore_module:nnnn {\l__stex_current_module_str}{%
2382       \exp_after:wN \prop_to_keyval:N \cs:w
2383         c_stex_module_\l__stex_current_module_str _morphisms_prop
2384     }
2385   }
2386 }

```

```

2384     \cs_end:
2385   }{
2386     \exp_after:wN \prop_to_keyval:N \cs:w
2387       c_stex_module_ \l_stex_current_module_str _symbols_prop
2388     \cs_end:
2389   }{
2390     \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
2391     \exp_after:wN \exp_after:wN \exp_after:wN
2392       { \cs:w c_stex_module_ \l_stex_current_module_str _code \cs_end: }
2393   }{}
2394   \prop_map_function:cN{c_stex_module_ \l_stex_current_module_str _notations_prop}
2395     \__stex_modules_persist_nots_i:nn
2396   \exp_not:N \STEXRestoreNotsEnd {}
2397 }
2398 }
2399
2400 \cs_new_protected:Nn \__stex_modules_restore_module:nnnn {
2401   \prop_gset_from_keyval:cn{c_stex_module_ \tl_to_str:n{\#1}_morphisms_prop}{\#2}
2402   \cs_set:Npn \__stex_modules_tl {\#3}
2403   \exp_args:Nno \prop_gset_from_keyval:cn{c_stex_module_ \tl_to_str:n{\#1}_symbols_prop}\__ste
2404   \prop_map_inline:cn{c_stex_module_ \tl_to_str:n{\#1}_symbols_prop}%
2405     \stex_ref_new_symbol:n{\#1?\#\#1}
2406   }
2407   \cs_gset:cpn{c_stex_module_ \tl_to_str:n{\#1}_code}{\#4}
2408   \prop_gclear:c{c_stex_module_ \tl_to_str:n{\#1} _notations_prop}
2409   \str_set:Nn \l__stex_modules_restore_mod_str {\#1}
2410   \group_begin:
2411     \catcode`_=8\relax
2412     \catcode`:=12\relax
2413     \__stex_modules_restore_nots:n
2414 }
2415
2416 \cs_new:Nn \__stex_modules_persist_nots_i:nn {
2417   \exp_not:n{\#2}
2418 }
2419
2420 \quark_new:N \STEXRestoreNotsEnd
2421
2422 \cs_new_protected:Nn \__stex_modules_restore_nots:n {
2423   \__stex_modules_restore_nots_i:n
2424 }
2425
2426 \cs_new_protected:Nn \__stex_modules_restore_nots_i:n {
2427   \tl_if_eq:nnTF{\#1}{\STEXRestoreNotsEnd}{%
2428     \group_end:
2429   }{
2430     \__stex_modules_restore_nots_ii:nnnn {\#1}
2431   }
2432 }
2433
2434 \cs_new_protected:Nn \__stex_modules_restore_nots_ii:nnnn {
2435   \cs_set:Npn \l__stex_modules_tl {{\#4}{\#5}}
2436   \exp_args:NN\use:nn\prop_gput:cnn{
2437     {c_stex_module_ \l__stex_modules_restore_mod_str _notations_prop}

```

```

2438     {\tl_to_str:n{#1!#2}}{
2439         {\tl_to_str:n{#1}}{\tl_to_str:n{#2}}{#3}
2440         \exp_args:No \exp_not:n \l__stex_modules_tl
2441     }
2442 }
2443 \__stex_modules_restore_nots_i:n
2444 }
```

(End definition for `\stex_close_module`. This function is documented on page 111.)

`\l_stex_metatheory_uri`

```
2445 \tl_new:N \l_stex_metatheory_uri
```

(End definition for `\l_stex_metatheory_uri`. This variable is documented on page ??.)

`\setmetatheory`

```

2446 \cs_new_protected:Nn \__stex_modules_set_matatheory:nn {
2447     \group_begin:
2448         \stex_debug:nn{metatheory}{Setting-matatheory~[#1]#2}
2449         \stex_import_module_uri:nn { #1 } { #2 }
2450         \stex_debug:nn{metatheory}{Here:^^J
2451             \l_stex_import_archive_str^^J
2452             \l_stex_import_path_str^^J
2453             \l_stex_import_name_str^^J
2454         }
2455         \stex_import_require_module:ooo
2456             \l_stex_import_archive_str
2457             \l_stex_import_path_str
2458             \l_stex_import_name_str
2459         \stex_debug:nn{metatheory}{Found:~\l_stex_import_ns_str}
2460         \exp_args:Nne \use:nn {
2461             \group_end: \stex_uri_resolve:Nn \l_stex_metatheory_uri
2462             }{\l_stex_import_ns_str}
2463     }
2464
2465 \NewDocumentCommand \setmetatheory {O{} m} {
2466     \__stex_modules_set_matatheory:nn { #1 }{ #2 }
2467     \stex_smsmode_do:
2468 }
2469 \stex_sms_allow_escape:N \setmetatheory
```

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

Keys and key handling:

```

2470 \stex_keys_define:nnnn{smodule}{
2471     \str_clear:N \l_stex_key_sig_str
2472 }{
2473     meta .code:n = {
2474         \str_if_empty:nTF {#1} {
2475             \tl_clear:N \l_stex_metatheory_uri
2476         }{
2477             \stex_uri_resolve:Nx \l_stex_metatheory_uri { #1 }
2478         }
2479     },
2480     ns .code:n = {
```

```

2481     \stex_uri_resolve:Nx \l_stex_current_ns_uri { #1 }
2482   } ,
2483   lang .code:n = {
2484     \stex_set_language:n { #1 }
2485   } ,
2486   sig .str_set_x:N = \l_stex_key_sig_str ,
2487   creators .code:n = {} , % todo ?
2488   contributors .code:n = {} , % todo ?
2489   srccite .code:n = {} % todo ?
2500 }{id, title, style, deprecate}

smodule (env.)
2501 \stex_new_stylable_env:nnnnnnn {module} {O{} m} {
2502   \stex_keys_set:nn { smodule }{ #1 }
2503   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
2504   \tl_if_empty:NF \thistitle {
2505     \exp_args:No \stexdoctitle \thistitle
2506   }
2507   \exp_args:Nx \stex_module_setup:n { \tl_to_str:n{ #2 } }
2508
2509   \stex_if_do_html:T {
2510     \exp_args:Nne \begin{stex_annotation_env} {
2511       shtml:theory={\l_stex_current_module_str},
2512       shtml:language={ \l_stex_current_language_str},
2513       shtml:signature={\l_stex_key_sig_str}
2514       \tl_if_empty:NF \l_stex_metatheory_uri {
2515         shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
2516       }
2517     }
2518     \stex_annotation_invisible:n
2519   }
2520   \stex_if_smsmode:F {
2521     \str_set_eq:NN \thismoduleuri \l_stex_current_module_str
2522     \tl_set:Nn \thismodulename {#2}
2523     \stex_style_apply:
2524   }
2525   \stex_smsmode_do:
2526 }{
2527   \stex_close_module:
2528   \stex_if_smsmode:F \stex_style_apply:
2529   \stex_if_do_html:T{ \end{stex_annotation_env} }
2530 }{}{}{s}
2531
2532 \stex_sms_allow_env:n{smodule}

```

\stex\_if\_in\_module\_p: Are we currently in a module?

\stex\_if\_in\_module:TF 2523 \prg\_new\_conditional:Nnn \stex\_if\_in\_module: {p, T, F, TF} {
2524 \str\_if\_empty:NTF \l\_stex\_current\_module\_str
2525 \prg\_return\_false: \prg\_return\_true:
2526 }

(End definition for \stex\_if\_in\_module:TF. This function is documented on page 111.)

\stex\_if\_module\_exists\_p:n Does a module with this URI exist?

\stex\_if\_module\_exists:nTF

```

2527 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
2528   \tl_if_exist:cTF { c_stex_module_#1_code }
2529   \prg_return_true: \prg_return_false:
2530 }

```

(End definition for `\stex_if_module_exists:nTF`. This function is documented on page 111.)

`\stex_do_up_to_module:n` Execute code in the current module (i.e. as if the `\begin{smodule}` was the current tex group)

```

2531 \cs_new_protected:Nn \stex_do_up_to_module:n {
2532   \exp_args:No \stex_metagroup_do_in:nn \l_stex_current_module_str {#1}
2533 }
2534 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}

```

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 112.)

`\stex_module_add_code:n`

`\stex_module_add_code:x`

```

2535 \cs_new_protected:Nn \stex_module_add_code:n {
2536   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
2537 }
2538 \cs_generate_variant:Nn \stex_module_add_code:n {x}

```

(End definition for `\stex_module_add_code:n`. This function is documented on page 112.)

`\stex_execute_in_module:n`

`\stex_execute_in_module:x`

```

2539 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:TF {
2540   \stex_module_add_code:n { #1 }
2541   \stex_do_up_to_module:n { #1 }
2542 }{ #1 }}
2543 \cs_generate_variant:Nn \stex_execute_in_module:n {x}

```

(End definition for `\stex_execute_in_module:n`. This function is documented on page 111.)

`\STEXexport`

```

2544 \NewDocumentCommand \STEXexport {} {
2545   \ExplSyntaxOn
2546   \__stex_modules_export:n
2547 }
2548 \cs_new_protected:Nn \__stex_modules_export:n {
2549   \stex_ignore_spaces_and_pars:#1\ExplSyntaxOff
2550   \stex_module_add_code:n { \stex_ignore_spaces_and_pars:#1}
2551   \stex_smsmode_do:
2552 }

```

Only allowed in modules, and allowed (escaped) in sms mode:

```

2553 \stex_deactivate_macro:Nn \STEXexport {module~environments}
2554 \stex_sms_allow_escape:N \STEXexport
2555 \stex_every_module:n {\stex_reactivate_macro:N \STEXexport}

```

(End definition for `\STEXexport`. This function is documented on page 75.)

```

\stex_module_add_morphism:nnnn
\stex_module_add_morphism:nnonn
\stex_module_add_morphism:ooox
2556 \cs_new_protected:Nn \stex_module_add_morphism:nnnn {
2557   \exp_args:Nne \prop_gput:cnn{c_stex_module_}\l_stex_current_module_str _morphisms_prop}[
2558     \tl_if_empty:nTF{#1}{[#2]}{#1}
2559   }{#1}{#2}{#3}{#4}
2560 }
2561 \cs_generate_variant:Nn \stex_module_add_morphism:nnnn {nonn,ooox}

(End definition for \stex_module_add_morphism:nnnn. This function is documented on page 119.)
```

\stex\_module\_add\_symbol:nnnnnnN #1 : {⟨Macro name⟩}  
#2 : {⟨Name⟩}  
#3 : {⟨arity⟩}  
#4 : {({⟨Arg num⟩}{⟨Arg str⟩})\*}  
#5 : Definiens  
#6 : type  
#7 : Return  
#8 : Command

```

2562 \cs_new_protected:Nn \stex_module_add_symbol:nnnnnnnnN {
2563   \stex_debug:nn{declaration}{New~declaration:~\l_stex_current_module_str?#2^~J
2564     Macro:#1^~JArity:#3~(#4)^~J
2565     Def:~\tl_to_str:n{#5}^~J
2566     Type:~\tl_to_str:n{#6}^~J
2567     Returns:~\tl_to_str:n{#7}
2568   }
2569   \%prop_gput:cnx{c_stex_module_}\l_stex_current_module_str _symbols_prop}
2570   %{#2}{\exp_not:n{#1}{#2}{#3}{#4}{#5}{#6}{#7}\exp_not:n{#8}}
2571   \prop_gput:cnn{c_stex_module_}\l_stex_current_module_str _symbols_prop}
2572   %{#2}{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}
2573   \tl_if_empty:nF{#1}{%
2574     \stex_execute_in_module:n {
2575       \__stex_modules_activate_sym:n {#2}
2576     }
2577   }
2578 }

2579 \cs_new_protected:Nn \__stex_modules_activate_sym:n {
2580   \prop_map_inline:cn{c_stex_module_}\l_stex_current_module_str _symbols_prop}[
2581     \str_if_eq:nnT{#1}{##1}{%
2582       \__stex_modules_activate_i:nnnnnnnn ##2
2583     }
2584   }
2585 }

2586 \cs_new_protected:Nn \__stex_modules_activate_i:nnnnnnnn {
2587   \stex_debug:nn{activating}{#1:\l_stex_current_module_str^~J
2588     \tl_to_str:n{#2}{#3}{#4}{#5}{#6}{#7}{#8}
2589   }
2590   \cs_set:cpx{#1} {
2591     \__stex_invoke_symbol:nnnnnnnN
2592     {\l_stex_current_module_str}
2593     \exp_not:n{#2}{#3}{#4}{#5}{#6}{#7}{#8}
2594   }
2595 }

2596 \stex_debug:nn{activating}{done}
```

```

2597     \prop_map_break:
2598 }

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

```

\stex_module_add_notation:nnnnn #1 : URI
\stex_module_add notation:eoeeoo #2 : variant
\stex_set_notation_macro:nnnnn #3 : arity
#4 : macro body
#5 : op

2599 \cs_new_protected:Nn \stex_module_add_notation:nnnnn {
2600   \stex_debug:nn{notations}{Adding~notation:^^J
2601     #1~\c_hash_str#2~#3^^J
2602     to~\l_stex_current_module_str
2603   }
2604   \prop_gput:cnn{\c_stex_module_\l_stex_current_module_str _notations_prop}
2605   {#1!#2}{[#1]{#2}{#3}{#4}{#5}}
2606   \stex_execute_in_module:n {
2607     \__stex_modules_activate_not:nn{#1}{#2}
2608   }
2609 }
2610 \cs_generate_variant:Nn \stex_module_add_notation:nnnnn {eoeeoo}
2611
2612
2613 \cs_new_protected:Nn \__stex_modules_activate_not:nn {
2614   \prop_map_inline:cn{\c_stex_module_\l_stex_current_module_str _notations_prop} {
2615     \str_if_eq:nnT{#1!#2}{[#1]}{
2616       \prop_map_break:n{\stex_set_notation_macro:nnnnn ##2 }
2617     }
2618   }
2619 }
```

(End definition for \stex\_module\_add\_notation:nnnnn and \stex\_set\_notation\_macro:nnnnn. These functions are documented on page 114.)

```

\stex_set_notation_macro:nnnnn
\stex_set_notation_macro:eoexo
2620 \cs_new_protected:Nn \stex_set_notation_macro:nnnnn {
2621   \tl_set:cn {\l_stex_notation_#1#2_cs}{#4}
2622   \cs_if_exist:cF{\l_stex_notation_#1__cs} {
2623     \tl_set:cn {\l_stex_notation_#1__cs}{#4}
2624   }
2625   \tl_if_empty:nF{#5} {
2626     \tl_set:cn{\l_stex_notation_#1_op_#2_cs}{#5}
2627     \cs_if_exist:cF{\l_stex_notation_#1_op__cs} {
2628       \cs_set_eq:cc {\l_stex_notation_#1_op__cs}{\l_stex_notation_#1_op_#2_cs}
2629     }
2630   }
2631 }
2632 \cs_generate_variant:Nn \stex_set_notation_macro:nnnnn {eoexo}
```

(End definition for \stex\_set\_notation\_macro:nnnnn. This function is documented on page 115.)

```

\stex_activate_module:n
\stex_activate_module:o
\stex_activate_module:x
2633 \cs_new_protected:Nn \stex_activate_module:n {
```

```

2634 \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
2635   \stex_debug:nn{modules}{Activating module~#1^J\expandafter\meaning\csname c_stex_module
2636   \seq_put_right:Nn \l_stex_all_modules_seq { #1 }
2637   \stex_pseudogroup:nn{
2638     \str_set:Nn \l_stex_current_module_str {#1}
2639     \use:c{ c_stex_module_#1_code }
2640   }{
2641     \stex_pseudogroup_restore:N \l_stex_current_module_str
2642   }
2643 }
2644 }
2645 \cs_generate_variant:Nn \stex_activate_module:n {o,x}

```

(End definition for `\stex_activate_module:n`. This function is documented on page 111.)

Iterating:

```
2646 <@=stex_iterate>
```

### \stex\_iterate\_symbols:n

```

2647 \cs_new_protected:Nn \stex_iterate_symbols:n {
2648   \stex_pseudogroup_with:nn{\_\stex_iterate_sym_cs:nnnnnnnnN\stex_iterate_break:\stex_iterat
2649   \cs_set:Npn \_\stex_iterate_sym_cs:nnnnnnnnN
2650   ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
2651   \cs_set:Npn \stex_iterate_break: {
2652     \prop_map_break:n{\seq_map_break:}
2653   }
2654   \cs_set:Npn \stex_iterate_break:n ##1 {
2655     \prop_map_break:n{\seq_map_break:n{##1}}
2656   }
2657   \seq_map_inline:Nn \l_stex_all_modules_seq {
2658     \prop_map_inline:cn{c_stex_module_##1_symbols_prop} {
2659       \_\stex_iterate_sym_cs:nnnnnnnnN {##1} #####2
2660     }
2661   }
2662 }
2663 }

```

(End definition for `\stex_iterate_symbols:n`. This function is documented on page 112.)

### \stex\_iterate\_symbols:nn

```

2664 \cs_new_protected:Nn \stex_iterate_symbols:nn {
2665   \seq_clear:N \l__stex_iterate_mods_seq
2666   \stex_pseudogroup_with:nn{\_\stex_iterate_sym_cs:nnnnnnnnN} {
2667     \cs_set:Npn \_\stex_iterate_sym_cs:nnnnnnnnN
2668     ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #2 }
2669     \clist_map_function:nN {#1} \_\stex_iterate_it_decl_i:n
2670   }
2671 }
2672
2673 \cs_new_protected:Nn \_\stex_iterate_it_decl_i:n {
2674   \seq_if_in:NnF \l__stex_iterate_mods_seq {#1} {
2675     \seq_put_left:Nn \l__stex_iterate_mods_seq {#1}
2676     \prop_map_inline:cn{c_stex_module_#1_morphisms_prop} {
2677       \_\stex_iterate_it_decl_check:nnnn ##2
2678     }

```

```

2679     \prop_map_inline:cn{c_stex_module_#1_symbols_prop}{
2700         \__stex_iterate_sym cs:nnnnnnnnN {#1} ##2
2701     }
2702 }
2703 }
2704 \cs_new_protected:Nn \__stex_iterate_it_decl_check:nnnn {
2705     \tl_if_empty:nT{#1}{%
2706         \__stex_iterate_it_decl_i:n {#2}
2707     }
2708 }

```

(End definition for `\stex_iterate_symbols:nn`. This function is documented on page [112](#).)

### `\stex_iterate_notations:nn`

```

2689 \cs_new_protected:Nn \stex_iterate_notations:nn {
2700     \seq_clear:N \l__stex_iterate_mods_seq
2701     \stex_pseudogroup_with:nn{\__stex_iterate_not_cs:nnnnn}{%
2702         \cs_set:Npn \__stex_iterate_not_cs:nnnnn
2703             ##1 ##2 ##3 ##4 ##5 { #2 }
2704         \clist_map_function:nN {#1} \__stex_iterate_it_not_i:n
2705     }
2706 }
2707
2708 \cs_new_protected:Nn \__stex_iterate_it_not_i:n {
2709     \seq_if_in:NnF \l__stex_iterate_mods_seq {#1} {
2710         \seq_put_left:Nn \l__stex_iterate_mods_seq {#1}
2711         \prop_map_inline:cn{c_stex_module_#1_notations_prop}{
2712             \__stex_iterate_not_cs:nnnnn ##2
2713         }
2714         \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2715             \__stex_iterate_it_not_check:nnnn ##2
2716         }
2717     }
2718 }
2719 \cs_new_protected:Nn \__stex_iterate_it_not_check:nnnn {
2720     \tl_if_empty:nT{#1}{%
2721         \__stex_iterate_it_not_i:n {#2}
2722     }
2723 }

```

(End definition for `\stex_iterate_notations:nn`. This function is documented on page [115](#).)

### `\stex_iterate_morphisms:nn`

```

2714 \cs_new_protected:Nn \stex_iterate_morphisms:nn {
2715     \seq_clear:N \l__stex_iterate_mods_seq
2716     \bool_set_true:N \l__stex_iterate_continue_bool
2717     \cs_set:Npn \__stex_iterate_morphism cs:nnnn ##1 ##2 ##3 ##4 ##5 {
2718         #2
2719         \bool_if:NT \l__stex_iterate_continue_bool {
2720             \str_if_eq:nnTF{##1}{##2}{%
2721                 \tl_put_right:Nn \l__stex_iterate_todo_tl {
2722                     \__stex_iterate_iterate_morphism:nn{##5}{##2}
2723                 }
2724             }{
2725                 \tl_put_right:Nn \l__stex_iterate_todo_tl {

```

```

2726         \__stex_iterate_iterate_morphism:nn{##5 / ##1}{##2}
2727     }
2728   }
2729 }
2730 \cs_set:Npn \stex_iterate_break:n ##1 {
2731   \bool_set_false:N \l__stex_iterate_continue_bool
2732   \prop_map_break:n{##1}
2733 }
2734 \__stex_iterate_iterate_morphism:nn{}{##1}
2735 }
2736 }
2737
2738 \cs_new_protected:Nn \__stex_iterate_iterate_morphism:nn {
2739   \tl_clear:N \l__stex_iterate_todo_tl
2740   \seq_if_in:NnF \l__stex_iterate_mods_seq {#1 #2} {
2741     \seq_put_right:Nn \l__stex_iterate_mods_seq {#1 #2}
2742     \prop_map_inline:cn{c_stex_module_#2_morphisms_prop} {
2743       \__stex_iterate_morphism_cs:nnnn ##2 {##1}
2744       % TODO
2745       % ##1: name or [mpath]
2746       % #####1 = {#####1}{#####2}{#####3}{#####4}
2747       % #####1 = name
2748       % #####2 = mpath
2749       % #####3 = type
2750       % #####4 = {origname}{newname}*
2751     }
2752     \bool_if:NT \l__stex_iterate_continue_bool \l__stex_iterate_todo_tl
2753   }
2754 }
```

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

### 13.6.2 Structural Features

```

2755 <@=stex_features>
2756 \stex_structural_feature_module:nn
2757 \stex_structural_feature_module_end:
2758 \cs_new_protected:Nn \stex_structural_feature_module:nn {
2759   \stex_if_do_html:TF {
2760     \exp_args:Nne \begin{stex_annotation_env} {
2761       \shtml:feature-#2={
2762         \l_stex_current_module_str/#1}
2763       \str_if_empty:NF \l_stex_mroname_str {,
2764         \shtml:macroname={\l_stex_mroname_str}
2765       }
2766     }
2767     \stex_annotation_invisible:n{}
2768   }\group_begin:
2769   \stex_module_setup:n {#1-module}
2770 }
2771 \cs_new_protected:Nn \stex_structural_feature_module_end: {
2772   \tl_gset_eq:NN \g_stex_last_feature_str \l_stex_current_module_str
2773   \stex_close_module:
2774   \stex_if_do_html:TF{
```

```

2774     \end{stex_annotation_env}
2775 } \group_end:
2776 }

(End definition for \stex_structural_feature_module:nn and \stex_structural_feature_module_end:. These functions are documented on page 117.)
```

```

\stex_structural_feature_morphism:nnn
\stex_structural_feature_morphism_end:
2777 \bool_new:N \l__stex_features_implicit_bool
2778 \cs_new_protected:Nn \stex_structural_feature_morphism:nnnnn {
2779   \str_clear:N \l_stex_current_domain_str
2780   \tl_if_empty:nT{\#3} {
2781     \stex_get_mathstructure:n{\#4}
2782     \str_set_eq:NN \l_stex_current_domain_str \l_stex_get_structure_module_str
2783   }
2784   \str_if_empty:NT \l_stex_current_domain_str {
2785     \stex_import_module_uri:nn { \#3 }{ \#4 }
2786     \group_begin:
2787     \stex_import_require_module:ooo
2788       \l_stex_import_archive_str
2789       \l_stex_import_path_str
2790       \l_stex_import_name_str
2791     \exp_args:Nnx \use:nn \group_end: {
2792       \str_set:Nn \exp_not:N\l_stex_current_domain_str {\l_stex_import_ns_str}
2793     }
2794   }
2795   \tl_if_empty:nTF{\#1} {
2796     \bool_set_true:N \l__stex_features_implicit_bool
2797     \str_set:Nx \l_tmpa_str {[ \l_stex_current_domain_str ] }
2798   }{
2799     \bool_set_false:N \l__stex_features_implicit_bool
2800     \str_set:Nn \l_tmpa_str {\#1}
2801   }
2802
2803 \stex_if_do_html:TF {
2804   \begin{stex_annotation_env} {
2805     \shtml:feature-\#2={\l_stex_current_module_str?\l_tmpa_str},
2806     \shtml:domain={\l_stex_current_domain_str}
2807     \#5
2808   }
2809   \stex_annotation_invisible:n{}
2810 } \group_begin:
2811 \str_set:Nn \l__stex_features_feature_str {\#2}
2812 \str_set_eq:NN \l_stex_feature_name_str \l_tmpa_str
2813 \__stex_features_setup:
2814 \__stex_features_reactivate:
2815 \%^A\stex_activate_module:o \l_stex_current_domain_str
2816 \exp_args:Ne \stex_metagroup_new:n {\l_stex_current_module_str / \l_stex_feature_name_str}
2817 }
2818
2819 \cs_new_protected:Nn \__stex_features_do_for_list: {
2820   \seq_clear:N \l_stex_fors_seq
2821   \clist_map_inline:Nn \l_stex_key_for_clist {
2822     \exp_args:Ne\stex_get_in_morphism:n{\tl_to_str:n{\#1}}
```

```

2823     \seq_put_right:Nx \l_stex_fors_seq
2824         {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
2825     }
2826 }
2827
2828 \cs_new_protected:Nn \__stex_features_add_definiens:nn {
2829     \__stex_features_set_definiens_macros: #1\__stex_features_break:
2830     \_stex_assign_do:n{#2}
2831     #2
2832 }
2833 \cs_new_protected:Npn \__stex_features_set_definiens_macros: #1?#2?#3\__stex_features_break:
2834     \str_set:Nn \l_stex_get_symbol_mod_str {#1?#2}
2835     \str_set:Nn \l_stex_get_symbol_name_str {#3}
2836     \exp_args:Nne\use:nn{\__stex_features_set_definiens_macros_i:nnnnnnn}{%
2837         \prop_item:Nn \l_stex_morphism_symbols_prop {[#1?#2]/[#3]}}
2838     }
2839 }
2840 \cs_new_protected:Nn \__stex_features_set_definiens_macros_i:nnnnnnn {
2841     \tl_set:Nn \l_stex_get_symbol_def_t1{#4}
2842 }
2843
2844 \cs_new_protected:Nn \stex_structural_feature_morphism_end: {
2845     \str_gset_eq:NN \l_stex_feature_name_str \l_stex_feature_name_str
2846     \str_gset_eq:NN \l_stex_current_domain_str \l_stex_current_domain_str
2847     \seq_gset_eq:NN \l_stex_morphism_symbols_prop \l_stex_morphism_symbols_prop
2848     \seq_gset_eq:NN \l_stex_morphism_renames_prop \l_stex_morphism_renames_prop
2849     \seq_gset_eq:NN \l_stex_morphism_morphisms_seq \l_stex_morphism_morphisms_seq
2850     \__stex_features_do_elaboration:
2851     \stex_if_do_html:TF{
2852         \end{stex_annotation_env}
2853     }\group_end:
2854 }
2855
2856 \cs_new_protected:Nn \__stex_features_setup: {
2857     \prop_clear:N \l_stex_morphism_symbols_prop
2858     \prop_clear:N \l_stex_morphism_renames_prop
2859     \seq_clear:N \l_stex_morphism_morphisms_seq
2860     \__stex_features_do_decls:
2861     \exp_args:No \__stex_features_do_morphisms:n \l_stex_current_domain_str
2862 }
2863
2864 \cs_new_protected:Nn \__stex_features_rename_all: {
2865
2866 }
2867
2868 \cs_new:Npn \__stex_features_clean:nnw [#1] / #2 \stex_end: {
2869     [#1]/[#2]
2870 }
2871
2872 \cs_new_protected:Nn \__stex_features_do_decls: {
2873     \exp_args:No \stex_iterate_symbols:nn \l_stex_current_domain_str {
2874         \stex_str_if_starts_with:nnTF{##3}[{
2875             \exp_args:NNe \prop_put:Nnn \l_stex_morphism_symbols_prop {
2876                 \__stex_features_clean:nnw ##3 \stex_end:

```

```

2877     }
2878   }{
2879     \prop_put:Nnn \l_stex_morphism_symbols_prop
2880       {##1}/[##3]}
2881   }{
2882     {##2}{##4}{##5}{##6}{##7}{##8}##9
2883   }
2884 }
2885 }
2886
2887 \cs_new_protected:Nn \stex_structural_morphism_check_total: {
2888   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2889     \__stex_features_total_check: ##1 ##2
2890   }
2891 }
2892
2893 \cs_new_protected:Npn \__stex_features_total_check: [##1]/[##2] #3 #4 #5 #6 #7 #8 #9 {
2894   \tl_if_empty:nT{##6}{%
2895     \msg_error:nnxx{stex}{error/needsdefiniens}{##1?##2}{total~morphism}}
2896   }
2897 }
2898
2899 \cs_new:Npn \__stex_features_split_qm:w #1 ? #2 ? #3 { #3 }
2900 \cs_new_protected:Nn \__stex_features_do_elaboration: {
2901   \stex_debug:nn{morphisms}{%
2902     Elaborating:^^J\prop_to_keyval:N \l_stex_morphism_symbols_prop
2903     ^^J
2904     Renamings:^^J
2905     \prop_to_keyval:N \l_stex_morphism_renames_prop
2906   }
2907   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2908     \__stex_features_elab_check: ##1 ##2
2909   }
2910   \exp_args:No\stex_iterate_notations:nn\l_stex_current_domain_str{%
2911     \prop_get:NnNTF \l_stex_morphism_renames_prop {##1}\l__stex_features_tmp {
2912       \exp_args:Ne \stex_module_add_notation:nnnnn
2913       {\l_stex_current_module_str ? \exp_after:wN \use_i:nn \l__stex_features_tmp}
2914     }{
2915       \exp_args:Ne \stex_module_add_notation:nnnnn
2916       {\l_stex_current_module_str ? \l_stex_feature_name_str
2917         / \__stex_features_split_qm:w ##1}
2918     }{##2}{##3}{##4}{##5}
2919   }
2920   \stex_module_add_morphism:ooox
2921     \l_stex_feature_name_str
2922     \l_stex_current_domain_str
2923     \l__stex_features_feature_str
2924     {\prop_map_function:NN \l_stex_morphism_renames_prop \__stex_features_rename:nn}
2925   }
2926
2927 \cs_new:Nn \__stex_features_rename:nn{
2928   {#1}{\use_i:nn#2}
2929 }
2930

```

```

2931 \cs_new_protected:Npn \__stex_features_elab_check: [#1]/[#2] #3 {
2932   \prop_get:NnNTF \l_stex_morphism_renames_prop {#1?#2} \l_stex_features_tmp {
2933     \stex_debug:nn{morphisms}{Generating~\l_stex_features_tmp}
2934     \exp_after:wN \stex_module_add_symbol:nnnnnnNN \l_stex_features_tmp
2935   }{
2936     \bool_if:NTF \l_stex_features_implicit_bool {
2937       \stex_debug:nn{morphisms}{Generating~\l_stex_feature_name_str / #2}
2938       \exp_args:Nno \stex_module_add_symbol:nnnnnnNN {#3}{\l_stex_feature_name_str / #2}
2939     }{
2940       \stex_debug:nn{morphisms}{Generating~\l_stex_feature_name_str / #2}
2941       \exp_args:Nno \stex_module_add_symbol:nnnnnnNN {}{\l_stex_feature_name_str / #2}
2942     }
2943   }
2944 }
2945
2946 \cs_new_protected:Nn \__stex_features_do_morphisms:n {
2947   \prop_map_inline:cn {c_stex_module_#1_morphisms_prop} {
2948     \__stex_features_do_morph:nnnn ##2
2949   }
2950 }
2951
2952 \cs_new_protected:Nn \__stex_features_do_morph:nnnn {
2953   \tl_if_empty:nF{#3} {
2954     \seq_put_right:Nn \l_stex_morphism_morphisms_seq {{#1}{#2}{#3}}
2955   }
2956   \__stex_features_do_morphisms:n{#2}
2957 }
2958
2959 \cs_new_protected:Npn \__stex_features_reactivate: {
2960   \stex_deactivate_macro:Nn \symdecl {module~environments}
2961   \stex_deactivate_macro:Nn \textsymdecl {module~environments}
2962   \stex_deactivate_macro:Nn \symdef {module~environments}
2963   \stex_deactivate_macro:Nn \notation {module~environments}
2964   \stex_deactivate_macro:Nn \importmodule {module~environments}
2965   \stex_deactivate_macro:Nn \requiremodule {module~environments}
2966   \stex_deactivate_macro:Nn \smodule {outside~of~morphisms}
2967   \stex_reactivate_macro:N \assign
2968   \stex_reactivate_macro:N \assignMorphism
2969   \stex_reactivate_macro:N \renamedecl
2970   \cs_set_eq:NN \stex_do_for_list: \__stex_features_do_for_list:
2971   \cs_set_eq:NN \stex_add_definiens:nn \__stex_features_add_definiens:nn
2972 }

```

(End definition for `\stex_structural_feature_morphism:nnn` and `\stex_structural_feature_morphism_end::`. These functions are documented on page ??.)

### `\stex_get_in_morphism:n`

```

2973 \cs_new_protected:Nn \stex_get_in_morphism:n {
2974   \str_clear:N \l_stex_get_symbol_name_str
2975   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2976     \exp_args:Nx\__stex_features_get_check:nnnn{\tl_to_str:n{#1}}##1##2
2977   }
2978   \str_if_empty:NT \l_stex_get_symbol_name_str {
2979     \prop_map_inline:Nn \l_stex_morphism_renames_prop {

```

```

2980     \__stex_features_renamed_check:nnnnn{#1}##1=##2
2981 }
2982 \str_if_empty:NT \l_stex_get_symbol_name_str {
2983     \msg_error:nxxx{stex}{error/unknownsymbolin}{#1}{
2984         morphism~\l_stex_feature_name_str
2985     }
2986 }
2987 }
2988 }
2989
2990 \cs_new_protected:Npn \__stex_features_renamed_check:nnnnn #1#2?#3?#4=#5#6 {
2991     \str_if_eq:nnTF{#1}{#5} {
2992         \exp_args:Nnx \use:nn{\__stex_features_check_break:nnnnnnnnn{#2?#3}{#4}}{
2993             \prop_item:Nn \l_stex_morphism_symbols_prop {[#2?#3]/[#4]}
2994         }
2995     }{
2996         \str_if_eq:nnT{#1}{#6} {
2997             \exp_args:Nnx \use:nn{\__stex_features_check_break:nnnnnnnnn{#2?#3}{#4}}{
2998                 \prop_item:Nn \l_stex_morphism_symbols_prop {[#2?#3]/[#4]}
2999             }
3000         }
3001     }
3002 }
3003
3004 \cs_new_protected:Npn \__stex_features_get_check:nnnn #1[#2]/[#3]#4 {
3005     \str_if_eq:nnTF{#1}{#3} {
3006         \__stex_features_check_break:nnnnnnnnn{#2}{#3}{#4}
3007     }{
3008         \str_if_eq:nnTF{#1}{#4} {
3009             \__stex_features_check_break:nnnnnnnnn{#2}{#3}{#4}
3010         }{
3011             \use_none:nnnnn
3012         }
3013     }
3014 }
3015
3016 \cs_new_protected:Nn \__stex_features_check_break:nnnnnnnnn {
3017     \prop_map_break:n{
3018         \str_set:Nn \l_stex_get_symbol_mod_str{#1}
3019         \str_set:Nn \l_stex_get_symbol_name_str{#2}
3020         \str_set:Nn \l_stex_get_symbol_macro_str{#3}
3021         \int_set:Nn \l_stex_get_symbol_arity_int {#4}
3022         \tl_set:Nn \l_stex_get_symbol_args_tl {#5}
3023         \tl_set:Nn \l_stex_get_symbol_def_tl {#6}
3024         \tl_set:Nn \l_stex_get_symbol_type_tl {#7}
3025         \tl_set:Nn \l_stex_get_symbol_return_tl {#8}
3026         \tl_set:Nn \l_stex_get_symbol_invoke_cs {#9}
3027     }
3028 }

```

(End definition for `\stex_get_in_morphism:n`. This function is documented on page 120.)

## 13.7 Inheritance

### 13.7.1 \importmodule/\usemodule

```
3029 <@@=stex_importmodule>

usemodule

3030 \stex_new_stylable_cmd:nnnn {usemodule} { 0{} m } {
3031   \stex_import_module_uri:nn { #1 }{ #2 }
3032   \stex_import_require_module:ooo
3033     \l_stex_import_archive_str
3034     \l_stex_import_path_str
3035     \l_stex_import_name_str
3036   \stex_if_do_html:T {
3037     \hbox{\stex_annotate_invisible:nn
3038       {shtml:usemodule=\l_stex_import_ns_str} {}}
3039   }
3040   \stex_if_smsmode:F{
3041     \group_begin:
3042     \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3043     \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3044     \tl_clear:N \thisstyle
3045     \stex_style_apply:
3046     \group_end:
3047   }
3048 }{}
```

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

\stex\_import\_module\_uri:nn

```
3049 \cs_new_protected:Nn \stex_import_module_uri:nn {
3050   \stex_debug:nn{importmodule}{URI:~>#1<~#2<}
3051   \exp_args:NNnx \seq_set_split:Nnn \__stex_importmodule_seq ? { \tl_to_str:n{ #2 } }
3052   \seq_pop_right:NN \__stex_importmodule_seq \l_stex_import_name_str
3053   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \__stex_importmodule_seq ? }
3054   \tl_if_empty:nTF { #1 } {
3055     \stex_debug:nn{importmodule}{No~archive}
3056     \prop_if_exist:NTF \l_stex_current_archive_prop {
3057       \stex_debug:nn{importmodule}{Picking~current~archive}
3058       \str_set:Nx \l_stex_import_archive_str {
3059         \prop_item:Nn \l_stex_current_archive_prop { id }
3060       }
3061     }{
3062       \str_clear:N \l_stex_import_archive_str
3063       \str_set:Nn \l_stex_import_uri_str {file:}
3064       \str_if_empty:NTF \l_stex_import_path_str {
3065         \stex_debug:nn{importmodule}{Empty~Path}
3066         \stex_file_split_off_ext:NN \l__stex_importmodule_path_seq \g_stex_current_file
3067         \stex_file_split_off_lang:NN \l__stex_importmodule_path_seq \l__stex_importmodule_pa
3068         \str_set:Nx \l_stex_import_path_str {
3069           \stex_file_use:N \l__stex_importmodule_path_seq
3070         }
3071       }{
3072         \stex_debug:nn{importmodule}{Resolving~path~\l_stex_import_path_str~relative~to~\ste
3073         \stex_file_resolve:Nx \l__stex_importmodule_seq { \stex_file_use:N \g_stex_current_f
```

```

3074     \str_set:Nx \l_stex_import_path_str {
3075         \stex_file_use:N \l_stex_importmodule_seq
3076     }
3077     \stex_debug:nn{importmodule}{...yields~\l_stex_import_path_str}
3078 }
3079 }
3080 } {
3081     \stex_debug:nn{importmodule}{Archive~#1}
3082     \str_set:Nx \l_stex_import_archive_str { #1 }
3083     \stex_require_archive:o \l_stex_import_archive_str
3084     \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ } \l_stex_import_archive_str }
3085 }
3086 }
```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 119.)

```

\stex_import_require_module:nnn
\stex_import_require_module:ooo
3087 \cs_new_protected:Npn \stex_import_require_module:nnn #1 {
3088     \tl_if_empty:nTF { #1 } {
3089         \str_clear:N \l_stex_importmodule_archive_str
3090         \str_set:Nn \l_stex_import_uri_str {file:{}}
3091         \__stex_importmodule_get_module:nnn {}
3092     }{
3093         \stex_require_archive:n { #1 }
3094         \str_set:Nx \l_stex_importmodule_archive_str {#1}
3095         \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ } #1 _manifest_prop}{}
3096         \str_set:Nx \l_stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / source }
3097         \exp_args:No \__stex_importmodule_get_module:nnn \l_stex_importmodule_str
3098     }
3099 }
3100 \cs_generate_variant:Nn \stex_import_require_module:nnn {ooo}
3101
3102 \cs_new_protected:Npn \stex_import_require_module_safe:nnn #1 {
3103     \tl_if_empty:nTF { #1 } {
3104         \str_clear:N \l_stex_importmodule_archive_str
3105         \str_set:Nn \l_stex_import_uri_str {file:{}}
3106         \__stex_importmodule_get_module_safe:nnn {}
3107     }{
3108         \stex_require_archive:n { #1 }
3109         \str_set:Nx \l_stex_importmodule_archive_str {#1}
3110         \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ } #1 _manifest_prop}{}
3111         \str_set:Nx \l_stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / source }
3112         \exp_args:No \__stex_importmodule_get_module_safe:nnn \l_stex_importmodule_str
3113     }
3114 }
3115
3116 \cs_new_protected:Nn \__stex_importmodule_get_module_uri:nnn {
3117     \tl_if_empty:nF {#2} {
3118         \str_set:Nx \l_stex_import_uri_str { \l_stex_import_uri_str / #2 }
3119     }
3120     \stex_debug:nn{importmodule}{~>#1<^J>#2<^J>#3<^J>\l_stex_import_uri_str<^J>
3121     Current~file:\stex_file_use:N \g_stex_current_file<^J>
3122     Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3123 }
```

```

3124 \stex_if_module_exists:nTF {\l_stex_import_uri_str?#3} {
3125   \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3126 }{
3127   \stex_if_module_exists:nTF{\stex_uri_use:N \l_stex_current_ns_uri ? #3} {
3128     \str_set:Nx \l_stex_import_ns_str {\stex_uri_use:N \l_stex_current_ns_uri ? #3}
3129 }{
3130   \__stex_importmodule_get_from_file:nnn{#1}{#2}{#3}
3131   \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3132 }
3133 }
3134 }
3135 \cs_new_protected:Nn \__stex_importmodule_get_module_uri_safe:nnn {
3136   \tl_if_empty:nF {#2} {
3137     \str_set:Nx \l_stex_import_uri_str {\l_stex_import_uri_str / #2}
3138   }
3139   \stex_debug:nn{importmodule}{~>#1<^J>#2<^J>#3<^J>\l_stex_import_uri_str<^J>
3140     Current~file:\stex_file_use:N \g_stex_current_file^J
3141     Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3142   }
3143   \stex_if_module_exists:nTF {\l_stex_import_uri_str?#3} {
3144     \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3145 }{
3146   \stex_if_module_exists:nTF{\stex_uri_use:N \l_stex_current_ns_uri ? #3} {
3147     \str_set:Nx \l_stex_import_ns_str {\stex_uri_use:N \l_stex_current_ns_uri ? #3}
3148 }{
3149   \__stex_importmodule_get_from_file_safe:nnn{#1}{#2}{#3}
3150   \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3151 }
3152 }
3153 }
3154
3155 \cs_new_protected:Nn \__stex_importmodule_get_module:nnn {
3156   \stex_debug:nn{importmodule}{Requiring~>[#1]#2?#3<}
3157   \__stex_importmodule_get_module_uri:nnn{#1}{#2}{#3}
3158   \stex_activate_module:o \l_stex_import_ns_str
3159 }
3160
3161 \cs_new_protected:Nn \__stex_importmodule_get_module_safe:nnn {
3162   \stex_debug:nn{importmodule}{Requiring~>[#1]#2?#3<}
3163   \__stex_importmodule_get_module_uri_safe:nnn{#1}{#2}{#3}
3164 }
3165
3166 \cs_new_protected:Nn \__stex_importmodule_get_from_file:nnn {
3167   \stex_file_resolve:Nx \l__stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3168   \str_set:Nx \l__stex_importmodule_str {\stex_file_use:N \l__stex_importmodule_seq}
3169   \stex_debug:nn{imports}{Looking~for~\l_stex_import_uri_str?#3...}
3170   \__stex_importmodule_check_file:nn{ /#3.tex }{
3171     \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
3172       \__stex_importmodule_check_file:nn{/#3.en.tex}{
3173         \__stex_importmodule_check_file:nn{.tex}{
3174           \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex}{
3175             \__stex_importmodule_check_file:nn{.en.tex}{
3176               \msg_error:nnx{stex}{error/unknownmodule}{\l_stex_import_uri_str?#3}
3177             }
3178           }
3179         }
3180       }
3181     }
3182   }
3183 }
```

```

3178         }
3179     }
3180   }
3181 }
3182 \stex_if_smsmode:TF{
3183   \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
3184     \stex_debug:nn{imports}{Skipping-current-file}
3185   }{
3186     \__stex_importmodule_load_file:n{#3}
3187   }
3188 }{
3189   \__stex_importmodule_load_file:n{#3}
3190 }
3191 }
3192 }
3193 \cs_new_protected:Nn \__stex_importmodule_get_from_file_safe:nnn {
3194   \stex_file_resolve:Nx \l__stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3195   \str_set:Nx \l__stex_importmodule_str {\stex_file_use:N \l__stex_importmodule_seq}
3196   \stex_debug:nn{imports}{Looking-for-\l_stex_import_uri_str?#3...}
3197   \__stex_importmodule_check_file:nn{ /#3.tex }{
3198     \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
3199       \__stex_importmodule_check_file:nn{/#3.en.tex}(
3200         \__stex_importmodule_check_file:nn{.tex}(
3201           \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex}(
3202             \__stex_importmodule_check_file:nn{.en.tex}(
3203               )
3204             )
3205           )
3206         )
3207       )
3208     }
3209   \stex_if_smsmode:TF{
3210     \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
3211       \stex_debug:nn{imports}{Skipping-current-file}
3212     }{
3213       \IfFileExists{ \l__stex_importmodule_str }{
3214         \__stex_importmodule_load_file:n{#3}
3215       }{}
3216     }
3217   }{
3218     \IfFileExists{ \l__stex_importmodule_str }{
3219       \__stex_importmodule_load_file:n{#3}
3220     }{}
3221   }
3222 }
3223 \cs_new_protected:Nn \__stex_importmodule_load_file:n {
3224   \stex_file_in_smsmode:on \l__stex_importmodule_str {
3225     \str_if_empty:NF \l__stex_importmodule_archive_str {
3226       \stex_set_current_archive:n \l__stex_importmodule_archive_str
3227     }
3228     \stex_debug:nn{modules}{Loading-\l__stex_importmodule_str}
3229   }
3230 \stex_if_module_exists:nF {\l_stex_import_uri_str?#1} {

```

```

3232     \msg_error:nnn{stex}{error/unknownmodule}{\l_stex_import_uri_str?#1}
3233 }
3234 }
3235
3236 \cs_new_protected:Npn \__stex_importmodule_check_file:nn #1 {
3237     \stex_debug:nn{imports}{Checking~ \l_stex_importmodule_str #1}
3238     \IfFileExists{ \l_stex_importmodule_str #1 }{
3239         \stex_debug:n{imports}{Success}
3240         \str_set:Nx \l_stex_importmodule_str { \l_stex_importmodule_str #1 }
3241     }
3242 }

```

(End definition for `\stex_import_module:nnn`. This function is documented on page [119](#).)

### \importmodule

```

3243 \stex_new_stylable_cmd:nnnn{importmodule} { O{} m } {
3244     \__stex_importmodule_import_module:nn {#1}{#2}
3245     \stex_smsmode_do:
3246 }
3247 \stex_deactivate_macro:Nn \importmodule {module~environments}
3248
3249 \cs_new_protected:Nn \__stex_importmodule_import_module:nn {
3250     \stex_import_module_uri:nn { #1 }{ #2 }
3251     \stex_import_require_module:ooo
3252         \l_stex_import_archive_str
3253         \l_stex_import_path_str
3254         \l_stex_import_name_str
3255     \stex_execute_in_module:x{
3256         \stex_activate_module:n{\l_stex_import_ns_str}
3257     }
3258     \stex_module_add_morphism:nonn
3259         {}{\l_stex_import_ns_str}{import}(){}
3260     \stex_if_do_html:T {
3261         \stex_annotation_invisible:nn
3262             {shtml:import=\l_stex_import_ns_str} {}
3263     }
3264     \stex_if_smsmode:F{
3265         \group_begin:
3266         \tl_set:Nn \thisarchive {#1}
3267         \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3268         \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3269         \tl_clear:N \thisstyle
3270         \stex_style_apply:
3271         \group_end:
3272     }
3273 }
3274
3275 \cs_new_protected:Nn \__stex_importmodule_import_module_presms:nn {
3276     \stex_import_module_uri:nn { #1 }{ #2 }
3277     \tl_gput_right:Nx \g_stex_sms_import_code {
3278         \stex_import_require_module_safe:nnn
3279             {\l_stex_import_archive_str}
3280             {\l_stex_import_path_str}
3281             {\l_stex_import_name_str}

```

```

3282     }
3283 }
3284
3285 \stex_sms_allow_escape:N \importmodule
3286 \stex_every_module:n {\stex_reactivate_macro:N \importmodule}
3287 \stex_sms_allow_import:Nn \importmodule {
3288   \stex_reactivate_macro:N \importmodule
3289   \let \_stex_importmodule_import_module:nn \__stex_importmodule_import_module_presms:nn
3290 }
3291
3292 \stex_new_styable_cmd:nnnn{requiremodule} { 0{} m } {
3293   \stex_import_module_uri:nn { #1 }{ #2 }
3294   \stex_import_require_module:ooo
3295     \l_stex_import_archive_str
3296     \l_stex_import_path_str
3297     \l_stex_import_name_str
3298   \stex_do_up_to_module:x{
3299     \stex_activate_module:n{\l_stex_import_ns_str}
3300   }
3301   \stex_if_do_html:T {
3302     \stex_annotation_invisible:nn
3303     {shtml:import=\l_stex_import_ns_str} {}
3304   }
3305   \stex_if_smsmode:F{
3306     \group_begin:
3307     \tl_set:Nn \thisarchive {#1}
3308     \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3309     \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3310     \tl_clear:N \thisstyle
3311     \stex_style_apply:
3312     \group_end:
3313   }
3314   \stex_smsmode_do:
3315 }
3316 \stex_deactivate_macro:Nn \requiremodule {module-environments}

```

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

### 13.7.2 Theory Morphisms

```

3317 <@=stex_morphisms>
3318 \stex_new_styable_cmd:nnnn {assign} { m m }{
3319   \stex_get_in_morphism:n{#1}
3320   \stex_assign_do:n{#2}
3321   \stex_smsmode_do:
3322 }
3323 \stex_sms_allow_escape:N\assign
3324
3325 \cs_new_protected:Nn \stex_assign_do:n{
3326   \stex_debug:nn{assign}{Assigning~\l_stex_get_symbol_name_str~to~\tl_to_str:n{#1}}
3327   \tl_if_empty:NF \l_stex_get_symbol_def_tl {
3328     \%msg_error:nnxx{stex}{error/symbolalreadydefined}{\l_stex_get_symbol_name_str}{
3329       % morphism~\l_stex_feature_name_str
3330       %
3331     }

```

```

3331 }
3332 \stex_check_term:n{#1}
3333 \stex_debug:nn{HERE!}{
3334   \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str^~J
3335   \tl_to_str:n{#1}
3336 }
3337 \stex_if_do_html:T{
3338   \stex_annotation_invisible:nn{shtml:assign={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_
3339     \l_stex_annotation_force_break:n{
3340       \mode_if_math:T\hbox{\$}\stex_annotation:nn{shtml:definiens={}{}{#1}{\$}}
3341     }
3342   }
3343 }
3344 \exp_args:Ne \stex_metagroup_do_in:nx{
3345   \l_stex_current_module_str / \l_stex_feature_name_str
3346 }{
3347   \prop_put:Nnn \exp_not:N \l_stex_morphism_symbols_prop
3348   {[ \l_stex_get_symbol_mod_str ]/[ \l_stex_get_symbol_name_str ]}
3349   {
3350     {\l_stex_get_symbol_macro_str}
3351     {\int_use:N \l_stex_get_symbol_arity_int}
3352     {\l_stex_get_symbol_args_tl}
3353     {\exp_not:n{#1}}
3354     {\exp_args:No\exp_not:n\l_stex_get_symbol_type_tl}
3355     {\exp_args:No\exp_not:n\l_stex_get_symbol_return_tl}
3356     {\l_stex_get_symbol_invoke_cs}
3357   }
3358 }
3359 }
3360
3361
3362 \stex_new_stylable_cmd:nnnn {renamedecl} { m 0{} m }{
3363   \stex_get_in_morphism:n{#1}
3364   \l_stex_renamedecl_do:nn{#2}{#3}
3365   \stex_smsmode_do:
3366 }{}
3367 \stex_sms_allow_escape:N\renamedecl
3368
3369 \cs_new_protected:Nn \l_stex_renamedecl_do:nn {
3370   \stex_debug:nn{renamedecl}{Renaming` \l_stex_get_symbol_name_str ~to~ [#1]{#2}}
3371   \stex_if_do_html:T{
3372     \exp_args:Ne \stex_annotation_invisible:nn{
3373       shtml:rename={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str},
3374       shtml:macroname={#2}
3375       \str_if_empty:nF{#1}{ ,shtml:to={#1} }
3376     }{}
3377   }
3378   \exp_args:Ne \stex_metagroup_do_in:nx{
3379     \l_stex_current_module_str / \l_stex_feature_name_str
3380   }{
3381     \prop_put:Nnn \exp_not:N \l_stex_morphism_renames_prop
3382     {[ \l_stex_get_symbol_mod_str ]/[ \l_stex_get_symbol_name_str ]{}{#2}{{
3383       \tl_if_empty:nTF{#1}{\l_stex_feature_name_str/\l_stex_get_symbol_name_str}{#1}
3384     }}}

```

```

3385     }
3386 }
3387
3388 \stex_new_stylable_cmd:nnnn {assignMorphism} { m m }{
3389   \str_clear:N \l__stex_morphisms_morphism_dom_str
3390   \stex_iterate_morphisms:nn\l_stex_current_domain_str{
3391     \stex_debug:nn{assignMorphism}{}
3392     Checking:~#1 vs:^^J##1^^J##2^^J##3^^J##4
3393   }
3394   \str_if_eq:nnTF{#1}{##1}{%
3395     \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3396   }{%
3397     \stex_str_if_ends_with:nnT{##2}{#1}{%
3398       \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3399     }
3400   }
3401 }
3402 \str_if_empty:NT \l__stex_morphisms_morphism_dom_str {
3403   \msg_error:nnn{stex}{error/nomorphism}{#1}
3404 }
3405 \bool_set_false:N \l_tmpa_bool
3406 \stex_iterate_morphisms:nn \l_stex_current_module_str {
3407   \stex_debug:nn{assignMorphism}{}
3408   Checking:~#2 vs:^^J##1^^J##2^^J##3^^J##4
3409 }
3410 \str_if_eq:nnTF{#2}{##1}{%
3411   \stex_debug:nn{assignMorphism}{match!}
3412   \stex_iterate_break:n{%
3413     \stex_annotation_invisible:nn{%
3414       \shtml:assignMorphismFrom={\l__stex_morphisms_morphism_dom_str}
3415       \ahtml:assignMorphismTo={\l_stex_current_module_str?##1}
3416     }{}%
3417     \bool_set_true:N \l_tmpa_bool
3418   }
3419 }{%
3420   \stex_str_if_ends_with:nnT{##2}{#2}{%
3421     \stex_debug:nn{assignMorphism}{match!}
3422     \stex_iterate_break:n{%
3423       \stex_annotation_invisible:nn{%
3424         \shtml:assignMorphismFrom={\l__stex_morphisms_morphism_dom_str},
3425         \shtml:assignMorphismTo={\l_stex_current_module_str?##1}
3426       }{}%
3427       \bool_set_true:N \l_tmpa_bool
3428     }
3429   }
3430 }
3431 }
3432 \bool_if:NF \l_tmpa_bool {
3433   \msg_error:nnn{stex}{error/nomorphism}{#2}
3434 }
3435 }{}%
3436 \cs_new_protected:Nn \__stex_morphisms_do_morph_assign:nnn {
3437   \stex_iterate_break:n{%
3438     \str_set:Nx \l__stex_morphisms_morphism_dom_str { \l_stex_current_domain_str ? #1 }

```

```

3439   \stex_debug:nn{assignMorphism}{match!}
3440   \stex_iterate_symbols:nn{#2}{
3441     \stex_debug:nn{assignMorphism}{removing~##1?##3}
3442     % TODO: non-trivial assignments
3443     \prop_remove:Nn \l_stex_morphism_symbols_prop {
3444       [##1]/[##3]
3445     }
3446   }
3447 }
3448 }

3449 \stex_deactivate_macro:Nn \assign {morphism~environments}
3450 \stex_deactivate_macro:Nn \renamedecl {morphism~environments}
3451 \stex_deactivate_macro:Nn \assignMorphism {morphism~environments}
3452 \stex_new_stylable_env:nnnnnnn {copymodule}{m 0{} m}{
3453
3454   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3455
3456   \stex_if_smsmode:F {
3457     \tl_set:Nn \thiscopyname { #1 }
3458     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3459     \stex_style_apply:
3460   }
3461
3462   \stex_smsmode_do:
3463 }{
3464   \stex_if_smsmode:F {
3465     \stex_style_apply:
3466   }
3467   \stex_structural_feature_morphism_end:
3468 }{}{}{}

3469 \stex_deactivate_macro:Nn \copymodule {module~environments}
3470 \stex_every_module:n {
3471   \stex_reactivate_macro:N \copymodule
3472 }
3473 \stex_sms_allow_env:n{copymodule}

3474 \stex_new_stylable_env:nnnnnnn {interpretmodule}{m 0{} m}{
3475   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3476   \stex_if_smsmode:F {
3477     \tl_set:Nn \thiscopyname { #1 }
3478     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3479     \stex_style_apply:
3480   }
3481
3482   \stex_smsmode_do:
3483 }{
3484   \stex_structural_feature_morphism_check_total:
3485   \stex_if_smsmode:F {
3486     \stex_style_apply:
3487   }
3488   \stex_structural_feature_morphism_end:
3489 }{}{}{}

3490 \stex_deactivate_macro:Nn \interpretmodule {module~environments}
3491 \stex_every_module:n {
3492   \stex_reactivate_macro:N \interpretmodule

```

```

3493 }
3494 \stex_sms_allow_env:n{interpretmodule}
3495 \stex_new_stylable_env:nnnnnnn {realization}{0{} m}{
3496
3497   \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}
3498   \%stex_execute_in_module:x{
3499     % \stex_activate_module:n{\l_stex_current_domain_str}
3500   %}
3501   \stex_if_smsmode:F {
3502     \tl_set:Nn \thiscopyname { #2 }
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 \realization {module~environments}
3515 \stex_every_module:n {
3516   \stex_reactivate_macro:N \realization
3517 }
3518 \stex_sms_allow_env:n{realization}

3519 \cs_new_protected:Nn \__stex_morphisms_parse_assign:n {
3520   \str_clear:N \l_stex_morphisms_name_str
3521   \str_clear:N \l_stex_morphisms_newname_str
3522   \tl_clear:N \l_stex_morphisms_ass_tl
3523   \exp_args:NNN \seq_set_split:Nnn \l_stex_morphisms_seq {\tl_to_str:n{@}} {#1}
3524   \int_compare:nNnTF {\seq_count:N \l_stex_morphisms_seq} = 1 {
3525     \seq_pop_left:NN \l_stex_morphisms_seq \l_stex_morphisms_next_tl
3526   }{
3527     \seq_pop_left:NN \l_stex_morphisms_seq \l_stex_morphisms_name_str
3528     \exp_args:NNN \str_set:Nn \l_stex_morphisms_name_str \l_stex_morphisms_name_str
3529     \tl_set:Nx \l_stex_morphisms_next_tl {\seq_use:Nn \l_stex_morphisms_seq @}
3530   }
3531   \exp_args:NNN \seq_set_split:Nnn \l_stex_morphisms_seq = \l_stex_morphisms_next_tl
3532   \str_if_empty:NTF \l_stex_morphisms_name_str {
3533     \seq_pop_left:NN \l_stex_morphisms_seq \l_stex_morphisms_name_str
3534     \exp_args:NNN \str_set:Nn \l_stex_morphisms_name_str \l_stex_morphisms_name_str
3535     \tl_set:Nx \l_stex_morphisms_ass_tl {\seq_use:Nn \l_stex_morphisms_seq =}
3536   }{
3537     \seq_pop_left:NN \l_stex_morphisms_seq \l_stex_morphisms_newname_str
3538     \exp_args:NNN \str_set:Nn \l_stex_morphisms_newname_str \l_stex_morphisms_newname_str
3539     \tl_set:Nx \l_stex_morphisms_ass_tl {\seq_use:Nn \l_stex_morphisms_seq =}
3540   }
3541   \__stex_morphisms_do_parsed_assign:
3542 }
3543
3544 \cs_new_protected:Nn \__stex_morphisms_do_parsed_assign: {
3545   \exp_args:No \stex_get_in_morphism:n \l_stex_morphisms_name_str

```

```

3546 \str_if_empty:NF \l__stex_morphisms_newname_str {
3547   \exp_after:wN \__stex_morphisms_do_parsed_newname: \l__stex_morphisms_newname_str \__ste
3548 }
3549 \tl_if_empty:NF \l__stex_morphisms_ass_tl {
3550   \exp_args:No \_stex_assign_do:n \l__stex_morphisms_ass_tl
3551 }
3552 }
3553
3554 \cs_new_protected:Nn \__stex_morphisms_do_parsed_newname: {
3555   \peek_charcode:NTF [ {
3556     \__stex_morphisms_do_parsed_newname:w
3557   }{
3558     \__stex_morphisms_do_parsed_newname:w []
3559   }
3560 }
3561
3562 \cs_new_protected:Npn \__stex_morphisms_do_parsed_newname:w [#1] #2 \__stex_morphisms_end: {
3563   \_stex_renamedecl_do:nn{#1}{#2}
3564 }
3565
3566 \stex_new_stylable_cmd:nnnn{copymod}{m O{} m m}{
3567   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3568
3569 \clist_map_function:nN{#4}\__stex_morphisms_parse_assign:n
3570
3571 \stex_if_smsmode:F {
3572   \tl_set:Nn \thiscopyname { #1 }
3573   \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3574   \stex_style_apply:
3575 }
3576 \stex_structural_feature_morphism_end:
3577 \stex_smsmode_do:
3578 }{}
3579 \stex_deactivate_macro:Nn \copymod {module~environments}
3580 \stex_every_module:n {
3581   \stex_reactivate_macro:N \copymod
3582 }
3583 \stex_sms_allow_escape:N\copymod
3584
3585
3586 \stex_new_stylable_cmd:nnnn{interpretmod}{m O{} m m}{
3587   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3588
3589 \clist_map_function:nN{#4}\__stex_morphisms_parse_assign:n
3590
3591 \stex_if_smsmode:F {
3592   \tl_set:Nn \thiscopyname { #1 }
3593   \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3594   \stex_style_apply:
3595 }
3596 \stex_structural_feature_morphism_check_total:
3597 \stex_structural_feature_morphism_end:
3598 \stex_smsmode_do:
3599 }{}

```

```

3600 \stex_deactivate_macro:Nn \interpretmod {module~environments}
3601 \stex_every_module:n {
3602   \stex_reactivate_macro:N \interpretmod
3603 }
3604 \stex_sms_allow_escape:N\interpretmod
3605
3606
3607 \stex_new_stylable_cmd:nnnn{realize}{0}{m m}{
3608   \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}
3609
3610 \clist_map_function:nN{#3}\_stex_morphisms_parse_assign:n
3611
3612 \stex_if_smsmode:F {
3613   \tl_set:Nn \thiscopyname { #1 }
3614   \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3615   \stex_style_apply:
3616 }
3617 \stex_structural_feature_morphism_check_total:
3618 \stex_structural_feature_morphism_end:
3619 \stex_smsmode_do:
3620 }{}
3621 \stex_deactivate_macro:Nn \realize {module~environments}
3622 \stex_every_module:n {
3623   \stex_reactivate_macro:N \realize
3624 }
3625 \stex_sms_allow_escape:N\realize

```

## 13.8 Symbols

### 13.8.1 Declarations

```
3626 <@=stex_symdecl>
```

Some setup:

```
\stex_if_check_terms_p:
\stex_if_check_terms:TF
3627 \stex_if_html_backend:TF {
3628   \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3629     \prg_return_false:
3630   }
3631 }{
3632   \stex_get_env:Nn\__stex_symdecl_env_str{STEX_CHECKTERMS}
3633   \str_if_empty:NF\__stex_symdecl_env_str{
3634     \exp_args:No \str_if_eq:nnF \__stex_symdecl_env_str{false} {
3635       \bool_set_true:N \c_stex_check_terms_bool
3636     }
3637   }
3638   \bool_if:NTF \c_stex_check_terms_bool {
3639     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3640       \prg_return_true:
3641     }
3642   }{
3643     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3644       \prg_return_false:
3645     }
3646 }
```

```
3646     }
3647 }
```

(End definition for `\stex_if_check_terms:TF`. This function is documented on page 113.)

### `\stex_check_term:n`

```
3648 \stex_if_check_terms:TF{
3649   \cs_new_protected:Nn \stex_check_term:n {
3650     \hbox_set:Nn \l_tmpa_box {
3651       \group_begin:
3652         $#1$%
3653       \group_end:
3654     }
3655   }
3656 }{
3657   \cs_new_protected:Nn \stex_check_term:n {}
3658 }
```

(End definition for `\stex_check_term:n`. This function is documented on page 113.)

symdecl arguments:

```
3659 \stex_keys_define:nnnn{symargs}{
3660   \str_clear:N \l_stex_key_args_str
3661   \str_clear:N \l_stex_key_role_str
3662   \str_clear:N \l_stex_key_reorder_str
3663   \str_clear:N \l_stex_key_assoc_str
3664 }{
3665   args      .str_set:N = \l_stex_key_args_str ,
3666   reorder   .str_set:N = \l_stex_key_reorder_str ,
3667   assoc     .choices:nn  = {bin,binl,binr,pre,conj,pwconj}
3668   {\str_set:Nx \l_stex_key_assoc_str \l_keys_choice_tl},
3669   role      .str_set:N = \l_stex_key_role_str
3670 }{}
3671
3672 \stex_keys_define:nnnn{decl}{
3673   \str_clear:N \l_stex_key_name_str
3674   \str_clear:N \l_stex_key_args_str
3675   \tl_clear:N \l_stex_key_type_tl
3676   \tl_clear:N \l_stex_key_def_tl
3677   \tl_clear:N \l_stex_key_return_tl
3678   \clist_clear:N \l_stex_key_argtypes_clist
3679 }{
3680   name      .str_set:N = \l_stex_key_name_str ,
3681
3682   return    .tl_set:N = \l_stex_key_return_tl ,
3683   argtypes  .clist_set:N = \l_stex_key_argtypes_clist ,
3684
3685   type      .tl_set:N = \l_stex_key_type_tl ,
3686   def       .tl_set:N = \l_stex_key_def_tl ,
3687
3688   align     .code:n   = {},
3689   gfc      .code:n   = {}
3690 }{style,deprecate,symargs}
3691 % \stex_do_deprecation:n{#2}
```

```

\symdecl

3692 \str_new:N \l_stex_mroname_str
3693 \stex_new_styable_cmd:nnnn {\symdecl} { s m O{} } {
3694   \stex_keys_set:nn{decl}{#3}
3695   \IfBooleanTF #1 {
3696     \str_clear:N \l_stex_mroname_str
3697   }{
3698     \str_set:Nx \l_stex_mroname_str { #2 }
3699   }
3700 \stex_symdecl_top:n{#2}

3701 \stex_if_smsmode:F{
3702   \group_begin:
3703   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3704   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3705   \tl_set_eq:NN \thistype \l_stex_key_type_tl
3706   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
3707   \tl_set_eq:NN \thisargs \l_stex_key_args_str
3708   \tl_clear:N \thisstyle
3709   \stex_style_apply:
3710   \group_end:
3711 }
3712 \stex_smsmode_do:
3713 }{}
3714 \stex_deactivate_macro:Nn \symdecl {module~environments}
3715 \stex_every_module:n {\stex_reactivate_macro:N \symdecl}
3716 \stex_sms_allow_escape:N \symdecl

```

(End definition for `\symdecl`. This function is documented on page 76.)

```

\stex_symdecl_top:n

3718 \cs_new_protected:Nn \stex_symdecl_top:n {
3719   \str_if_empty:NT \l_stex_key_name_str {
3720     \str_set:Nx \l_stex_key_name_str { #1 }
3721   }
3722   \stex_symdecl_do:
3723   \stex_symdecl_check_terms:
3724   \stex_symdecl_add_decl:
3725   \stex_if_do_html:T {
3726     \stex_symdecl_html:
3727   }
3728 }

3729 \cs_new_protected:Nn \stex_symdecl_add_decl: {
3730   \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnnN} {
3731     {\l_stex_mroname_str}
3732     {\l_stex_key_name_str}
3733     {\int_use:N \l_stex_get_symbol_arity_int}
3734     {\l_stex_get_symbol_args_tl}
3735     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }%{\exp_args:No \exp_not:n \l_stex_key_def_tl
3736     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
3737     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
3738     \stex_invoke_symbol:
3739   }
3740 }
```

```

3741     \exp_args:Nn \stex_ref_new_symbol:n
3742         {\l_stex_current_module_str?\l_stex_key_name_str}
3743     }
3744
3745     \cs_new:Nn \stex_return_args:nn {
3746         {\svar{ARGUMENT_#1}\stex_eat_exclamation_point:}
3747     }
3748
3749     \cs_new_protected:Nn \stex_symdecl_html: {
3750         \exp_args:Nn \stex_annotation_invisible:nn {
3751             shtml:symdecl = {\l_stex_current_module_str ? \l_stex_key_name_str},
3752             shtml:args = {\l_stex_key_args_str}
3753             \str_if_empty:NF \l_stex_macroname_str {
3754                 shtml:macroname={\l_stex_macroname_str}
3755             }
3756             \str_if_empty:NF \l_stex_key_assoc_str {
3757                 shtml:assocotype={\l_stex_key_assoc_str}
3758             }
3759             \str_if_empty:NF \l_stex_key_reorder_str {
3760                 shtml:reorderargs={\l_stex_key_reorder_str}
3761             }
3762             \str_if_empty:NF \l_stex_key_role_str {
3763                 shtml:role={\l_stex_key_role_str}
3764             }
3765         }{\hbox\bgroup\stex_annotation_force_break:n{
3766             \bool_set_true:N \stex_in_invisible_html_bool
3767             \tl_if_empty:NF \l_stex_key_type_tl {
3768                 $\stex_annotation:nn{shtml:type={}}{\l_stex_key_type_tl}$
3769             }
3770             \tl_if_empty:NF \l_stex_key_def_tl {
3771                 $\stex_annotation:nn{shtml:definiens={}}{\l_stex_key_def_tl}$
3772             }
3773             \tl_if_empty:NF \l_stex_key_return_tl{
3774                 \exp_args:Nno \use:n{
3775                     \cs_generate_from_arg_count:NNnn \l_stex_symdecl_cs
3776                     \cs_set:Npn \l_stex_get_symbol_arity_int \l_stex_key_return_tl
3777                     \tl_set:Nx \l_stex_symdecl_args_tl {\l_stex_map_args:N \stex_return_args:nn}
3778                     $\stex_annotation:nn{shtml:returntype={}}{
3779                         \exp_after:wN \l_stex_symdecl_cs \l_stex_symdecl_args_tl!
3780                     }$}
3781             }
3782             \clist_if_empty:NF \l_stex_key_argtypes_clist {
3783                 \stex_annotation:nn{shtml:argtypes={}}{\l_stex_annotation_force_break:n{
3784                     \clist_map_inline:Nn \l_stex_key_argtypes_clist {
3785                         $\stex_annotation:nn{shtml:type={}}{\##1}$$
3786                     }
3787                 }}
3788             }
3789         }\egroup}
3790     }

```

(End definition for `\stex_symdecl_top:n`. This function is documented on page [114](#).)

`\stex_symdecl_do:` Requires the above keys and `\l_stex_macroname_str` to be set first

```

3791 \cs_new_protected:Nn \stex_symdecl_do: {
3792   \stex_do_deprecation:n \l_stex_key_name_str
3793   \__stex_symdecl_parse_arity:
3794   \__stex_symdecl_do_args:
3795 }
3796
3797 \int_new:N \l_stex_assoc_args_count
3798
3799 \cs_new_protected:Nn \__stex_symdecl_parse_arity: {
3800   \int_zero:N \l_stex_get_symbol_arity_int
3801   \int_zero:N \l_stex_assoc_args_count
3802   \str_map_inline:Nn \l_stex_key_args_str {
3803     \str_case:nnF ##1 {
3804       0 { \str_map_break: }
3805       1 { \str_map_break:n{
3806         \int_set:Nn \l_stex_get_symbol_arity_int {1}
3807         \str_set:Nn \l_stex_key_args_str {i}
3808       } }
3809       2 { \str_map_break:n{
3810         \int_set:Nn \l_stex_get_symbol_arity_int {2}
3811         \str_set:Nn \l_stex_key_args_str {ii}
3812       } }
3813       3 { \str_map_break:n{
3814         \int_set:Nn \l_stex_get_symbol_arity_int {3}
3815         \str_set:Nn \l_stex_key_args_str {iii}
3816       } }
3817       4 { \str_map_break:n{
3818         \int_set:Nn \l_stex_get_symbol_arity_int {4}
3819         \str_set:Nn \l_stex_key_args_str {iiii}
3820       } }
3821       5 { \str_map_break:n{
3822         \int_set:Nn \l_stex_get_symbol_arity_int {5}
3823         \str_set:Nn \l_stex_key_args_str {iiiii}
3824       } }
3825       6 { \str_map_break:n{
3826         \int_set:Nn \l_stex_get_symbol_arity_int {6}
3827         \str_set:Nn \l_stex_key_args_str {iiiiii}
3828       } }
3829       7 { \str_map_break:n{
3830         \int_set:Nn \l_stex_get_symbol_arity_int {7}
3831         \str_set:Nn \l_stex_key_args_str {iiiiiii}
3832       } }
3833       8 { \str_map_break:n{
3834         \int_set:Nn \l_stex_get_symbol_arity_int {8}
3835         \str_set:Nn \l_stex_key_args_str {iiiiiiii}
3836       } }
3837       9 { \str_map_break:n{
3838         \int_set:Nn \l_stex_get_symbol_arity_int {9}
3839         \str_set:Nn \l_stex_key_args_str {iiiiiiiii}
3840       } }
3841       i {\int_incr:N \l_stex_get_symbol_arity_int}
3842       b {\int_incr:N \l_stex_get_symbol_arity_int}
3843       a {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3844       B {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}

```

```

3845     }{
3846         \msg_error:n{stex}{error/wrongargs}{
3847             \l_stex_current_module_str ? \l_stex_key_name_str
3848             }{##1}
3849         }
3850     }
3851 }
3852
3853 \cs_new_protected:Nn \__stex_symdecl_do_args: {
3854     \tl_clear:N \l_stex_get_symbol_args_tl
3855     \int_step_inline:nn \l_stex_get_symbol_arity_int {
3856         \tl_put_right:Nn \l_stex_get_symbol_args_tl {##1}
3857         \tl_put_right:Nx \l_stex_get_symbol_args_tl {
3858             \str_item:Nn \l_stex_key_args_str {##1}
3859         }
3860     }
3861 }

```

(End definition for `\stex_symdecl_do:`. This function is documented on page 113.)

### `\_stex_symdecl_check_terms:`

```

3862 \cs_new_protected:Nn \_stex_symdecl_check_terms: {
3863     \stex_check_term:n{
3864         \stex_debug:nn{check_terms}{Checking~type...}
3865         \group_begin:\l_stex_key_type_tl\group_end:
3866         \stex_debug:nn{check_terms}{Checking~definiens...}
3867         \group_begin:\l_stex_key_def_tl\group_end:
3868         \stex_debug:nn{check_terms}{Checking~return...}
3869         \group_begin:\l_stex_key_return_tl!\group_end:
3870         \stex_debug:nn{check_terms}{Checking~argument~types...}
3871         \group_begin:\l_stex_key_argtypes_clist\group_end:
3872     }
3873 }

```

(End definition for `\_stex_symdecl_check_terms:`. This function is documented on page 114.)

### `\textsymdecl`

```

3874
3875 \stex_keys_define:nnnn{textsymdecl}{
3876     \str_clear:N \l_stex_key_name_str
3877     \tl_clear:N \l_stex_key_type_tl
3878     \tl_clear:N \l_stex_key_def_tl
3879 }{
3880     name      .str_set:N = \l_stex_key_name_str ,
3881     type      .tl_set:N   = \l_stex_key_type_tl ,
3882     def       .tl_set:N   = \l_stex_key_def_tl
3883 }{style,deprecate}
3884
3885 \stex_new_stylable_cmd:nnnn {textsymdecl} {m O{} m} {
3886     \stex_keys_set:nn{symdef}{}
3887     \stex_keys_set:nn{textsymdecl}{#2}
3888     \str_set:Nx \l_stex_macroname_str { #1 }
3889     \str_if_empty:NT \l_stex_key_name_str {
3890         \str_set:Nn \l_stex_key_name_str {#1}
3891     }%
3892 }

```

```

3892 % \str_set:Nx \l_stex_key_name_str {\l_stex_key_name_str-sym}
3893 %}
3894 \str_set:Nn \l_stex_key_role_str {textsymdecl}

3895
3896 \stex_symdecl_do:
3897 \_stex_symdecl_check_terms:
3898 \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN} {
3899   {\l_stex_mroname_str}
3900   {\l_stex_key_name_str}
3901   {0}{}
3902   {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
3903   {}% type
3904   {\use:c{\#1name_nospace}}% return
3905   \stex_invoke_text_symbol:
3906 }
3907 \exp_args:Ne \stex_ref_new_symbol:n
3908   {\l_stex_current_module_str?\l_stex_key_name_str}
3909 \stex_if_do_html:T {
3910   \_stex_symdecl_html:
3911 }

3912
3913 \int_set:Nn \l_stex_get_symbol_arity_int 0
3914 \tl_clear:N \l_stex_key_op_tl
3915 \str_clear:N \l_stex_key_intent_str
3916 \str_clear:N \l_stex_key_prec_str
3917 \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
3918 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
3919 \stex_notation_parse:n{\hbox{#3}}
3920 \_stex_notation_add:
3921 \stex_if_do_html:T {
3922   \def\comp{\_comp}
3923   \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
3924 }
3925 \stex_execute_in_module:x{
3926   \_stex_symdecl_set_textsymdecl_macro:nnn{#1}{\l_stex_current_module_str?\l_stex_key_name_str}
3927   \exp_not:n{#3}
3928 }

3929
3930 \stex_if_smsmode:F{
3931   \group_begin:
3932   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3933   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3934   \tl_clear:N \thisstyle
3935   \stex_style_apply:
3936   \group_end:
3937 }
3938 \stex_smsmode_do:
3939 }{}

3940 \stex_deactivate_macro:Nn \textsymdecl {module-environments}
3941 \stex_every_module:n {\stex_reactivate_macro:N \textsymdecl}
3942 \stex_sms_allow_escape:N \textsymdecl

3943
3944 \cs_new_protected:Nn \_stex_symdecl_set_textsymdecl_macro:nnn {
3945   \cs_set_protected:cpn{\#1name_nospace}{#3}

```

```

3946   \cs_set_protected:cpn{\#1name}{
3947     \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3948     \mode_if_math:T\hbox{\let\xspace\relax #3}
3949     \mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3950   }
3951 }
3952
3953 \cs_new_protected:Nn \stex_invoke_text_symbol: {
3954   \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3955   \stex_term_oms_or_omv:nnn{}{\maincomp{\let\xspace\relax\l_stex_current_return_tl}}
3956   \group_end:\mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3957 }

```

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

### `\stex_get_symbol:n`

```

3958 \cs_new_protected:Nn \stex_get_symbol:n {
3959   \stex_get_symbol:n{ #1 }
3960   \str_if_empty:NT \l_stex_get_symbol_name_str {
3961     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3962   }
3963 }
3964
3965 \cs_new_protected:Nn \stex_get_symbol:n {
3966   \str_clear:N \l_stex_get_symbol_mod_str
3967   \str_clear:N \l_stex_get_symbol_name_str
3968   \cs_if_exist:cTF { #1 }{
3969     \cs_set_eq:Nc \l_stex_symdecl_cs { #1 }
3970     % command name
3971     \exp_args:Ne \tl_if_empty:nTF { \cs_argument_spec:N \l_stex_symdecl_cs }{
3972       % ...that takes no arguments
3973       \exp_args:Ne \cs_if_eq:NNTF {\tl_head:N \l_stex_symdecl_cs}
3974         \stex_invoke_symbol:nnnnnnnn
3975         \stex_symdecl_get_symbol_from_cs:
3976         {\stex_symdecl_get_symbol_from_string:n { #1 }}
3977     }{
3978       \stex_symdecl_get_symbol_from_string:n { #1 }
3979     }
3980   }{
3981     \stex_symdecl_get_symbol_from_string:n { #1 }
3982   }
3983 }
3984
3985 \int_new:N \l_stex_get_symbol_arity_int
3986 \cs_new_protected:Nn \stex_symdecl_get_symbol_from_cs: {
3987   \stex_debug:nn{symbols}{Getting~from~cs...}
3988   \stex_pseudogroup_with:nn{\stex_invoke_symbol:nnnnnnnn}{%
3989     \cs_set:Npn \stex_invoke_symbol:nnnnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {
3990       \str_set:Nn \l_stex_get_symbol_mod_str {##1}
3991       \str_set:Nn \l_stex_get_symbol_name_str {##2}
3992       \int_set:Nn \l_stex_get_symbol_arity_int {##3}
3993       \tl_set:Nn \l_stex_get_symbol_args_tl {##4}
3994       \tl_set:Nn \l_stex_get_symbol_def_tl {##5}
3995       \tl_set:Nn \l_stex_get_symbol_type_tl {##6}

```

```

3996     \tl_set:Nn \l_stex_get_symbol_return_tl {##7}
3997     \tl_set:Nn \l_stex_get_symbol_invoke_cs {##8}
3998 }
3999 \l_stex_symdecl_cs
4000 }
4001 }
4002
4003 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
4004     \stex_debug:nn{symbols}{Getting~from~string~#1...}
4005     \seq_set_split:Nnn \l_stex_symdecl_seq ? {#1}
4006     \seq_pop_right:NN \l_stex_symdecl_seq \l_stex_symdecl_name
4007     \seq_if_empty:NTF \l_stex_symdecl_seq {
4008         \exp_args:No \__stex_symdecl_get_from_one_string:n {#1}
4009     }{
4010         \exp_args:NNe \exp_args:Nno \__stex_symdecl_get_symbol_from_modules:nn {
4011             \seq_use:Nn \l_stex_symdecl_seq ?
4012         } \l_stex_symdecl_name
4013     }
4014 }
4015
4016 \cs_new_protected:Nn \__stex_symdecl_sym_from_str_i:nnnn {
4017     \bool_lazy_any:nTF{
4018         {\str_if_eq_p:nn{#2}{#3}}
4019         {\str_if_eq_p:nn{#2}{#4}}
4020         {\str_if_ends_with_p:nn{#4}{/#2}}
4021     }{
4022         \__stex_symdecl_sym_i_finish:nnnnnnN{#1}{#4}
4023     }{
4024         \__stex_symdecl_sym_i_gobble:nnnnnn
4025     }
4026 }
4027 \cs_new_protected:Nn \__stex_symdecl_sym_i_gobble:nnnnnn {}
4028
4029 \cs_new_protected:Nn \__stex_symdecl_sym_i_finish:nnnnnnN {
4030     \prop_map_break:n{\seq_map_break:n{
4031         \str_set:Nn \l_stex_get_symbol_mod_str {#1}
4032         \str_set:Nn \l_stex_get_symbol_name_str {#2}
4033         \int_set:Nn \l_stex_get_symbol_arity_int {#3}
4034         \tl_set:Nn \l_stex_get_symbol_args_tl {#4}
4035         \tl_set:Nn \l_stex_get_symbol_def_tl {#5}
4036         \tl_set:Nn \l_stex_get_symbol_type_tl {#6}
4037         \tl_set:Nn \l_stex_get_symbol_return_tl {#7}
4038         \tl_set:Nn \l_stex_get_symbol_invoke_cs {#8}
4039     }}
4040 }
4041
4042 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_modules:nn {
4043     \stex_debug:nn{symbols}{Getting~#2~in~#1...}
4044     \seq_map_inline:Nn \l_stex_all_modules_seq {
4045         \str_if_ends_with:nnT{##1}{#1} {
4046             \prop_map_inline:cN{c_stex_module_##1_symbols_prop} {
4047                 \__stex_symdecl_sym_from_str_i:nnnn{##1}{#2} ####2
4048             }
4049     }

```

```

4050     }
4051 }
4052
4053 \cs_new_protected:Nn \__stex_symdecl_get_from_one_string:n {
4054   \stex_debug:nn{symbols}{Getting~#1~anywhere...}
4055   \stex_iterate_symbols:n{
4056     \%stex_debug:nn{symbols}{>#1==##2~|~#1==##3<...}
4057     \bool_lazy_any:nT{
4058       {\str_if_eq_p:nn{#1}{##2}}
4059       {\str_if_eq_p:nn{#1}{##3}}
4060       {\stex_str_if_ends_with_p:nn{##3}{/#1}}
4061     }{
4062       \stex_iterate_break:n{
4063         \str_set:Nn \l_stex_get_symbol_mod_str {##1}
4064         \str_set:Nn \l_stex_get_symbol_name_str {##3}
4065         \int_set:Nn \l_stex_get_symbol_arity_int {##4}
4066         \tl_set:Nn \l_stex_get_symbol_args_tl {##5}
4067         \tl_set:Nn \l_stex_get_symbol_def_tl {##6}
4068         \tl_set:Nn \l_stex_get_symbol_type_tl {##7}
4069         \tl_set:Nn \l_stex_get_symbol_return_tl {##8}
4070         \tl_set:Nn \l_stex_get_symbol_invoke_cs {##9}
4071       }
4072     }
4073   }
4074 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 113.)

### 13.8.2 Notations

```

4075 <@=stex_notations>
4076 \_stex_map_args:N
\stex_map_notation_args:N
4077 \cs_new:Nn \_stex_map_args:N {
4078   \tl_if_empty:NF \l_stex_get_symbol_args_tl {
4079     \exp_after:wN \__stex_notations_map_args_i:w \exp_after:wN
4080     #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
4081   }
4082 \cs_new:Npn \__stex_notations_map_args_i:w #1 #2 #3 #4 \__stex_notations_args_end: {
4083   #1 #2 #3
4084   \tl_if_empty:nF{#4}{
4085     \__stex_notations_map_args_i:w #1 #4 \__stex_notations_args_end:
4086   }
4087 }
4088
4089 \cs_new:Nn \_stex_map_notation_args:N {
4090   \tl_if_empty:NF \l_stex_notation_args_tl {
4091     \exp_after:wN \__stex_notations_map_args_ii:w \exp_after:wN
4092     #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
4093   }
4094 }
4095 \cs_new:Npn \__stex_notations_map_args_ii:w #1 #2 #3 #4 #5 #6 \__stex_notations_args_end: {
4096   #1 #2 #3 #4 #5
4097   \tl_if_empty:nF{#6}{
```

```

4098     \__stex_notations_map_args_ii:w #1 #6 \__stex_notations_args_end:
4099   }
4100 }

```

(End definition for `\_stex_map_args:N` and `\_stex_map_notation_args:N`. These functions are documented on page ??.)

notation arguments:

```

4101 \stex_keys_define:nnnn{notation}{
4102   \str_clear:N \l_stex_key_variant_str
4103   \str_clear:N \l_stex_key_prec_str
4104   \str_clear:N \l_stex_key_op_tl
4105   \str_clear:N \l_stex_key_intent_str
4106   \clist_clear:N \l_stex_key_intent_args_clist
4107 }{
4108   variant    .str_set_x:N = \l_stex_key_variant_str ,
4109   prec      .str_set_x:N = \l_stex_key_prec_str ,
4110   op        .tl_set:N     = \l_stex_key_op_tl ,
4111   intent     .str_set:N = \l_stex_key_intent_str ,
4112   argnames   .clist_set:N = \l_stex_key_intent_args_clist ,
4113   unknown    .code:n     = {
4114     \str_set_eq:NN \l_stex_key_variant_str \l_keys_key_str
4115   }
4116 }{style}

```

### \notation

```

4117 \stex_new_stylable_cmd:nnnn {notation} { s m 0{} m} {
4118   \stex_keys_set:nn{notation}{#3}
4119   \stex_get_symbol:n{#2}
4120   \stex_notation_parse:n{#4}
4121   \stex_if_check_terms:T{ \_stex_notation_check: }
4122   \_stex_notation_add:
4123   \stex_if_do_html:T {
4124     \def\comp{\_comp}
4125     \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4126   }
4127   \IfBooleanTF#1{
4128     \_stex_notation_set_default:n{
4129       \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4130     }
4131   }{}
4132   \stex_if_smsmode:F{
4133     \group_begin:
4134     \__stex_notations_styledefs:
4135     \stex_style_apply:
4136     \group_end:
4137   }
4138   \stex_smsmode_do:
4139 }{}
4140
4141 \cs_new_protected:Nn \__stex_notations_styledefs: {
4142   \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4143   \str_set:Nn \thisdeclname \l_stex_get_symbol_name_str
4144   \tl_set:Nx \thisdecluri {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4145   \def\thisnotation{

```

```

4146   $
4147   \tl_set_eq:NN \l_stex_current_symbol_str\thisdecluri
4148   \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{
4149     \l_stex_notation_make_args:
4150   }$}
4151 }
4152 }
4153
4154 \stex_deactivate_macro:Nn \notation {module~environments}
4155 \stex_every_module:n {\stex_reactivate_macro:N \notation}
4156 \stex_sms_allow_escape:N \notation

```

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

`\stex_notation_parse:n` requires the above keys, `\l_stex_get_symbol_arity_int`, and `\l_stex_get_symbol_args_tl`

```

4157 \cs_new_protected:Nn \stex_notation_parse:n {
4158   \tl_if_empty:NF \l_stex_key_op_tl {
4159     \tl_set:Nx \l_stex_key_op_tl { \exp_not:N\maincomp {
4160       \exp_args:No \exp_not:n \l_stex_key_op_tl
4161     } }
4162   }
4163   \seq_clear:N \l__stex_notations_precs_seq
4164   \tl_clear:N \l_stex_notation_args_tl
4165   \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0 {
4166     \__stex_notations_const_precs:
4167     \tl_if_empty:NT \l_stex_key_op_tl {
4168       \tl_set:Nn \l_stex_key_op_tl { \maincomp{#1} }
4169     }
4170   }{
4171     \__stex_notations_fun_precs:
4172     \str_set:Nn \l__stex_notations_missing_str {#1}
4173     \tl_clear:N \l__stex_notations_missing_tl
4174     \stex_map_args:N \__stex_notations_add_missing_args:nn
4175     \tl_if_empty:NT \l_stex_key_op_tl {
4176       \hbox_set:Nn \l_tmpa_box {
4177         \str_set:Nn \l_stex_current_symbol_str {}
4178         \cs_set:Npn \l_tmpa cs ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
4179         \cs_set:Npn \maincomp ##1 {
4180           \tl_gset:Nn \l_stex_key_op_tl { \maincomp{##1} }
4181           ##1
4182         }
4183         \cs_set:Npn \argsep ##1 ##2 {##1 ##2}
4184         \cs_set:Npn \argmap ##1 ##2 ##3 {##1 ##3}
4185         \cs_set:Npn \argarraymap ##1 ##2 ##3 ##4 {
4186           ##1 ##2
4187         }
4188         \stex_suppress_html:n{$\l_tmpa cs abcdefghj$}
4189       }
4190     }
4191   }
4192   \exp_args:NNe
4193   \tl_set:Nn \l_stex_notation_macrocode_cs {
4194     \STEXInternalNotation

```

```

4195 { \l_stex_key_variant_str }
4196 { \l_stex_notations_opprec_tl }
4197 { \l_stex_key_intent_str }
4198 { \l_stex_notation_args_tl }
4199 {
4200     \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0
4201     { \exp_not:n { \maincomp{ #1 } } }
4202     { \exp_not:n { #1 } \l_stex_notations_missing_tl }
4203 }
4204 }
4205 \stex_debug:nn{notation}{Notation:\meaning\l_stex_notation_macrocode_cs}
4206 }
4207
4208 \cs_new_protected:Nn \__stex_notations_const_precs: {
4209     \str_if_empty:NTF \l_stex_key_prec_str {
4210         \tl_set:No \l_stex_notations_opprec_tl { \neginfpref }
4211     }{
4212         \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4213             \tl_set:No \l_stex_notations_opprec_tl { \neginfpref }
4214         }{
4215             \tl_set_eq:NN \l_stex_notations_opprec_tl \l_stex_key_prec_str
4216         }
4217     }
4218 }
4219
4220 \cs_new_protected:Nn \__stex_notations_fun_precs: {
4221     \str_if_empty:NTF \l_stex_key_prec_str {
4222         \tl_set:No \l_stex_notations_opprec_tl { \neginfpref }
4223     }{
4224         \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4225             \tl_set:No \l_stex_notations_opprec_tl { \neginfpref }
4226         }{
4227             \tl_set_eq:NN \l_stex_notations_opprec_tl \l_stex_key_prec_str
4228         }
4229     }
4230     \str_if_empty:NTF \l_stex_key_prec_str {
4231         \tl_set:Nn \l_stex_notations_opprec_tl { 0 }
4232         \int_step_inline:nn \l_stex_get_symbol_arity_int {
4233             \seq_put_right:Nn \l_stex_notations_precs_seq {0}
4234         }
4235     }{
4236         \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4237             \stex_debug:nn{notation}{No~brackets}
4238             \tl_set:No \l_stex_notations_opprec_tl { \neginfpref }
4239             \int_step_inline:nn \l_stex_get_symbol_arity_int {
4240                 \exp_args:NNo \seq_put_right:Nn \l_stex_notations_precs_seq \infpref
4241             }
4242             } \__stex_notations_parse_precs:
4243         }
4244     \__stex_notations_do_argnames:
4245 }
4246
4247 \cs_new_protected:Nn \__stex_notations_parse_precs: {
4248     \stex_debug:nn{notation}{parsing~precedence~\l_stex_key_prec_str}

```

```

4249 \seq_set_split:NnV \l__stex_notations_seq ; \l_stex_key_prec_str
4250 \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4251   \tl_set_eq:NN \l__stex_notations_oppref_tl \l__stex_notations_str
4252   \seq_pop_left:NNT \l__stex_notations_seq \l__stex_notations_str {
4253     \exp_args:NNo \seq_set_split:NnV \l__stex_notations_seq
4254       {\tl_to_str:n{x}} \l__stex_notations_str
4255   }
4256 }{
4257   \tl_set:No \l__stex_notations_oppref_tl { 0 }
4258 }
4259 \int_step_inline:nn \l_stex_get_symbol_arity_int {
4260   \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4261     \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_str
4262   }{
4263     \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_oppref_tl
4264   }
4265 }
4266 }
4267
4268 \cs_new_protected:Nn \__stex_notations_do_argnames: {
4269   \tl_clear:N \l_stex_notation_args_tl
4270   \stex_map_args:N \__stex_notations_do_argname:nn
4271 }
4272
4273 \cs_new_protected:Nn \__stex_notations_do_argname:nn {
4274   \clist_if_empty:NTF \l_stex_key_intent_args_clist {
4275     \tl_put_right:Nx \l_stex_notation_args_tl {
4276       #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1} {
4277         \str_if_empty:NF \l_stex_key_intent_str {#1}
4278       }
4279     }
4280   }{
4281     \tl_put_right:Nx \l_stex_notation_args_tl {
4282       #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}
4283       {\c_dollar_str\clist_item:Nn \l_stex_key_intent_args_clist 1}
4284     }
4285     \clist_pop:NN \l_stex_key_intent_args_clist \l_tmpa_tl
4286   }
4287 }
4288
4289 \cs_new:Nn \__stex_notations_add_missing_args:nn {
4290   \exp_args:NNe \str_if_in:NnF \l__stex_notations_missing_str {\c_hash_str\c_hash_str#1} {
4291     \tl_put_right:Nn \l__stex_notations_missing_tlf{\STEXinvis{## #1}}
4292   }
4293 }

```

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

```

\stex_notation_check:
  \stex_notation_add:
    \cs_new_protected:Nn \stex_notation_check: {
      \stex_check_term:n{
        \str_set:Nn \l_stex_current_symbol_str {test}
        \cs_set:Npn \comp ##1 {##1}
        \stex_debug:nn{check_terms}{Checking~notation...}

```

```

4299   \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}{
4300     \_stex_notation_make_args:
4301   }
4302 }
4303 }
4304
4305 \cs_new_protected:Nn \_stex_notation_add: {
4306   \stex_module_add_notation:eoecoc{
4307     \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4308     } \l_stex_key_variant_str
4309     {\int_use:N \l_stex_get_symbol_arity_int}
4310     \l_stex_notation_macrocode_cs
4311     \l_stex_key_op_tl
4312 }
4313
4314 \cs_new_protected:Nn \_stex_notation_do_html:n {
4315   \hbox{\stex_annotation_invisible:nn {
4316     \shtml:notation={#1},
4317     \shtml:notationfragment={\l_stex_key_variant_str},
4318     \shtml:precedence={\l_stex_notations_oppref_tl},
4319     \shtml:argprecs={\seq_use:Nn \l_stex_notations_precs_seq ,}
4320   }{
4321     \cs_set_protected:Npn \argsep ##1 ##2 {
4322       \stex_annotation:nn{\shtml:argsep={}}{
4323         ##1 ##2
4324       }
4325     }
4326     \cs_set_protected:Npn \argmap ##1 ##2 ##3 {
4327       \cs_set:Npn \l_stex_notations_map_cs: #####1 { ##2 }
4328       \stex_annotation:nn{\shtml:argmap={}}{
4329         \l_stex_notations_map_cs:{##1} ##3
4330       }
4331     }
4332     \cs_set_protected:Npn \maincomp {
4333       \do_comp:nNn {\maincomp}\compemph@uri
4334     }
4335   $ 
4336     \str_set:Nx \l_stex_current_symbol_str {#1}
4337     \stex_annotation:nn{\shtml:notationcomp={}}{
4338       \exp_args:Nne \use:nn {
4339         \l_stex_notation_macrocode_cs {}
4340       }{
4341         \l_stex_map_args:N \l_stex_notations_make_arg_html:nn
4342       }
4343     }
4344   $ 
4345   \tl_if_empty:NF \l_stex_key_op_tl {
4346     $
4347     \str_set:Nx \l_stex_current_symbol_str {#1}
4348     \stex_annotation:nn{\shtml:notationopcomp={}}{
4349       \l_stex_term_oms:nnn{\l_stex_key_variant_str}{}{\l_stex_key_op_tl}
4350     }
4351   $
4352 }

```

```

4353     }
4354 }
4355
4356 \cs_new:Nn \__stex_notations_make_arg_html:nn {
4357 % \str_case:nnF #2 {
4358 %   a {{
4359 %     \stex_annotation:n{html:argnum=#1a}{x},
4360 %     \stex_annotation:n{html:argnum=#1b}{x}
4361 %   }}
4362 %   B {{
4363 %     \stex_annotation:n{html:argnum=#1a}{x},
4364 %     \stex_annotation:n{html:argnum=#1b}{x}
4365 %   }}
4366 % }{
4367 {
4368   \stex_annotation:n{html:argnum=#1}{x}
4369 }
4370 %
4371 }
4372
4373 \cs_new:Nn \_stex_notation_make_args: {
4374   \stex_map_notation_args:N \__stex_notations_make_arg:nnnn
4375 }
4376
4377
4378 \cs_new:Nn \__stex_notations_make_arg:nnnn {
4379 % \str_case:nnF #2 {
4380   a {{
4381     a\c_math_subscript_token{#1,1},
4382     a\c_math_subscript_token{#1,2}
4383   }}
4384   B {{
4385     B\c_math_subscript_token{#1,1},
4386     B\c_math_subscript_token{#1,2}
4387   }}
4388 }{
4389   \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}
4390   {{#2}\c_math_subscript_token{#1}}
4391 }
4392 }

```

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

### `\setnotation`

```

\_stex_notation_set_default:n
4393 \cs_new_protected:Npn \setnotation #1 #2 {
4394   \stex_get_symbol:n{#1}
4395   \cs_if_exist:cTF{l_stex_notation_
4396     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4397     _#2_cs
4398 }{
4399   \tl_set_eq:Nc \l_stex_notation_macrocode_cs {l_stex_notation_
4400     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4401     _#2_cs
4402 }

```

```

4403 \cs_if_exist:cTF{l_stex_notation_
4404     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4405     _op_#2_cs
4406 }{
4407     \tl_set_eq:Nc \l_stex_key_op_tl {l_stex_notation_
4408         \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4409         _op_#2_cs
4410     }
4411 }{
4412     \tl_clear:N \l_stex_key_op_tl
4413 }
4414 \_stex_notation_set_default:n{
4415     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4416 }
4417 }{
4418     \msg_error:nnxx{stex}{unknownnotation}{#2}{
4419         \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4420     }
4421 }
4422 }
4423
4424 \cs_new_protected:Nn \_stex_notation_set_default:n{
4425     \stex_module_add_notation:eoeloo{#1}{}
4426     {\int_use:N \l_stex_get_symbol_arity_int}
4427     \l_stex_notation_macrocode_cs
4428     \l_stex_key_op_tl
4429 }

```

(End definition for `\setnotation` and `\_stex_notation_set_default:n`. These functions are documented on page 80.)

### \varnotation

```

4430 \stex_new_stylable_cmd:nnnn {varnotation} { s m 0{} m} {
4431     \stex_keys_set:nn{notation}{#3}
4432     \stex_get_var:n{#2}
4433     \str_set_eq:NN \l_stex_key_name_str \l_stex_get_symbol_name_str
4434     \stex_notation_parse:n{#4}
4435     \stex_if_check_terms:T{ \_stex_notation_check: }
4436     \_stex_vardecl_notation_macro:
4437     \IfBooleanTF{#1{
4438         \_stex_notation_set_default:n{\l_stex_get_symbol_name_str}
4439     }{}}
4440     \group_begin:
4441     \tl_set_eq:NN \thisvarname \l_stex_get_symbol_name_str
4442     \tl_clear:N \thisstyle
4443     \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4444     \def\thisnotation{
4445         $ \let\l_stex_current_symbol_str\thisvarname
4446         \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}
4447         \_stex_notation_make_args:
4448     }$}
4449 }
4450 \stex_style_apply:
4451 \group_end:
4452 }{}

```

(End definition for `\varnotation`. This function is documented on page 84.)

```
\symdef
4453 \stex_keys_define:nnnn{symdef}{}{}{decl,notation}
4454
4455 \cs_new_protected:Nn \stex_symdef_styledefs: {
4456   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
4457   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
4458   \tl_set_eq:NN \thistype \l_stex_key_type_tl
4459   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
4460   \tl_set_eq:NN \thisargs \l_stex_key_args_str
4461   \tl_clear:N \thisstyle
4462   \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4463   \def\thisnotation{
4464     $ \let\l_stex_current_symbol_str \thisdecluri
4465     \def\comp{\_comp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{%
4466       \l_stex_notation_make_args:
4467     }$}
4468   }
4469 }
4470
4471 \stex_new_stylable_cmd:nnnn {symdef} { m 0{} m} {
4472   \stex_keys_set:nn{symdef}{#2}
4473   \str_set:Nx \l_stex_mroname_str { #1 }
4474   \stex_symdecl_top:n{#1}
4475   \stex_debug:nn{symdef}{Doing~\l_stex_current_module_str ? \l_stex_key_name_str}
4476   \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
4477   \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
4478   \stex_notation_parse:n{#3}
4479   \stex_debug:nn{Here!}{\meaning\l_stex_notation_args_tl}
4480   \stex_notation_check:
4481   \stex_notation_add:
4482   \stex_if_do_html:T{
4483     \stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4484   }
4485   \stex_if_smsmode:F{
4486     \group_begin:
4487     \stex_symdef_styledefs:
4488     \stex_style_apply:
4489     \group_end:
4490   }
4491   \stex_smsmode_do:
4492 }{}
4493
4494 \stex_deactivate_macro:Nn \symdef {module~environments}
4495 \stex_every_module:n {\stex_reactivate_macro:N \symdef}
4496 \stex_sms_allow_escape:N \symdef
```

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

```
\stex_do_default_notation_op:
4497 \cs_new_protected:Nn \stex_do_default_notation: {
4498   \stex_do_default_notation_op:
4499   \tl_if_empty:NTF \l_stex_current_args_tl {
```

```

4500     \tl_clear:N \l__stex_notations_args_tl
4501 }
4502     \__stex_notations_make_name:
4503     \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
4504     \tl_set:Nx \l__stex_notations_args_tl {
4505         \stex_map_args:N \__stex_notations_augment_arg:nn
4506     }
4507     \tl_put_right:Nn \l_stex_default_notation {\comp{}}
4508     \seq_clear:N \l_tmpa_seq
4509     \int_step_inline:nn \l_stex_current_arity_str {
4510         \seq_put_right:Nn \l_tmpa_seq {#### ##1}
4511     }
4512     \tl_put_right:Nx \l_stex_default_notation {
4513         \seq_use:Nn \l_tmpa_seq {\mathpunct{\comp{,}}}
4514     }
4515     \tl_put_right:Nn \l_stex_default_notation {\comp{}}
4516 }
4517 \tl_set:Nx \l_stex_default_notation {\STEXInternalNotation{}{0}{}{\l__stex_notations_args_}
4518     \exp_args:No \exp_not:n \l_stex_default_notation
4519 }
4520 }
4521
4522 \cs_new:Nn \__stex_notations_augment_arg:nn {
4523     #1#2{0}{}
4524 }
4525
4526 \cs_new_protected:Nn \__stex_notations_make_name: {
4527     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq ? \l_stex_current_symbol_str
4528     \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
4529     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq / \l__stex_notations_name_str
4530     \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
4531 }
4532
4533 \cs_new_protected:Nn \stex_do_default_notation_op: {
4534     \__stex_notations_make_name:
4535     \tl_set:Nx \l_stex_default_notation {\exp_not:N \maincomp{ \exp_not:N \mathrm {\l__stex_no
4536 }

```

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

## \STEXInternalNotation

```

4537 % 1: variant 2: operator precedence 3: intent 4: arguments 5: code 6: next
4538
4539 \cs_new_protected:Npn \STEXInternalNotation #1 #2 #3 #4 #5 #6 {
4540     \__stex_notations_process_notation:nnnnnn{#1}{#2}{#3}{#4}{#5}{#
4541         \l__stex_notations_code_tl
4542         #6
4543     }
4544 }
4545
4546 \cs_new_protected:Npn \__stex_notations_process_notation:nnnnnn #1 #2 #3 #4 {
4547     \tl_if_empty:nTF{#4}{
4548         \__stex_notations_simple:nnnnn{#1}{#2}{#3}
4549     }

```

```

4550     \__stex_notations_complex:nnnnnn{#1}{#2}{#3}{#4}
4551 }
4552 }
4553
4554 \cs_new_protected:Nn \__stex_notations_simple:nnnn {
4555     \stex_debug:nn{Notation~code}{\tl_to_str:n{#4}}
4556     \tl_set:Nn \l__stex_notations_code_tl {
4557         \cs_set:Npn \l__stex_notations_cs {
4558             \__stex_maybe_brackets:nn{#2}{
4559                 \__stex_term_oms_or_omv:nnn{#1}{#3}{#4}
4560             }
4561         }
4562         \l__stex_notations_cs
4563     }
4564     \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4565     #5
4566 }
4567
4568 \cs_new_protected:Nn \__stex_notations_complex:nnnnnn {
4569     \stex_debug:nn{Notation~code}{\tl_to_str:n{#5}}
4570     \int_zero:N \l_tmpa_int
4571     \tl_set:Nn \l__stex_notations_pre_tl {\cs_set_eq:NN \__stex_term_oma_or_omb:nnn \__stex_term_
4572     \tl_set:Nn \l__stex_notations_code_tl {
4573         \cs_generate_from_arg_count:NNnn \l__stex_notations_cs \cs_set:Npn \l_tmpa_int
4574         {
4575             \__stex_maybe_brackets:nn{#2}{
4576                 \__stex_term_oma_or_omb:nnn{#1}{#3}{#4}
4577                 \bool_set_false:N \l_stex_brackets_dones_bool
4578                 #5
4579             }
4580         }
4581     }
4582     \l__stex_notations_cs
4583 }
4584 \tl_set:Nn \l__stex_notations_after_tl{
4585     \exp_args:NNo
4586     \tl_put_left:Nn \l__stex_notations_code_tl \l__stex_notations_pre_tl
4587     \tl_put_left:Nx \l__stex_notations_code_tl {
4588         \int_set:Nn \l_tmpa_int {\int_use:N \l_tmpa_int}
4589     }
4590     \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4591     #6
4592 }
4593 \__stex_notations_parse_notation_args:nnnw #4 \__stex_notations_args_end:
4594 }
4595
4596 \cs_new_protected:Npn \__stex_notations_parse_notation_args:nnnw #1 #2 #3 #4 #5 \__stex_no
4597     \tl_if_empty:nTF{#5}{
4598         \__stex_notations_add_last:nnnnn{#1}{#2}{#3}{#4}{#5}
4599     }{
4600         \__stex_notations_add_next:nnnnnn{#1}{#2}{#3}{#4}{#5}{#5}
4601     }
4602 }
4603

```

```

4604 \cs_new_protected:Nn \__stex_notations_add_next:nnnnnn {
4605   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#6}
4606   \__stex_notations_parse_notation_args:nnnw #5 \__stex_notations_args_end:
4607 }
4608
4609 \cs_new_protected:Nn \__stex_notations_add_last:nnnnn {
4610   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#5}
4611   \l__stex_notations_after_tl
4612 }
4613
4614 \cs_new_protected:Nn \__stex_notations_add:nnnnn {
4615   \int_incr:N \l_tmpa_int
4616   \str_case:nn{#2} {
4617     i {
4618       \tl_put_right:Nn \l__stex_notations_code_tl {
4619         {\__stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4620       }
4621     }
4622     b {
4623       \tl_set:Nn \l__stex_notations_pre_tl {
4624         \cs_set_eq:NN \__stex_term_oma_or_omb:nnn \__stex_term_omb:nnn
4625       }
4626       \tl_put_right:Nn \l__stex_notations_code_tl {
4627         {\__stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4628       }
4629     }
4630     a {
4631       \tl_put_right:Nn \l__stex_notations_code_tl {
4632         {\__stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4633       }
4634     }
4635     B {
4636       \tl_set:Nn \l__stex_notations_pre_tl {
4637         \cs_set_eq:NN \__stex_term_oma_or_omb:nnn \__stex_term_omb:nnn
4638       }
4639       \tl_put_right:Nn \l__stex_notations_code_tl {
4640         {\__stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4641       }
4642     }
4643   }
4644 }

```

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

## a/B-mode argument handling

`\argsep`

```

4645 \cs_new_protected:Nn \__stex_notations_check_aB_arg:Nn {
4646   \exp_args:Ne \cs_if_eq:NNF {\tl_head:n{#2}}
4647   \__stex_term_arg_aB:nnnnn {
4648     \msg_error:nnx{\stex}{error/assocarg}{\tl_to_str:n{#1}}
4649   }
4650 }
4651

```

```

4652 \cs_new_protected:Npn \argsep #1 #2 {
4653   \__stex_notations_check_aB_arg:Nn\argsep{#1}
4654   \stex_pseudogroup_with:nn{\__stex_term_do_aB_clist:}{%
4655     \tl_set:Nn \__stex_term_do_aB_clist: {%
4656       \seq_use:Nn \l_stex_aB_args_seq {#2}%
4657     }%
4658   }%
4659 }%
4660 }

```

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

### \argmap

```

4661 \cs_new_protected:Npn \argmap #1 #2 #3 {
4662   \__stex_notations_check_aB_arg:Nn\argmap{#1}
4663   \stex_pseudogroup_with:nn{%
4664     \__stex_term_do_aB_clist: %
4665     \__stex_notations_map_cs: %
4666   }{%
4667     \cs_set:Npn \__stex_notations_map_cs: ##1 { #2 }%
4668     \tl_set:Nn \__stex_term_do_aB_clist: {%
4669       \seq_clear:N \l_tmpa_seq%
4670       \seq_map_inline:Nn \l_stex_aB_args_seq {%
4671         \tl_if_eq:nnTF{##1}{\ellipses}{%
4672           \seq_put_right:Nn \l_tmpa_seq \ellipses%
4673         }{%
4674           \seq_put_right:Nx \l_tmpa_seq {%
4675             \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }%
4676           }%
4677         }%
4678       }%
4679       \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq%
4680       \seq_use:Nn \l_stex_aB_args_seq {#3}%
4681     }%
4682   }%
4683 }%
4684 }

```

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

### \argarraymap

```

4685 \int_new:N \l__stex_notations_clist_count_int
4686 \cs_new_protected:Npn \argarraymap #1 #2 #3 #4 {
4687   \__stex_notations_check_aB_arg:Nn\argarraymap{#1}
4688   \stex_pseudogroup_with:nn{%
4689     \__stex_term_do_aB_clist: %
4690     \__stex_notations_map_cs: %
4691   }{%
4692     \cs_set:Npn \__stex_notations_map_cs: ##1 { #3 }%
4693     \int_set:Nn \l__stex_notations_clist_count_int {\exp_args:No\clist_count:n{\tl_to_str:n{%
4694       \tl_set:Nn \__stex_term_do_aB_clist: {%
4695         \tl_clear:N \l_tmpa_t1%
4696         \int_zero:N \l_tmpa_int%
4697         \seq_map_inline:Nn \l_stex_aB_args_seq {%
4698           \int_incr:N \l_tmpa_int%

```

```

4699   \int_compare:nNnT \l_tmpa_int > \l__stex_notations_clist_count_int {
4700     \int_set:Nn \l_tmpa_int 1
4701   }
4702   \tl_put_right:Nx \l_tmpa_tl {
4703     \exp_after:wN \exp_not:n \exp_after:wN \l__stex_notations_map_cs: {##1} }
4704     \clist_item:nn{#4}\l_tmpa_int
4705   }
4706 }
4707 \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4708 \begin{array}{#2}
4709   \l_tmpa_tl
4710 \end{array}
4711 }
4712 #1
4713 }
4714 }
```

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

### 13.8.3 Variables

```
4715 <@=stex_vars>
```

`\vardef`

```

4716 \tl_new:N \l_stex_variables_prop
4717 \bool_new:N \l__stex_vars_bind_bool
4718 \cs_new_protected:Nn \stex_variable:nnnnnnnN {}
4719 \stex_keys_define:nnnn{\vardef}{
4720   \bool_set_false:N \l__stex_vars_bind_bool
4721 }{
4722   bind .bool_set:N = \l__stex_vars_bind_bool
4723 }{symdef}
4724
4725 \stex_new_stylable_cmd:nnnn {\vardef} { m 0{} m} {
4726   \stex_keys_set:nn{\vardef}{#2}
4727   \str_set:Nx \l_stex_macroname_str { #1 }
4728   \str_if_empty:NT \l_stex_key_name_str {
4729     \str_set:Nx \l_stex_key_name_str { #1 }
4730   }
4731
4732 \stex_symdecl_do:
4733 \stex_symdecl_check_terms:
4734 \__stex_vars_add:
4735 \__stex_vars_macro:
4736 \stex_if_do_html:T \__stex_vars_html:
4737
4738 \int_set:Nn \l_stex_get_symbol_arity_int {\l_stex_get_symbol_arity_int}
4739 \stex_debug:nn{\vardef}{Doing~\l_stex_key_name_str}
4740 \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4741 \stex_notation_parse:n{#3}
4742 \stex_if_check_terms:T{ \stex_notation_check: }
4743 \stex_vardecl_notation_macro:
4744 \stex_if_do_html:T {
4745   \def\comp{\_varcomp}
4746   \stex_notation_do_html:n \l_stex_key_name_str
```

```

4747 }
4748 \group_begin:
4749 \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4750 \tl_clear:N \thisstyle
4751 \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4752 \def\thisnotation{
4753   $ \let\l_stex_current_symbol_str\thisvarname
4754     \def\comp{\_varcomp}\exp_args:Nne \use:nf{\l_stex_notation_macrocode_{\cs}}{
4755       \l_stex_notation_make_args:
4756     }$}
4757 }
4758 \stex_style_apply:
4759 \group_end:\ignorespaces
4760 }{ }
4761
4762 \cs_new_protected:Nn \__stex_vars_add: {
4763   \exp_args:NNNo \exp_args:NNnx
4764   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4765     {\l_stex_mroname_str}
4766     {\l_stex_key_name_str}
4767     {\int_use:N \l_stex_get_symbol_arity_int}
4768     {\l_stex_get_symbol_args_tl}
4769     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4770     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4771     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4772     \stex_invoke_symbol:
4773   }
4774 }
4775
4776 \cs_new_protected:Nn \__stex_vars_macro: {
4777   \tl_set:cx{\l_stex_mroname_str}{%
4778     \stex_invoke_variable:nnnnnnN
4779     {\l_stex_key_name_str}
4780     {\int_use:N \l_stex_get_symbol_arity_int}
4781     {\l_stex_get_symbol_args_tl}
4782     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4783     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4784     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4785     \stex_invoke_symbol:
4786   }
4787 }
4788
4789 \cs_new_protected:Nn \__stex_vars_html: {
4790   \stex_if_do_html:T {
4791     \hbox\bgroup\exp_args:Ne \stex_annotation_invisible:nn {
4792       \shtml:vardef = {\l_stex_key_name_str},
4793       \shtml:args = {\l_stex_key_args_str}
4794       \str_if_empty:NF \l_stex_mroname_str {,
4795         \shtml:macroname={\l_stex_mroname_str}
4796       }
4797       \str_if_empty:NF \l_stex_key_assoc_str {,
4798         \shtml:assotype={\l_stex_key_assoc_str}
4799       }
5000       \str_if_empty:NF \l_stex_key_role_str {,

```

```

4801         shtml:role={\l_stex_key_role_str}
4802     }
4803     \str_if_empty:NF \l_stex_key_reordered_str {
4804         shtml:reorderargs={\l_stex_key_reordered_str}
4805     }
4806     \bool_if:NT \l_stex_vars_bind_bool {
4807         shtml:bind={}
4808     }
4809 }
4810 \l_stex_annotation_force_break:n{
4811     \bool_set_true:N \stex_in_invisible_html_bool
4812     \tl_if_empty:NF \l_stex_key_type_tl {
4813         \stex_annotation:nn{shtml:type={}{}$ \l_stex_key_type_tl$}
4814     }
4815     \tl_if_empty:NF \l_stex_key_def_tl {
4816         \stex_annotation:nn{shtml:definiens={}{}$ \l_stex_key_def_tl$}
4817     }
4818     \tl_if_empty:NF \l_stex_key_return_tl{
4819         \exp_args:Nno \use:n{
4820             \cs_generate_from_arg_count:NNnn \l_stex_vars_cs
4821             \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4822             \tl_set:Nx \l_stex_vars_args_tl {\l_stex_map_args:N \l_stex_return_args:nn}
4823             $ \stex_annotation:nn{shtml:returntype={}{}{
4824                 \exp_after:wN \l_stex_vars_cs \l_stex_vars_args_tl!}{}$}
4825     }
4826     \tl_if_empty:NF \l_stex_key_argtypes_clist {
4827         \stex_annotation:nn{shtml:argtypes={}{}{
4828             \l_stex_annotation_force_break:n{
4829                 \clist_map_inline:NN \l_stex_key_argtypes_clist {
4830                     $ \stex_annotation:nn{shtml:type={}{}##1}{}$}
4831             }
4832         }
4833     }
4834 }
4835 }
4836 }\egroup
4837 }
4838 }

```

(End definition for `\vardef`. This function is documented on page 84.)

#### `\_stex_vardecl_notation_macro:`

```

4839 \cs_new_protected:Nn \_stex_vardecl_notation_macro: {
4840     \tl_set_eq:cN {l_stex_notation_
4841         \l_stex_key_name_str _ 
4842         \l_stex_key_variant_str _cs
4843     } \l_stex_notation_macrocode_cs
4844     \cs_if_exist:cF {l_stex_notation_\l_stex_key_name_str __cs} {
4845         \tl_set_eq:cN {l_stex_notation_\l_stex_key_name_str __cs}
4846             \l_stex_notation_macrocode_cs
4847     }
4848     \tl_if_empty:NF \l_stex_key_op_tl {
4849         \tl_set_eq:cN {l_stex_notation_\l_stex_key_name_str _op_
4850             \l_stex_key_variant_str _cs} \l_stex_key_op_tl

```

```

4851   \cs_if_exist:cF {l_stex_notation_\l_stex_key_name_str _op__cs} {
4852     \cs_set_eq:cN{l_stex_notation_\l_stex_key_name_str _op__cs}
4853     \l_stex_key_op_tl
4854   }
4855 }
4856 }
```

(End definition for `\_stex_vardecl_notation_macro`. This function is documented on page 114.)

```

\stex_get_symbol_or_var:n
  \stex_get_var:n
    4857 \cs_new_protected:Nn \__stex_vars_set_vars:nnnnnnN {
    4858   \stex_debug:nn{symbols}{Variable~#1~found}
    4859   \cs_set:Npn \_stex_variable:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {}
    4860   \str_clear:N \l_stex_get_symbol_mod_str
    4861   \str_set:Nn \l_stex_get_symbol_name_str {\#1}
    4862   \int_set:Nn \l_stex_get_symbol_arity_int {\#2}
    4863   \tl_set:Nn \l_stex_get_symbol_args_tl {\#3}
    4864   \tl_set:Nn \l_stex_get_symbol_def_tl {\#4}
    4865   \tl_set:Nn \l_stex_get_symbol_type_tl {\#5}
    4866   \tl_set:Nn \l_stex_get_symbol_return_tl {\#6}
    4867   \tl_set:Nn \l_stex_get_symbol_invoke_cs {\#7}
    4868 }
    4869
    4870 \cs_new_protected:Nn \__stex_vars_get_var:n {
    4871   \prop_map_inline:Nn \l_stex_variables_prop {
    4872     \__stex_vars_check_var:nnnnnnnnN {\#1} ##2
    4873   }
    4874 }
    4875
    4876 \cs_new_protected:Nn \__stex_vars_check_var:nnnnnnnnN {
    4877   \str_if_eq:nnTF{\#1}{\#2} {
    4878     \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {\#3}{\#4}{\#5}{\#6}{\#7}{\#8}{\#9}}
    4879   }{
    4880     \str_if_eq:nnT{\#1}{\#3} {
    4881       \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {\#3}{\#4}{\#5}{\#6}{\#7}{\#8}{\#9}}
    4882     }
    4883   }
    4884 }
    4885
    4886 \cs_new_protected:Nn \stex_get_var:n {
    4887   \str_clear:N \l_stex_get_symbol_name_str
    4888   \__stex_vars_get_var:n{\#1}
    4889   \str_if_empty:NT \l_stex_get_symbol_name_str {
    4890     \msg_error:nnn{stex}{error/unknownsymbol}{\#1}
    4891   }
    4892 }
    4893
    4894 \cs_new_protected:Nn \stex_get_symbol_or_var:n {
    4895   \str_clear:N \l_stex_get_symbol_name_str
    4896   \__stex_vars_get_var:n{\#1}
    4897   \str_if_empty:NT \l_stex_get_symbol_name_str {
    4898     \stex_debug:nn{symbols}{No~variable~#1~found}
    4899     \stex_get_symbol:n{\#1}
    4900   }
    4901 }
```

(End definition for `\stex_get_symbol_or_var:n` and `\stex_get_var:n`. These functions are documented on page 114.)

```
\svar
4902 \NewDocumentCommand \svar {O{} m}{
4903   \group_begin:
4904     \tl_if_empty:nTF{#1}{
4905       \str_set:Nn \l_stex_current_symbol_str {#2}
4906     }{
4907       \str_set:Nn \l_stex_current_symbol_str {#1}
4908     }
4909     \bool_if:NTF \l_stex_allow_semantic_bool{
4910       \tl_clear:N \l_stex_current_term_tl
4911       \stex_term_omv:nnn{}{}{\l_varcomp{#2}}
4912     }{
4913       \msg_error:nxxx{stex}{error/notallowed}{Variable}{\l_stex_current_symbol_str}
4914     }
4915   \group_end:
4916 }
```

(End definition for `\svar`. This function is documented on page 84.)

### 13.8.4 Sequences

```
4917 <@=stex_seqs>
```

```
\varseq
```

```
4918 \stex_new_stylable_cmd:nnnn {varseq}{m O{} m m} {
4919   \stex_keys_set:nn{symdef}{#2}
4920   \str_set:Nx \l_stex_mroname_str { #1 }
4921   \str_if_empty:NT \l_stex_key_name_str {
4922     \str_set:Nx \l_stex_key_name_str { #1 }
4923   }
4924   \str_if_empty:NT \l_stex_key_args_str {
4925     \str_set:Nn \l_stex_key_args_str {1}
4926   }
4927   \stex_symdecl_do:
4928
4929   \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4930   \clist_set:Nn \l_stex_seqs_range_clist {#3}
4931   \tl_if_empty:NTF \l_stex_key_op_tl {
4932     \stex_notation_parse:n{#4}
4933     \tl_clear:N \l_stex_key_op_tl
4934   }{
4935     \stex_notation_parse:n{#4}
4936   }
4937   \stex_if_do_html:T \l_stex_seqs_html:
4938   \stex_if_check_terms:T \l_stex_seqs_check_terms:
4939   \l_stex_seqs_add:
4940   \l_stex_seqs_macro:
4941   \stex_if_check_terms:T \l_stex_notation_check:
4942   \l_stex_vardecl_notation_macro:
4943   \group_begin:
4944   \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4945   \tl_clear:N \thisstyle
```

```

4946 \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4947 \def\thisnotation{
4948   \$\let\l_stex_current_symbol_str\thisvarname
4949   \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_mmacrocode_{}}{
4950     \__stex_seqs_make_args:
4951   }$}
4952 }
4953 \stex_style_apply:
4954 \group_end:\ignorespaces
4955 }{}}

4956 \cs_new_protected:Nn \__stex_seqs_add: {
4957   \exp_args:NNNo \exp_args:NNnx
4958   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4959     {\l_stex_mroname_str}
4960     {\l_stex_key_name_str}
4961     {\int_use:N \l_stex_get_symbol_arity_int}
4962     {\l_stex_get_symbol_args_tl}
4963     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4964     {\exp_args:No \exp_not:n \l_stex_seqs_range_clist}
4965     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4966     \stex_invoke_sequence:
4967   }
4968 }
4969 }

4970 \cs_new_protected:Nn \__stex_seqs_macro: {
4971   \tl_set:cx{\l_stex_mroname_str}{%
4972     \__stex_invoke_variable:nnnnnnN
4973       {\l_stex_key_name_str}
4974       {\int_use:N \l_stex_get_symbol_arity_int}
4975       {\l_stex_get_symbol_args_tl}
4976       {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4977       {\exp_args:No \exp_not:n \l_stex_seqs_range_clist}
4978       {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4979       \stex_invoke_sequence:
4980   }
4981 }
4982 }

4983 \cs_new_protected:Nn \__stex_seqs_make_args: { \TODO }
4984 \cs_new_protected:Nn \__stex_seqs_check_terms: { \TODO }
4985

4986 \cs_new_protected:Nn \__stex_seqs_html: {
4987   \exp_args:Ne \stex_annotation_invisible:nn {
4988     \shtml:vareq = {\l_stex_key_name_str},
4989     \shtml:args = {\l_stex_key_args_str}
4990     \str_if_empty:NF \l_stex_mroname_str {,
4991       \shtml:macroname={\l_stex_mroname_str}
4992     }
4993     \str_if_empty:NF \l_stex_key_assoc_str {,
4994       \shtml:assoc_type={\l_stex_key_assoc_str}
4995     }
4996     \str_if_empty:NF \l_stex_key_role_str {,
4997       \shtml:role={\l_stex_key_role_str}
4998     }
4999   }

```

```

5000     \str_if_empty:NF \l_stex_key_reorder_str {
5001         shtml:reorderargs={\l_stex_key_reorder_str}
5002     }
5003 }{\hbox\bgroup
5004     \_stex_annotate_force_break:n{
5005         \tl_if_empty:NF \l_stex_key_type_tl {
5006             \stex_annotate:nn{shtml:type={}{}{$\l_stex_key_type_tl$}}
5007         }
5008         \tl_if_empty:NF \l_stex_key_def_tl {
5009             \stex_annotate:nn{shtml:definiens={}{}{$\l_stex_key_def_tl$}}
5010         }
5011         \tl_if_empty:NF \l_stex_key_return_tl{
5012             \exp_args:Nno \use:n{
5013                 \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs
5014                 \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
5015                 \tl_set:Nx \l__stex_seqs_args_tl {\l_stex_map_args:N \l_stex_return_args:nn}
5016                 \stex_annotate:nn{shtml:returntype={}{}{
5017                     $\exp_after:wN \l__stex_seqs_cs \l__stex_seqs_args_tl!$}
5018                 }
5019                 \tl_if_empty:NF \l_stex_key_argtypes_clist {
5020                     \stex_annotate:nn{shtml:argtypes={}{}{
5021                         \_stex_annotate_force_break:n{
5022                             \clist_map_inline:Nn \l_stex_key_argtypes_clist {
5023                                 \stex_annotate:nn{shtml:type={}{}{$\#\#1$}}
5024                             }
5025                         }
5026                     }
5027                 }
5028             }
5029         \egroup}
5030 }

```

(End definition for `\varseq`. This function is documented on page 84.)

```

\stex_invoke_sequence:
5031 \cs_new_protected:Nn \stex_invoke_sequence: {
5032     \peek_charcode_remove:NTF ! {
5033         \peek_charcode:NTF [ \__stex_seqs_do_op:w { \__stex_seqs_do_op:w [] } ]
5034         } \__stex_seqs_do_first:
5035     }
5036
5037 \cs_new_protected:Npn \__stex_seqs_do_op:w [#1] {
5038     \cs_if_exist:cTF {l_stex_notation_\l_stex_current_symbol_str _op_#1_cs} {
5039         \_stex_maybe_brackets:nn{\neginfpref}{}
5040         \_stex_term_oms_or_omv:nnn{#1}{}
5041         {\use:c{l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}}
5042     }
5043     \group_end:
5044 }{
5045     \__stex_seqs_get_index_notation:n{#1}
5046     \peek_charcode:NTF [ \__stex_seqs_doop_range:w { \__stex_seqs_doop_range:w[] } ]
5047 }
5048
5049

```

```

5050 \cs_new_protected:Npn \__stex_seqs_doop_range:w [#1] {
5051   \bool_set_true:N \l_stex_allow_semantic_bool
5052   \clist_clear:N \l__stex_seqs_clist
5053   \clist_map_function:NN \l_stex_current_type_tl \__stex_seqs_doop_arg:n
5054     \stex_annotation:nn{
5055       \shtml:term=OMV,
5056       \shtml:head={\l_stex_current_symbol_str},
5057       \shtml:notationid={}
5058     }{
5059       \l__stex_seqs_clist
5060     }
5061   \group_end:
5062 }
5063
5064 \cs_new_protected:Nn \__stex_seqs_doop_arg:n {
5065   \tl_if_eq:nnTF{#1}{\ellipses}{
5066     \clist_put_right:Nn \l__stex_seqs_clist {
5067       \ellipses
5068     }
5069   }{
5070     \clist_put_right:Nn \l__stex_seqs_clist {
5071       \exp_args:No \str_if_eq:nnTF \l_stex_current_arity_str {1} {
5072         \group_begin:
5073           \l__stex_seqs_cs \group_end: {#1}
5074       }{
5075         \group_begin:
5076           \l__stex_seqs_cs \group_end: #1
5077       }
5078     }
5079   }
5080 }
5081 }
5082
5083 \cs_new_protected:Nn \__stex_seqs_get_index_notation:n {
5084   \cs_if_exist:cTF {\l_stex_notation_\l_stex_current_symbol_str _#1_cs} {
5085     \cs_set_eq:Nc \l__stex_seqs_cs {\l_stex_notation_\l_stex_current_symbol_str _#1_cs}
5086   }{
5087     \stex_do_default_notation:
5088     \cs_set_eq:NN \l__stex_seqs_cs \l_stex_default_notation
5089   }
5090 }
5091
5092
5093 \cs_new:Nn \__stex_seqs_do_first_arg:n {{\exp_not:n{## #1}}}
5094
5095 \cs_new_protected:Nn \__stex_seqs_do_first: {
5096   \exp_args:Nnx \use:nn{
5097     \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs \cs_set:Npn
5098     \l_stex_current_arity_str }{
5099       \tl_set:Nn \exp_not:N \l__stex_seqs_first_args_tl {
5100         \int_step_function:nN \l_stex_current_arity_str \__stex_seqs_do_first_arg:n
5101       }
5102       \exp_not:N \__stex_seqs_do_first_next:
5103     }

```

```

5104   \l__stex_seqs_cs
5105 }
5106
5107 \cs_new_protected:Nn \__stex_seqs_do_first_next: {
5108   \peekCharCode_remove:NTF ! {
5109     \peekCharCode:NTF [ \__stex_seqs_do_one:w {\__stex_seqs_do_one:w []}
5110   }{
5111     \peekCharCode:NTF [ \__stex_seqs_do_all:w {\__stex_seqs_do_all:w []}
5112   }
5113 }
5114
5115 \cs_new_protected:Npn \__stex_seqs_do_one:w [#1] {
5116   \__stex_seqs_get_index_notation:n{#1}
5117   \stex_debug:nn{HERE~seq-one}{\meaning\l__stex_seqs_cs^J\meaning\l__stex_seqs_first_args_t}
5118   \exp_args:Nno\use:nn{\l__stex_seqs_cs\group_end:}\l__stex_seqs_first_args_tl
5119 }
5120
5121 \cs_new_protected:Npn \__stex_seqs_do_all:w [#1] {
5122   \stex_debug:nn{HERE~seq-all}{\meaning\l__stex_seqs_first_args_tl}
5123   \exp_args:Nno\use:nn{\l__stex_invoke_notation:w [#1]}\l__stex_seqs_first_args_tl
5124 }

```

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

### \seqmap

```

5125 \cs_new_protected:Npn \seqmap #1 #2 {
5126   \symuse{Metatheory?sequence-expression}{\seqmap{#1}{#2}}%\l_tmpa_tl {#2}
5127 }

```

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

## 13.8.5 Expressions

```
5128 <@=stex_expr>
```

Various variables:

```

5129 \bool_new:N \l_stex_allow_semantic_bool
5130 \bool_set_true:N \l_stex_allow_semantic_bool
5131 \tl_new:N \l_stex_current_term_tl
5132 \tl_set:Nn \l_stex_every_symbol_tl {
5133   \bool_set_false:N \l_stex_allow_semantic_bool
5134 }

```

### \\_stex\_next\_symbol:n

```

5135 \tl_new:N \l__stex_expr_reset_tl
5136 \cs_new_protected:Nn \_stex_next_symbol:n {
5137   \tl_set:Nx \l_stex_every_symbol_tl {
5138     \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5139       \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
5140         \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5141       }
5142       \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5143         \exp_args:No \exp_not:n \l__stex_expr_reset_tl
5144       }
5145 }

```

```

5146   \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
5147     \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5148   }
5149   \exp_not:n{ \aftergroup \l__stex_expr_reset_tl }
5150   \exp_not:N \l_stex_every_symbol_tl
5151   \exp_not:n{ #1 }
5152 }
5153 }
5154 \cs_generate_variant:Nn \_stex_next_symbol:n {e}

```

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

### \STEXinvisible

```

5155 \cs_new_protected:Npn \STEXinvisible #1 {
5156   \stex_annotation_invisible:n { #1 }
5157 }

```

(End definition for `\STEXinvisible`. This function is documented on page 70.)

## Invoking Semantic Macros

`\_stex_invoke_symbol:nnnnnnnN`

```

5158 \cs_new_protected:Nn \_stex_invoke_symbol:nnnnnnnN {
5159   \bool_if:NTF \l_stex_allow_semantic_bool{
5160     \stex_if_html_backend:T{\ifvmode\indent\fi}
5161     \__stex_expr_setup:nnnnnn{\_comp}{#1?#2}{#3}{#4}{#7}{#6}
5162     \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
5163     \tl_put_right:Nn \l_stex_current_redo_tl{
5164       \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
5165     }
5166     #8
5167   }{
5168     \msg_error:nnxx{stex}{error/notallowed}{#1?#2}{\l_stex_current_symbol_str}
5169   }
5170 }
5171 \cs_generate_variant:Nn \_stex_invoke_symbol:nnnnnnnN {ooxooooN}
5172
5173 \cs_new_protected:Nn \__stex_expr_setup:nnnnnn {
5174   \group_begin:
5175   \tl_clear:N \l_stex_return_notation_tl
5176   \tl_set:Nn \l_stex_current_redo_tl {
5177     \let \this \stex_current_this:
5178     \def\comp{#1}
5179     \def\maincomp{\comp}
5180     \str_set:Nn \l_stex_current_symbol_str {#2}
5181     \str_set:Nn \l_stex_current_arity_str{ #3 }
5182     \tl_set:Nn \l_stex_current_args_tl{ #4 }
5183     \tl_set:Nn \l_stex_current_return_tl{ #5 }
5184     \tl_set:Nn \l_stex_current_type_tl{ #6 }
5185     \tl_clear:N \l_stex_current_term_tl
5186   }
5187   \tl_put_right:Nx \l_stex_current_redo_tl {
5188     \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5189   }

```

```

5190   \l_stex_current_redo_tl
5191 }

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

\\_stex\_invoke\_variable:nnnnnn

```

5192 \cs_new_protected:Nn \_stex_invoke_variable:nnnnnnN {
5193   \bool_if:NTF \l_stex_allow_semantic_bool{
5194     \stex_if_html_backend:T{\ifvmode\indent\fi}
5195     \__stex_expr_setup:nnnnnn{\_varcomp}{#1}{#2}{#3}{#6}{#5}
5196     \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
5197     \tl_put_right:Nn \l_stex_current_redo_tl {
5198       \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
5199     }
5200     #7
5201   }{
5202     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
5203   }
5204 }
```

(End definition for \\_stex\_invoke\_variable:nnnnnn. This function is documented on page ??.)

### \symuse

```

5205 \cs_new_protected:Npn \symuse #1 {
5206   \stex_get_symbol:n{#1}
5207   \exp_args:Nno \use:n {\_stex_invoke_symbol:ooxxxxN
5208   \l_stex_get_symbol_mod_str
5209   \l_stex_get_symbol_name_str
5210   {\int_use:N \l_stex_get_symbol_arity_int}
5211   \l_stex_get_symbol_args_tl
5212   \l_stex_get_symbol_def_tl
5213   \l_stex_get_symbol_type_tl
5214   \l_stex_get_symbol_return_tl}
5215   \l_stex_get_symbol_invoke_cs
5216 }
```

(End definition for \symuse. This function is documented on page 78.)

**\stex\_invoke\_symbol:** Top-Level: Check whether text/math mode or custom notation, whether delimited by !, return code etc.

```

5217 \cs_new_protected:Nn \stex_invoke_symbol: {
5218   \stex_debug:nn{expressions}{Invoking-\l_stex_current_symbol_str}
5219   \mode_if_math:TF \__stex_expr_invoke_math: \__stex_expr_invoke_text:
5220 }
5221
5222 \cs_new_protected:Nn \__stex_expr_invoke_text: {
5223   \stex_debug:nn{expressions}{text-mode}
5224   \peek_charcode_remove:NTF ! \__stex_expr_invoke_op_custom:n \__stex_expr_invoke_custom:n
5225 }
5226
5227 \cs_new_protected:Nn \__stex_expr_invoke_math: {
5228   \stex_debug:nn{expressions}{math-mode}
5229   \peek_charcode_remove:NTF ! {
5230     % operator
5231     \peek_charcode_remove:NTF * \__stex_expr_invoke_op_custom:n {
```

```

5232     % op notation
5233     \peek_charcode:NTF [ \__stex_expr_invoke_op_notation:w {
5234         \__stex_expr_invoke_op_notation:w []
5235     }
5236 }
5237 }{
5238     \peek_charcode_remove:NTF * \__stex_expr_invoke_custom:n {
5239         % normal
5240         \peek_charcode:NTF [ \__stex_invoke_notation:w {
5241             \__stex_invoke_notation:w []
5242         }
5243     }
5244 }
5245 }

Notations:

5246 \cs_new_protected:Npn \__stex_invoke_notation:w [#1] {
5247     \stex_debug:nn{expressions}{using~notation~#1~for~\l_stex_current_symbol_str}
5248     \cs_if_exist:cTF{\l_stex_notation_\l_stex_current_symbol_str _#1_cs} {
5249         \tl_if_empty:NTF \l_stex_current_return_tl {
5250             \stex_debug:nn{expressions}{return~empty}
5251             \use:c{\l_stex_notation_\l_stex_current_symbol_str _#1_cs}{\group_end:\__stex_eat_exclamation_point}
5252         }{
5253             \stex_debug:nn{expressions}{return?}
5254             \exp_after:wN\exp_after:wN\exp_after:wN
5255             \__stex_expr_invoke_return_maybe:n
5256             \exp_after:wN\exp_after:wN\exp_after:wN
5257             {\cs:w \l_stex_notation_\l_stex_current_symbol_str _#1_cs \cs_end: {}}
5258         }
5259     }{
5260         \stex_do_default_notation:
5261         \tl_if_empty:NTF \l_stex_current_return_tl {
5262             \l_stex_default_notation{\group_end:\__stex_eat_exclamation_point:}
5263         }{
5264             \exp_after:wN
5265             \__stex_expr_invoke_return_maybe:n
5266             \exp_after:wN
5267             {\l_stex_default_notation {}}
5268         }
5269     }
5270 }

5271 \cs_new_protected:Npn \__stex_expr_invoke_op_notation:w [#1] {
5272     \stex_debug:nn{expressions}{op~notation~for~\l_stex_current_symbol_str}
5273     \cs_if_exist:cTF{\l_stex_notation_\l_stex_current_symbol_str _op_#1_cs} {
5274         \__stex_maybe_brackets:nn{\neginfpref}{%
5275             \__stex_term_oms_or_omv:nnn{#1}{}
5276             {\use:c{\l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}}}
5277         }
5278         \group_end:
5279     }{
5280         \int_compare:nNnTF \l_stex_current_arity_str = 0 {
5281             \tl_clear:N \l_stex_current_return_tl
5282             \__stex_invoke_notation:w [#1]

```

```

5284 }{
5285   \stex_do_default_notation_op:
5286   \_stex_maybe_brackets:nn{\neginfpref}{%
5287     \_stex_term_oms_or_omv:nnn{#1}{}
5288     {\l_stex_default_notation}
5289   }
5290   \group_end:
5291 }
5292 }
5293 }

Return:

5294 \cs_new_protected:Nn \__stex_expr_invoke_return_maybe:n {
5295   \tl_clear:N \l__stex_expr_return_args_tl
5296   \tl_set:Nn \l__stex_expr_return_this_tl {#1}
5297   \exp_args:Nnx \use:n {
5298     \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
5299     \cs_set:Npn \l_stex_current_arity_str } {
5300       \int_step_function:nN \l_stex_current_arity_str \__stex_expr_return_arg:n
5301       \__stex_expr_invoke_return_next:
5302     }
5303   \__stex_expr_ret_cs
5304 }

5305
5306 \cs_new:Nn \__stex_expr_return_arg:n {
5307   \tl_put_right:Nn \exp_not:N \l__stex_expr_return_args_tl {{#### #1}}
5308 }
5309
5310 \cs_new_protected:Nn \__stex_expr_invoke_return_next: {
5311   \peekCharCode_remove:NTF ! {
5312     \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl \group_end:
5313   }\__stex_expr_invoke_return:
5314 }

5315
5316 \cs_new_protected:Nn \__stex_expr_invoke_return: {
5317   \tl_set:Nx \l__stex_expr_return_this_tl {
5318     \__stex_expr_return_notation:n {
5319       \exp_after:wN \exp_after:wN \exp_after:wN
5320       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
5321         \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl
5322       }
5323     }
5324   }
5325   \stex_debug:nn{return}{Notation:\meaning\l__stex_expr_return_this_tl}
5326   \tl_put_left:Nx \l__stex_expr_return_this_tl {
5327     \group_begin:\exp_args:No \exp_not:n \l_stex_current_redo_tl
5328   }
5329   \exp_args:Nnx \use:n {
5330     \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
5331     \cs_set:Npn \l_stex_current_arity_str } {
5332       \exp_args:No \exp_not:n \l_stex_current_return_tl
5333     }
5334   \stex_debug:nn{return}{%
5335     \meaning\__stex_expr_ret_cs^~J

```

```

5336   \meaning\l__stex_expr_return_this_tl^^J
5337   \exp_args:No \exp_not:n \l__stex_expr_return_args_tl^^J
5338 }
5339 \exp_args:Nnx \use:nn {
5340   \exp_after:wN \group_end: \_stex_expr_ret_cs
5341 }{
5342   \exp_args:No \exp_not:n \l__stex_expr_return_args_tl
5343   {
5344     \exp_args:No \exp_not:n \l__stex_expr_return_this_tl
5345     \group_end:
5346   }
5347 }
5348 }
5349
5350
5351 \cs_new_protected:Nn \_stex_expr_return_notation:n {
5352   \tl_if_empty:NTF \l_stex_return_notation_tl { #1 }{
5353     \l_stex_return_notation_tl
5354   }
5355 }
5356

```

#### Custom Notations:

```

5357 \cs_new_protected:Nn \_stex_expr_invoke_op_custom:n {
5358   \stex_debug:nn{expressions}{custom~op}
5359   \bool_set_true:N \l_stex_allow_semantic_bool
5360   \_stex_term_oms_or_omv:nnn{}{}{\maincomp{#1}}
5361   \group_end:
5362 }
5363
5364 \int_new:N \l_stex_expr_arg_counter_int
5365 \cs_new_protected:Nn \_stex_expr_invoke_custom:n {
5366   \stex_debug:nn{custom}{custom~notation~for~\l_stex_current_symbol_str}
5367   \stex_pseudogroup:nn{
5368     \bool_set_true:N \l_stex_allow_semantic_bool
5369     \prop_gclear:N \l_stex_expr_customs_prop
5370     \seq_gclear:N \l_stex_expr_customs_seq
5371     \int_gzero:N \l_stex_expr_arg_counter_int
5372     \tl_if_empty:NF \l_stex_current_args_tl {
5373       \exp_after:wN \_stex_expr_add_prop_arg:nw \l_stex_current_args_tl \_stex_args_end:
5374       \cs_set_eq:NN \arg \_stex_expr_arg:n
5375     }
5376     \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
5377     \cs_set_eq:NN \_stex_expr_do_ab_next:nnn \_stex_term_oma:nnn
5378     \_stex_map_args:N \_stex_expr_check_b:nn
5379     \_stex_expr_do_ab_next:nnn{}{}{#1}
5380   }{
5381     \prop_if_exist:NT \l_stex_expr_customs_prop {
5382       \prop_gset_from_keyval:Nn \exp_not:N \l_stex_expr_customs_prop {
5383         \prop_to_keyval:N \l_stex_expr_customs_prop
5384       }
5385     }
5386     \int_gset:Nn \l_stex_expr_arg_counter_int { \int_use:N \l_stex_expr_arg_counter_int }
5387     \seq_if_exist:NT \l_stex_expr_customs_seq {

```

```

5388     \seq_gset_split:Nnn \exp_not:N \l__stex_expr_customs_seq , {
5389         \seq_use:Nn \l__stex_expr_customs_seq ,
5390     }
5391 }
5392 }
5393 % TODO check that all arguments are present
5394 \group_end:
5395 }
5396
5397 \cs_new_protected:Npn \__stex_expr_add_prop_arg:nw #1 #2 #3\_stex_args_end: {
5398     \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {}
5399     \seq_gput_right:Nn \l__stex_expr_customs_seq {#2}
5400     \tl_if_empty:nF{#3}{\__stex_expr_add_prop_arg:nw #3 \_stex_args_end:}
5401 }
5402
5403 \cs_new:Nn \__stex_expr_check_b:nn {
5404     \str_case:nn #2 {
5405         b {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \stex_term_omb:nnn}
5406         B {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \stex_term_omb:nnn}
5407     }
5408 }
5409
5410 \NewDocumentCommand \__stex_expr_arg:n {s O{} m} {
5411     \IfBooleanTF #1 {
5412         \stex_annotation_invisible:n{
5413             \__stex_expr_arg_inner:nn{#2}{#3}
5414         }
5415     }{
5416         \__stex_expr_arg_inner:nn{#2}{#3}
5417     }
5418 }
5419
5420 \cs_new_protected:Nn \__stex_expr_arg_inner:nn {
5421     \tl_if_empty:nTF{#1} {
5422         \int_gincr:N \l__stex_expr_arg_counter_int
5423         \exp_args:Ne \__stex_expr_check:nTF{ \int_use:N \l__stex_expr_arg_counter_int }{
5424             \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl
5425         }{
5426             \__stex_expr_arg_inner:nn{}
5427         }{ #2 }
5428     }{
5429         \__stex_expr_check:nTF {#1} {
5430             \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5431         }{
5432             \exp_args:No \str_case:nnTF \l_tmpb_tl {
5433                 {a} {
5434                     \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {
5435                         \l_tmpa_tl X
5436                     }
5437                     \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n { \tl_count:N \l_tmpa_tl + 1 } }
5438                 }
5439                 {B} {
5440                     \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {
5441                         \l_tmpa_tl X

```

```

5442         }
5443         \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5444     }
5445     }{
5446     \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5447     }{
5448     \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5449     }
5450   }
5451 }
5452 }
5453
5454 \prg_new_if:NNN \__stex_expr_check:n {TF} {
5455   \exp_args:NNN \prop_get:NnNTF \l__stex_expr_customs_prop {#1} \l_tmpa_tl {
5456     \tl_set:Nx \l_tmpb_tl {\seq_item:Nn \l__stex_expr_customs_seq {#1} }
5457     \tl_if_empty:NTF \l_tmpa_tl {
5458       \exp_args:NNN \prop_gput:Nnn \l__stex_expr_customs_prop
5459       { #1 }{X}
5460       \exp_args:No \str_case:nnF \l_tmpb_tl {
5461         {a} {
5462           \tl_set:Nx \l_tmpa_tl { #1 1 }
5463         }
5464         {B} {
5465           \tl_set:Nx \l_tmpa_tl { #1 1 }
5466         }
5467       }{
5468         \tl_set:Nx \l_tmpa_tl { #1 }
5469       }
5470       \prg_return_true:
5471     }{
5472       \prg_return_false:
5473     }
5474   }{
5475     \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5476     \prg_return_false:
5477   }
5478 }

5479 % #1 argnum #2 argmode #3 code
5480 \cs_new_protected:Nn \__stex_expr_arg_do:nnn {
5481   \stex_debug:nn{custom}{Doing~argument~#1~of~mode~#2:~\tl_to_str:n{#3}}
5482   \group_begin:
5483     \bool_set_true:N \l_stex_allow_semantic_bool
5484     \stex_term_arg:nnn {#2}{#1}{#3}
5485   \group_end:
5486 }
5487
5488 \cs_generate_variant:Nn \__stex_expr_arg_do:nnn {oon}

```

(End definition for `\stex_invoke_symbol`. This function is documented on page 120.)

## Argument Handling and Annotating

```

\stex_term_arg:nnnnn
\stex_term_arg:nnn
5489 % 1: argnum 2: argmode 3: precedence 4: argname 5: code

```

```

5490 \cs_new_protected:Nn \_stex_term_arg:nnnnn {
5491   \group_begin:
5492     \str_clear:N \l_stex_current_symbol_str
5493     \tl_clear:N \l_stex_current_term_tl
5494     \int_set:Nn \l_stex_notation_downprec { #3 }
5495     \bool_set_true:N \l_stex_allow_semantic_bool
5496     \_stex_term_arg:n {#2}{#1} {
5497       \tl_if_empty:nTF{#4} {
5498         #5
5499       } {
5500         \stex_annotation:nn{mml:arg={#4}}{#5}
5501       }
5502     }
5503   \group_end:
5504 }
5505
5506 \cs_new_protected:Nn \_stex_term_arg:nnn {
5507   \stex_annotation:nn{ shtml:arg={#2}, shtml:argmode={#1}}{
5508     \_stex_annotation_force_break:n {#3 }
5509   }
5510 }

```

(End definition for `\_stex_term_arg:nnnnn` and `\_stex_term_arg:nnn`. These functions are documented on page ??.)

### `\_stex_term_arg_aB:nnnnn`

```

5511 \tl_set:Nn \__stex_expr_do_aB_clist: {
5512   \seq_use:Nn \l_stex_aB_args_seq {
5513     \mathpunct{\comp{,}}
5514   }
5515 }
5516 \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5517
5518 \int_new:N \l__stex_expr_count_int
5519 \cs_new_protected:Nn \_stex_term_arg_aB:nnnnn {
5520   \tl_if_empty:nTF{#5} {
5521     \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{}
5522   } {
5523     \seq_clear:N \l_stex_aB_args_seq
5524     \int_zero:N \l__stex_expr_count_int
5525     \clist_map_inline:nn{#5} {
5526       \__stex_expr_aB_arg:nnnnn{##1}{#1}{#2}{#3}{#4}
5527     }
5528     \_stex_term_do_aB_clist:
5529   }
5530 }
5531
5532 % 1: code 2: argnum 3: argmode 4: precedence 5: argname
5533 \cs_new_protected:Npn \__stex_expr_aB_arg:nnnnn #1 {
5534   \int_incr:N \l__stex_expr_count_int
5535   \__stex_expr_is_varseq:nTF{#1} {
5536     \exp_after:wN \exp_after:wN \exp_after:wN
5537     \__stex_expr_assoc_seq:nnnnnnn
5538     \exp_after:wN

```

```

5539 \__stex_expr_gobble:nnnnnnnn #1 \__stex_expr_end:
5540 }
5541 \__stex_expr_is_seqmap:nTF{#1} {
5542   \exp_args:NNe \use:nn \__stex_expr_do_seqmap:nnnnnn {\tl_tail:n{#1}}
5543 }
5544   \__stex_expr_aB_simple_arg:nnnnn{#1}
5545 }
5546 }
5547 }
5548
5549 \cs_new:Npn \__stex_expr_gobble:nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8 #9 \__stex_expr_end: {
5550   {#2} #3 {#6}
5551 }
5552
5553 \cs_new_protected:Nn \__stex_expr_aB_simple_arg:nnnnn{
5554   \seq_put_right:Nx \l_stex_aB_args_seq {
5555     \stex_term_arg:nnnnn{#2}\int_use:N\l__stex_expr_count_int}{#3}{#4}{#5}{#6}
5556     \exp_not:n{
5557       \tl_set_eq:NN \stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5558       #1
5559     }
5560   }
5561 }
5562 }
5563
5564 \cs_new_protected:Nn \stex_is_sequentialized:n {
5565   \group_begin: #1 \group_end:
5566 }

```

Conditionals: Is the argument a sequence variable or a \seqmap?

```

5567 \prg_new_conditional:Nnn \__stex_expr_is_varseq:n {TF} {
5568   \int_compare:nNnTF {\tl_count:n{#1}} = 1 {
5569     \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_head:n{#1}}
5570     \stex_invoke_variable:nnnnnnN {
5571       \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No\tl_item:n{#1}{8}}
5572         \stex_invoke_sequence:
5573           \prg_return_true:\prg_return_false:
5574     } \prg_return_false:
5575   } \prg_return_false:
5576 }
5577
5578 \prg_new_conditional:Nnn \__stex_expr_is_seqmap:n {TF} {
5579   \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
5580     \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\seqmap}
5581       \prg_return_true:\prg_return_false:
5582   } \prg_return_false:
5583 }

```

Sequence variable:

```

5584 % 1: name 2: arity 3: clist 4: argnum 5: argmode 6: precedence 7: argname
5585 \cs_new_protected:Nn \__stex_expr_assoc_seq:nnnnnnn {
5586   \group_begin:
5587     \seq_clear:N \l_stex_aB_args_seq
5588     \__stex_expr_assoc_make_seq:nnn{#1}{#3}{#2}

```

```

5589 \exp_args:NNe \use:nn \group_end: {
5590   \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5591     \_stex_is_sequentialized:n{
5592       \_stex_term_arg:nnnn{#4\int_use:N\l_stex_expr_count_int}{#5}{#6}{#7} {
5593         \bool_set_true:N \l_stex_allow_semantic_bool
5594         \str_set:Nn \exp_not:N \l_stex_current_symbol_str
5595           {\l_stex_current_symbol_str}
5596         \tl_if_empty:NF \l_stex_current_term_tl {
5597           \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5598             \exp_args:No \exp_not:n \l_stex_current_term_tl
5599           }
5600         }
5601       \stex_annotation:nn{
5602         shtml:term=OMV,
5603         shtml:head={#1},
5604         shtml:notationid={}
5605       }{
5606         \_stex_annotation_force_break:n{
5607           \_stex_term_do_aB_clist:
5608         }
5609       }
5610     }
5611   }
5612 }
5613 }
5614 }
5615
5616 % #1: name, #2: clist, #3:arity
5617 \cs_new_protected:Nn \__stex_expr_assoc_make_seq:nnn {
5618   \cs_if_exist:cTF{\l_stex_notation_#1__cs} {
5619     \cs_set_eq:Nc \l_stex_expr_cs {\l_stex_notation_#1__cs}
5620   }{
5621     \stex_do_default_notation:
5622     \cs_set_eq:NN \l_stex_expr_cs \l_stex_default_notation
5623   }
5624 \clist_map_inline:nn{#2} {
5625   \tl_if_eq:nnTF{##1}{\ellipses} {
5626     \seq_put_right:Nn \l_stex_aB_args_seq { ##1 }
5627   }{
5628     \int_compare:nNnTF {##3} = 1 {
5629       \tl_set:Nn \l_stex_expr_iarg_tl { ##1 }
5630     }{
5631       \tl_set:Nn \l_stex_expr_iarg_tl { ##1 }
5632     }
5633     \seq_put_right:Nx \l_stex_aB_args_seq {
5634       \group_begin:
5635         \exp_not:n {
5636           \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5637           \def\comp{\varcomp}
5638           \str_set:Nn \l_stex_current_symbol_str{#1}
5639         }
5640         \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
5641         \exp_after:wN \exp_after:wN \exp_after:wN {
5642           \exp_after:wN \l_stex_expr_cs \exp_after:wN \group_end: \l_stex_expr_iarg_tl

```

```

5643 }
5644 }
5645 }
5646 }
5647 }

\seqmap:

5648 % 1: fun 2: clist 3: argnum 4: argmode 5: precedence 6: argname
5649 \cs_new_protected:Nn \__stex_expr_do_seqmap:nnnnnn {
5650   \group_begin:
5651     \cs_set:Npn \l_tmpa_cs ##1 {#1}
5652     \seq_clear:N \l_stex_aB_args_seq
5653     \clist_map_inline:nn{#2} {
5654       \__stex_expr_is_varseq:nTF{##1} {
5655         \exp_after:wN
5656         \__stex_expr_varseq_in_map:nnnnnnn ##1
5657       }{
5658         \seq_put_right:Nn \l_stex_aB_args_seq {##1}
5659       }
5660     }
5661     \seq_clear:N \l__stex_expr_old_seq
5662     \seq_map_inline:Nn \l_stex_aB_args_seq {
5663       \tl_if_eq:nnTF{##1}{\ellipses} {
5664         \seq_put_right:Nn \l__stex_expr_old_seq {##1}
5665       }
5666       % TODO \stex_is_sequentialized:n
5667     }
5668     \exp_args:NNo \seq_put_right:Nn \l__stex_expr_old_seq {
5669       \l_tmpa_cs {##1}
5670     }
5671   }
5672 }
5673 \seq_set_eq:NN \l_stex_aB_args_seq \l__stex_expr_old_seq
5674 \exp_args:NNe \use:nn \group_end: {
5675   \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5676     \stex_is_sequentialized:n{
5677       \stex_term_arg:nnnn{\#4}\int_use:N\l__stex_expr_count_int}{\#4}{\#5}{\#6}{%
5678         \bool_set_true:N \l_stex_allow_semantic_bool
5679         \str_set:Nn \exp_not:N \l_stex_current_symbol_str
5680           {\l_stex_current_symbol_str}
5681         \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5682           \symuse{Metatheory?sequence~map}
5683           {\exp_args:No \exp_not:n \l_tmpa_tl}
5684           {\stex_term_do_aB_clist:}
5685         }
5686         \__stex_expr_do_headterm:nn{}{
5687           \stex_term_do_aB_clist:
5688         }
5689       }
5690     }
5691   }
5692 }
5693 }

```

```

5695 \cs_new_protected:Nn \__stex_expr_varseq_in_map:nnnnnnn {
5696   \__stex_expr_assoc_make_seq:nnn{#2}{#6}{#3}
5697 }

```

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

## Term HTML Annotations

```

\stex_term_oms_or_omv:nnn
\stex_term_oms:nnn
\stex_term_omv:nnn
5698 \cs_new_protected:Nn \stex_eat_exclamation_point: {
5699   \peek_charcode_remove:NT ! {
5700     \stex_eat_exclamation_point:
5701   }
5702 }
5703
5704 \bool_new:N \stex_in_invisible_html_bool
5705 \bool_set_false:N \stex_in_invisible_html_bool
5706 \stex_if_html_backend:TF {
5707   % 1: variant 2: intent 3: code
5708   \cs_new_protected:Nn \stex_term_oms:nnn {
5709     \tl_if_empty:NTF \l_stex_current_term_tl {
5710       \stex_annotation:nn{
5711         shtml:term=OMID,
5712         shtml:head={\l_stex_current_symbol_str},
5713         shtml:notationid={#1},
5714       }{
5715         \stex_annotation_force_break:n{#3}
5716       }
5717     }{
5718       \__stex_expr_do_headterm:nn{#1}{#3}
5719     }
5720   }
5721   \cs_new_protected:Nn \stex_term_omv:nnn {
5722     \tl_if_empty:NTF \l_stex_current_term_tl {
5723       \stex_annotation:nn{
5724         shtml:term=OMV,
5725         shtml:head={\l_stex_current_symbol_str},
5726         shtml:notationid={#1}
5727       }{
5728         \stex_annotation_force_break:n{#3}
5729       }
5730     }{
5731       \__stex_expr_do_headterm:nn{#1}{#3}
5732     }
5733   }
5734   \cs_new_protected:Nn \__stex_expr_do_headterm:nn {
5735     \bool_if:NTF \stex_in_invisible_html_bool {
5736       {\bool_set_true:N \l_stex_allow_semantic_bool
5737         \ensuremath{\l_stex_current_term_tl}
5738       }
5739     }{
5740       \stex_annotation:nn{
5741         shtml:term=complex,
5742         shtml:head={\l_stex_current_symbol_str},

```

```

5743     shtml:notationid={#1}
5744   }{
5745     \_stex_annotation_force_break:n{
5746       \stex_annotation_invisible:nn{shtml:headterm={}){
5747         {\bool_set_true:N \l_stex_allow_semantic_bool
5748           \ensuremath{\l_stex_current_term_tl}}
5749         }
5750       }
5751     }
5752     #2
5753   }
5754 }
5755 }
5756 }{
5757   \cs_new_protected:Nn \_stex_term_oms:nnn {#3}
5758   \cs_new_protected:Nn \_stex_term_omv:nnn {#3}
5759   \cs_new_protected:Nn \__stex_expr_do_headterm:nn { #2 }
5760 }
5761 \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn

(End 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

```

5762 \stex_if_html_backend:TF {
5763   \cs_new_protected:Nn \_stex_term_oma:nnn {
5764     \tl_if_empty:NTF \l_stex_current_term_tl {
5765       \stex_annotation:nn{
5766         shtml:term=OMA,
5767         shtml:head={\l_stex_current_symbol_str},
5768         shtml:notationid={#1}
5769       }{
5770         \_stex_annotation_force_break:n{#3}
5771       }
5772     }{
5773       \stex_annotation:nn{
5774         shtml:term=OMA,
5775         shtml:head={\l_stex_current_symbol_str},
5776         shtml:notationid={#1}
5777       }{
5778         \_stex_annotation_force_break:n{
5779           \stex_annotation_invisible:nn{shtml:headterm={}){
5780             {\bool_set_true:N \l_stex_allow_semantic_bool
5781               \l_stex_current_term_tl}
5782             }
5783           #3
5784         }
5785       }
5786     }
5787   }{
5788     \cs_new_protected:Nn \_stex_term_oma:nnn {#3}
5789   }
5790 }
```

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

```

\_stex_term_omb:nnn
5791 \stex_if_html_backend:TF {
5792   \cs_new_protected:Nn \_stex_term_omb:nnn {
5793     \tl_if_empty:NTF \l_stex_current_term_tl {
5794       \stex_annotation:nn{
5795         shtml:term=OMBIND,
5796         shtml:head={\l_stex_current_symbol_str},
5797         shtml:notationid={#1}
5798       }{
5799         \_stex_annotation_force_break:n{#3}
5800       }
5801     }{
5802       \stex_annotation:nn{
5803         shtml:term=OMBIND,
5804         shtml:head={\l_stex_current_symbol_str},
5805         shtml:notationid={#1}
5806       }{
5807         \_stex_annotation_force_break:n{
5808           \stex_annotation_invisible:nn{shtml:headterm={}{{}
5809             \bool_set_true:N \l_stex_allow_semantic_bool
5810             \l_stex_current_term_tl
5811           }}}
5812         #3
5813       }
5814     }
5815   }
5816 }
5817 }{
5818   \cs_new_protected:Nn \_stex_term_omb:nnn { #3 }
5819 }

```

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

## Automated Bracketing

```

\infprec
\neginfprec
5820 \tl_const:Nx \infprec {\int_use:N \c_max_int}
5821 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}

(End definition for \infprec and \neginfprec. These functions are documented on page 80.)

```

```

\dobrackets
5822 \int_new:N \l_stex_notation_downprec
5823 \int_set:Nn \l_stex_notation_downprec \infprec
5824 \tl_set:Nn \l_stex_expr_left_bracket_str (
5825 \tl_set:Nn \l_stex_expr_right_bracket_str )
5826 \bool_new:N \l_stex_brackets_dones_bool
5827
5828 \cs_new_protected:Nn \_stex_maybe_brackets:nn {
5829   \bool_if:NTF \l_stex_brackets_dones_bool {
5830     \bool_set_false:N \l_stex_brackets_dones_bool
5831     #2
5832   } {
5833     \stex_debug:nn{brackets}{#1>\int_eval:n \l_stex_notation_downprec?}

```

```

5834   \int_compare:nNnTF { #1 } > \l_stex_notation_downprec {
5835     \%bool_if:NTF \l_stex_inpararray_bool { #2 }{
5836       \dobrackets {
5837         #2
5838       }
5839     %}
5840   }{
5841     #2
5842   }
5843 }
5844 }
5845
5846 \%RequirePackage{scalerel}
5847 \cs_new_protected:Npn \dobrackets #1 {
5848   \%ThisStyle{\if D\m@switch
5849     \% \exp_args:Nnx \use:nn
5850     { \exp_after:wN \left\lvert \l_stex_expr_left_bracket_str #1 }
5851     { \exp_not:N\right\rvert \l_stex_expr_right_bracket_str }
5852   \% \else
5853   \group_begin:
5854   \%stex_pseudogroup_with:nn{\l_stex_brackets_dones_bool\l_stex_notation_downprec} {
5855     \bool_set_true:N \l_stex_brackets_dones_bool
5856     \%int_set:Nn \l_stex_notation_downprec \infprec
5857     \mathopen{\cs_if_exist:NT\l_stex_current_symbol_str\comp
5858       \l_stex_expr_left_bracket_str
5859     }
5860     #1
5861   \group_end:%
5862   \mathclose{\cs_if_exist:NT\l_stex_current_symbol_str\comp
5863     \l_stex_expr_right_bracket_str
5864   }
5865   \%fi}
5866 }

```

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

### \withbrackets

```

5867 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
5868   \%stex_pseudogroup_with:nn{\l_stex_expr_left_bracket_str\l_stex_expr_right_bracket_str} {
5869     \tl_set:Nn \l_stex_expr_left_bracket_str { #1 }
5870     \tl_set:Nn \l_stex_expr_right_bracket_str { #2 }
5871     #3
5872   }
5873 }

```

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

### \dowithbrackets

```

5874 \cs_new_protected:Npn \dowithbrackets #1 #2 #3 {
5875   \withbrackets{#1}{#2}{\dobrackets{#3}}
5876 }

```

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

## Symname and Variants

```

\symname
  \sn 5877 \def\maincomp{\comp}
  \sns 5878
\Symname 5879 \stex_keys_define:nnnn{symname}{
  \Sn 5880   \tl_clear:N \l_stex_key_pre_tl
  \Sns 5881   \tl_clear:N \l_stex_key_post_tl
  \symref 5882   \%tl_clear:N \l_stex_key_proot_tl
  \sr 5883 }{
\varref 5884   pre .tl_set:N = \l_stex_key_pre_tl ,
\varname 5885   post .tl_set:N = \l_stex_key_post_tl ,
  \Varname 5886   root .code:n = {}%.tl_set:N = \l_stex_key_root_tl
  5887 }{}}

5888 \NewDocumentCommand \symref { O{} m m } {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_symbol:n{#2}
  \__stex_expr_do_ref:nNn{#3}\symrefemph@uri\_stex_term_oms:nnn
  5894 }
  \let\sr\symref
  5896
\NewDocumentCommand \symname { O{} m } {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_symbol:n{#2}
  \__stex_expr_do_ref:nNn{
    \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
  }\symrefemph@uri\_stex_term_oms:nnn
  5904 }
  \let\sn\symname
  5906 \protected\def\sns{\symname[post=s]}
  5907
\NewDocumentCommand \Symname { O{} m } {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_symbol:n{#2}
  \__stex_expr_do_ref:nNn{
    \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
  }\symrefemph@uri\_stex_term_oms:nnn
  5915 }
  \cs_new_protected:Nn \_stex_capitalize:n {
  5917   \uppercase{#1}
  5918 }
  \let\Sn\Symname
  5920 \protected\def\Sns{\Symname[post=s]}
  5921
\cs_new:Npn \_stex_split_slash: #1/#2/#3\_stex_args_end: {
  5923   \tl_if_empty:nTF{#3}{
    \#2
  }{
    \_stex_split_slash: #2 / #3 \_stex_args_end:
  }
  5927 }
```

```

5928 }
5929
5930 \NewDocumentCommand \varref { O{} m m} {
5931   \group_begin:
5932   \stex_keys_set:nn{symname}{#1}
5933   \stex_get_var:n{#2}
5934   \__stex_expr_do_ref:nNn{#3}\varemp@uri{
5935     \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5936     \def\comp{\_varcomp}
5937     \stex_term_omv:nnn
5938   }
5939 }
5940
5941 \NewDocumentCommand \varname { O{} m } {
5942   \group_begin:
5943   \stex_keys_set:nn{symname}{#1}
5944   \stex_get_var:n{#2}
5945   \__stex_expr_do_ref:nNn{
5946     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
5947   }\varemp@uri{
5948     \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5949     \def\comp{\_varcomp}
5950     \stex_term_omv:nnn
5951   }
5952 }
5953
5954 \NewDocumentCommand \Varname { O{} m } {
5955   \group_begin:
5956   \stex_keys_set:nn{symname}{#1}
5957   \stex_get_var:n{#2}
5958   \__stex_expr_do_ref:nNn{
5959     \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
5960   }\varemp@uri{
5961     \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5962     \def\comp{\_varcomp}
5963     \stex_term_omv:nnn
5964   }
5965 }
5966
5967
5968 \cs_new_protected:Nn \__stex_expr_do_ref:nNn {
5969   \stex_if_html_backend:T{\ifvmode\indent\fi}
5970   \bool_if:NTF \l_stex_allow_semantic_bool{
5971     \str_set:Nx\l_stex_current_symbol_str
5972       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
5973     \str_if_in:NnT \l_stex_get_symbol_name_str / {
5974       \str_set:Nx \l_stex_get_symbol_name_str {
5975         \exp_after:wN \l_stex_split_slash: \l_stex_get_symbol_name_str
5976         /\l_stex_args_end:
5977       }
5978     }
5979     \tl_clear:N \l_stex_current_term_tl
5980     \def\comp{\_comp}
5981     \let\compemph@uri#2

```

```

5982     #3{}{}{\comp{#1}}
5983   }{
5984     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
5985   }
5986   \group_end:
5987 }

```

(End definition for `\symname` and others. These functions are documented on page 78.)

## Highlighting

```

5988 <@=stex_notationcomps>
      \comp
\compemph@uri
      \compemph
      \defemph
\defemph@uri
      \symrefemph
\symrefemph@uri
      \varemph
\varemph@uri
      \varemph
\compemph@uri
      \do_comp:nNn {\comp}\compemph@uri
      \do_comp:nNn {\varcomp}\varemph@uri
      \do_comp:nNn {\definiendum}\defemph@uri
      \comp
      \compemph{#1}
      \compemph{#1}
      \defemph{#1}

```

```

6030 \cs_new_protected:Npn \defemph #1 {
6031     \ifmmode\else\expandafter\textbf\fi{#1}
6032 }
6033
6034 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
6035     \symrefemph{#1}
6036 }
6037
6038 \cs_new_protected:Npn \symrefemph #1 {
6039     \emph{#1}
6040 }
6041
6042 \cs_new_protected:Npn \varemph@uri #1 #2 {
6043     \varemph{#1}
6044 }
6045
6046 \cs_new_protected:Npn \varemph #1 {
6047     #1
6048 }

```

(End definition for `\comp` and others. These functions are documented on page 79.)

## 13.9 Mathematical Structures

```
6049 ⟨@@=stex_structures⟩
```

`\this`

```

6050 \cs_new_protected:Npn \stex_current_this: {
6051     { \bool_set_true:N \l_stex_allow_semantic_bool
6052         \tl_if_empty:NTF \l_stex_current_this_tl {{}}{
6053             \str_set:Nx \l_stex_current_symbol_str {
6054                 \l_stex_current_module_str ? \l__stex_structures_name_str
6055             }
6056             \maincomp{\l_stex_current_this_tl}
6057         }
6058     }
6059 }
6060 \let \this \stex_current_this:

```

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

`mathstructure (env.)`

```

6061 \stex_new_stylable_env:nnnnnnn {mathstructure}{m 0{}>{
6062     \__stex_structures_begin:nn{#1}{#2}
6063     \stex_smsmode_do:
6064 }{
6065     \stex_structural_feature_module_end:
6066     \__stex_structures_do_externals:
6067 }{}{}{}
6068
6069 \stex_keys_define:nnnn{mathstructure}{
6070     \tl_clear:N \l_stex_current_this_tl
6071     \str_clear:N \l__stex_structures_name_str
6072 }{

```

```

6073   this .tl_set:N = \l_stex_current_this_tl ,
6074   unknown .code:n  = {
6075     \str_set_eq:NN \l__stex_structures_name_str \l_keys_key_str
6076   }
6077 }{{
6078
6079 \cs_new_protected:Nn \__stex_structures_begin:nn {
6080   \stex_keys_set:nn {mathstructure}{#2}
6081   \str_if_empty:NT \l__stex_structures_name_str {
6082     \str_set:Nn \l__stex_structures_name_str {#1}
6083   }
6084   \def\comp{\_comp}
6085
6086   \exp_args:Nne \use:nn { \stex_module_add_symbol:nnnnnnnN }
6087   { {#1}{\l__stex_structures_name_str}{0}{}{defed}{%
6088     \l_stex_current_module_str / \l__stex_structures_name_str-module
6089   }%
6090   {} \stex_invoke_structure:
6091   \str_set:Nx \l_stex_mroname_str {#1}
6092   \stex_execute_in_module:xf {
6093     \seq_clear:c{\l_stex_structure_macros_\l_stex_current_module_str / \l__stex_structures_na
6094     \seq_put_right:cn{\l_stex_structure_macros_\l_stex_current_module_str / \l__stex_structur
6095   }
6096   \exp_args:No \stex_structural_feature_module:nn
6097   {\l__stex_structures_name_str}{structure}
6098 }
6099
6100 \stex_sms_allow_env:n{mathstructure}
6101 \stex_deactivate_macro:Nn \mathstructure {module-environments}
6102 \stex_every_module:n {\stex_reactivate_macro:N \mathstructure}
6103
6104 \cs_new_protected:Nn \__stex_structures_do_exernals: {
6105   \tl_set:Nn \l__stex_structures_replace_this_tl {####1}
6106   \exp_args:No \stex_iterate_symbols:nnf{\g_stex_last_feature_str}{%
6107     \__stex_structures_external_decl:nnnn{##5}{##4}{##3}{##8}
6108   }
6109 }
6110
6111 \cs_new_protected:Nn \__stex_structures_external_decl:nnnn {
6112   \%stex_debug:nn{structure}{%
6113     % Generating~external~declaration~\l__stex_structures_name_str/#3-in-
6114     % \l_stex_current_module_str^{^J}
6115     % \tl_to_str:n{#1}^{^J}\tl_to_str:n{#2}^{^J}\tl_to_str:n{#4}
6116   }%
6117   \%tl_set:Nn \l_stex_get_symbol_args_tl {#1}
6118   \%exp_args:Nnx \use:nn { \stex_module_add_symbol:nnnnnnnN } {
6119     % {}{\l__stex_structures_name_str/#3}{\int_eval:n{#2 + 1}}
6120     % {ii\tl_if_empty:nF{#1}{\stex_map_args:N \__stex_structures_shift_argls:nn}}
6121     % {defed}{typed}
6122     %}{#4}\stex_invoke_outer_field:
6123   }
6124
6125 \cs_new:Nn \__stex_structures_shift_argls:nn {
6126   \int_eval:n{#1+1}#2

```

```

6127 }

\stex_get_mathstructure:n
6128 \cs_new_protected:Nn \stex_get_mathstructure:n {
6129   \stex_get_mathstructure:n{#1}
6130   \str_if_empty:NT \l_stex_get_structure_module_str {
6131     \msg_error:nnn{\stex}{error/unknownstructure}{#1}
6132   }
6133 }
6134 \cs_new_protected:Nn \stex_get_mathstructure:n {
6135   \str_clear:N \l_stex_get_structure_module_str
6136   \stex_get_symbol:n{#1}
6137   \str_if_empty:NF \l_stex_get_symbol_name_str {
6138     \exp_args:No \tl_if_eq:NNT \l_stex_get_symbol_invoke_cs \stex_invoke_structure: {
6139       \str_set_eq:NN \l_stex_get_structure_module_str \l_stex_get_symbol_type_tl
6140     }
6141   }
6142 }

```

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

`extstructure (env.)`

```

6143 \stex_new_stylistable_env:nnnnnnn {extstructure}{m 0{} m} {
6144   \seq_clear:N \l__stex_structures_imports_seq
6145   \clist_map_inline:nn{#3}{%
6146     \stex_get_mathstructure:n{##1}
6147     \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6148       \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{####1}}{%
6149         \seq_put_right:Nn \l__stex_structures_imports_seq{####1}
6150       }
6151     }
6152     \stex_execute_in_module:x{
6153       \seq_put_right:cn{\l_stex_structure_macros_ \l_stex_get_structure_module_str _seq}{#1}
6154     }
6155   }
6156   \__stex_structures_begin:nn{#1}{#2}
6157   \seq_map_inline:Nn \l__stex_structures_imports_seq{%
6158     \stex_if_do_html:T {
6159       \hbox{\stex_annotation_invisible:nn
6160         {shtml:import={##1}} {}}
6161     }
6162     \stex_module_add_morphism:nonn
6163     {}{##1}{import}{}
6164     \stex_execute_in_module:x{
6165       \stex_activate_module:n{##1}
6166     }
6167   }
6168   \stex_smsmode_do:
6169 }{
6170   \stex_structural_feature_module_end:
6171   \__stex_structures_do_externals:
6172 }{}{}{}
6173 \stex_sms_allow_env:n{extstructure}

```

```

6175 \stex_deactivate_macro:Nn \extstructure {module-environments}
6176 \stex_every_module:n {
6177   \stex_reactivate_macro:N \extstructure
6178 }
6179
6180 \cs_new:Nn \__stex_structures_extend_structure_i:NnnnnnnN {
6181   \exp_not:n{#1{#2}{#3}{#4}{#5}{defed}}{\l__stex_structures_extmod_str,\#7}\exp_not:n{{#8}{#9}}
6182 }
6183 \cs_new_protected:Nn \__stex_structures_extend_structure:nn {
6184   \stex_debug:nn{ext}{Extending-#1-by-#2}
6185   \str_set:Nn \l__stex_structures_extmod_str{#2}
6186   \tl_set:cx{#1} {
6187     \exp_after:wN \exp_after:wN \exp_after:wN
6188     \__stex_structures_extend_structure_i:NnnnnnnN \cs:w #1 \cs_end:
6189   }
6190 }
6191
6192 \stex_new_stylable_env:nnnnnnn {extstructure*}{m} {
6193   \__stex_structures_new_extstruct_name:
6194   \seq_clear:N \l__stex_structures_imports_seq
6195   \stex_get_mathstructure:n{#1}
6196
6197   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6198     \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6199       \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6200     }
6201   }
6202
6203 \stex_execute_in_module:x{
6204   \seq_map_inline:cn{\l_stex_structure_macros_\l_stex_get_structure_module_str _seq} {
6205     \exp_not:N \tl_if_exist:cT{####1} {
6206       \__stex_structures_extend_structure:nn{####1}{\l_stex_current_module_str/\l__stex_st
6207     }
6208   }
6209 }
6210
6211 \exp_args:No \__stex_structures_begin:nn\l__stex_structures_exstruct_name_str{}
6212
6213 \seq_map_inline:Nn \l__stex_structures_imports_seq {
6214   \stex_if_do_html:T {
6215     \stex_annotation_invisible:nn
6216     {shtml:import= {##1}} {}
6217   }
6218   \stex_module_add_morphism:nonn
6219   {}{##1}{import} {}
6220   \stex_execute_in_module:x{
6221     \stex_activate_module:n{##1}
6222   }
6223 }
6224
6225 \stex_smsmode_do:
6226 }{
6227 \prop_map_inline:cn{
6228   c_stex_module_ \l_stex_current_module_str _symbols_prop

```

```

6229   }{
6230     \__stex_structures_check_def:nnnnnnnn ##2
6231   }
6232 \stex_structural_feature_module_end:
6233 \__stex_structures_do_externals:
6234 }{}{}{}
6235
6236 \stex_sms_allow_env:n{extstructure*}
6237 \exp_after:wN \stex_deactivate_macro:Nn
6238   \cs:w extstructure*\cs_end: {module~environments}
6239 \stex_every_module:n {
6240   \exp_after:wN \stex_reactivate_macro:N \cs:w extstructure*\cs_end:
6241 }
6242
6243 \cs_new_protected:Nn \__stex_structures_check_def:nnnnnnnn {
6244   \tl_if_empty:nT{#5} {
6245     \msg_error:nnnn{stex}{error/needsdefiniens}{#2}{extstructure*}
6246   }
6247 }
6248
6249 \stex_every_module:n{ \str_set:Nn \l__stex_structures_extname_count 0}
6250
6251 \cs_new_protected:Nn \__stex_structures_new_extstruct_name: {
6252   \stex_do_up_to_module:n {
6253     \str_set:Nx \l__stex_structures_extname_count {\int_eval:n{\l__stex_structures_extname_c
6254   }
6255   \str_set:Nx \l__stex_structures_exstruct_name_str {EXTSTRUCT_\l__stex_structures_extname_c
6256 }
6257

```

Invoking structures:

```

6258 \cs_new_protected:Nn \stex_invoke_structure: {
6259   \tl_set:Nn \l__stex_structures_set_comp_tl {\__stex_structures_set_thiscomp:}
6260   \__stex_structures_invoke_top:n {}
6261 }
6262
6263 \cs_new_protected:Nn \__stex_structures_invoke_top:n {
6264   \stex_debug:nn{structure} {
6265     invoking~structure~{\l_stex_current_type_tl}<\tl_to_str:n{#1}>
6266   }
6267   \peek_charcode:NTF [ {
6268     \__stex_structures_merge:nw{#1}
6269   }{
6270     \__stex_structures_invocation_type:n {#1}
6271     \tl_set:Nn \l__stex_structures_this_tl {}
6272     \peek_charcode_remove:NTF ! {
6273       \peek_charcode:NTF [ {
6274         \__stex_structures_maybe_notation:w
6275       }{
6276         \__stex_structures_maybe_notation:w []
6277       }
6278     }{
6279       \__stex_structures_invoke_this:n
6280     }

```

```

6281     }
6282 }
6283
6284 \cs_new_protected:Npn \__stex_structures_merge:nw #1 [ #2 ] {
6285     \exp_args:Ne \stex_if_starts_with:nNTF {\tl_to_str:n{#2}}{comp} {
6286         \__stex_structures_set_customcomp: #2 \__stex_structures_end:
6287         \__stex_structures_invoke_top:n{#1}
6288     }{
6289         \exp_args:Ne \stex_if_starts_with:nNTF {\tl_to_str:n{#2}}{this} {
6290             \__stex_structures_set_thisnotation: #2 \__stex_structures_end:
6291             \__stex_structures_invoke_top:n{#1}
6292         }{
6293             \tl_if_empty:nTF{#1} {
6294                 \__stex_structures_invoke_top:n{#2}
6295             }{
6296                 \tl_if_empty:nTF{#2} {
6297                     \__stex_structures_invoke_top:n{#1}
6298                 }{
6299                     \__stex_structures_invoke_top:n{#1, #2}
6300                 }
6301             }
6302         }
6303     }
6304 }
6305
6306 \cs_new_protected:Npn \__stex_structures_set_thisnotation: this= #1 \__stex_structures_end:
6307     \tl_set:Nn \l_stex_return_notation_tl { \comp{#1} }
6308     \tl_set:Nn \l__stex_structures_set_comp_tl {}
6309 }
6310
6311 \cs_new_protected:Npn \__stex_structures_set_customcomp: comp= #1 \__stex_structures_end: {
6312     \tl_set:Nn \l__stex_structures_set_comp_tl {
6313         \__stex_structures_set_custom_comp:n{#1}
6314     }
6315     \tl_set:Nn \l_stex_return_notation_tl { \comp{} }
6316 }

```

The structure type:

```

6317 \cs_new_protected:Nn \__stex_structures_invokation_type:n {
6318     \__stex_structures_do_assign_list:n{#1}
6319     \clist_if_empty:NNTF \l__stex_structures_fields_clist {
6320         \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6321             = 1 {
6322                 \tl_set:Nx \l__stex_structures_current_type_tl {
6323                     \exp_args:No \exp_not:n \l_stex_current_redo_tl
6324                     \stex_term_oms_or_omv:nnn{}{}{}
6325                 }
6326             }{
6327                 \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6328             }
6329         }{
6330             \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6331                 = 1 {
6332                     \__stex_structures_make_type:n {}
6333                 }

```

```

6334     \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6335   }
6336 }
6337 }
6338
6339 \cs_new_protected:Nn \__stex_structures_do_assign_list:n {
6340   \clist_clear:N \l_stex_structures_fields_clist
6341   \tl_if_empty:nF {#1} {
6342     \keyval_parse:NNn\TODO\__stex_structures_do_assign:nn{#1}
6343   }
6344 }
6345
6346 \cs_new_protected:Nn \__stex_structures_do_assign:nn {
6347   \clist_put_right:Nn \l_stex_structures_fields_clist {{#1}{#2}}
6348 }
6349
6350 \cs_new_protected:Nn \__stex_structures_make_type:n {
6351   \tl_if_empty:nTF{#1} {
6352     \seq_clear:N \l_tmpa_seq
6353   }{
6354     \seq_set_split:Nnn \l_tmpa_seq ,{#1}
6355     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
6356     \seq_reverse:N \l_tmpa_seq
6357   }
6358   \tl_set:Nx \l_stex_structures_current_type_tl {
6359     \symuse{Metatheory?module~type~merge}{}
6360   {
6361     \exp_args:No \exp_not:n \l_stex_current_redo_tl
6362     \stex_term_oms_or_omv:nnn{}{}{}
6363   }
6364   \seq_map_function:NN \l_tmpa_seq \__stex_structures_make_mod:n
6365   \clist_if_empty:NF \l_stex_structures_fields_clist {
6366     ,\symuse{Metatheory?anonymous~record}{}
6367     \exp_args:Ne \tl_tail:n{
6368       \clist_map_function:NN \l_stex_structures_fields_clist \__stex_structures_make_
6369     }
6370   }
6371 }
6372 }
6373 }
6374 }
6375
6376 \cs_new:Nn \__stex_structures_make_mod:n {
6377   ,\symuse{Metatheory?module~type}{}
6378   \stex_annotation:nn{shtml:term=OMMOD,shtml:head={#1}}{}
6379 }
6380 }
6381
6382 \cs_new:Nn \__stex_structures_make_oml:n {
6383   \__stex_structures_make_oml:nn #1
6384 }
6385 \cs_new:Nn \__stex_structures_make_oml:nn {
6386   ,\stex_annotation:nn{
6387     shtml:term=OML,

```

```

6388     shtml:head={#1}
6389   }{
6390     \_stex_annotation_force_break:n{
6391       \stex_annotation:nn{shtml:definiens={}}{\exp_not:n{#2!}}
6392     }
6393   }
6394 }

```

Insert the structure type as a term:

```

6395 \cs_new:Nn \_stex_structures_current_type: {
6396   \%exp_args:No \exp_not:n \l_stex_current_redo_tl
6397   \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6398     \exp_args:No\exp_not:n\l_stex_structures_current_type_tl
6399   }
6400   \_stex_term_oms_or_omv:nnn{}{}{}
6401 }

```

The structure type itself:

```

6402 \cs_new_protected:Npn \_stex_structures_maybe_notation:w [ #1 ] {
6403   \tl_set_eq:NN \l_stex_current_term_tl \l_stex_structures_current_type_tl
6404   \cs_if_exist:cTF{l_stex_notation_}\l_stex_current_symbol_str _#1_cs} {
6405     \use:c{l_stex_notation_}\l_stex_current_symbol_str _#1_cs}\group_end:
6406   }{
6407     \_stex_structures_make_prop:
6408     \_stex_structures_make_prop_assign:
6409     \_stex_structures_present_i:w [#1]
6410   }
6411 }
6412
6413 \cs_new_protected:Nn \_stex_structures_present: {
6414   \peek_charcode:NTF [ {
6415     \_stex_structures_present_i:w
6416   }{
6417     \_stex_structures_present:nn{}{}
6418   }
6419 }
6420
6421 \cs_new_protected:Npn \_stex_structures_present_i:w [#1] {
6422   \int_compare:nNnTF{\clist_count:n{#1}} = 1 {
6423     \_stex_structures_present:nn{}{#1}
6424   }{
6425     \peek_charcode:NTF [ {
6426       \_stex_structures_present_ii:nw[#1]
6427     }{
6428       \_stex_structures_present:nn{#1}{}
6429     }
6430   }
6431 }
6432
6433 %First: clist, second:notation-id
6434 \cs_new_protected:Npn \_stex_structures_present_ii:nw #1 [#2] {
6435   \_stex_structures_present:nn{#1}{#2}
6436 }
6437
6438 \cs_new_protected:Nn \_stex_structures_present:nn {

```

```

6439 \clist_clear:N \l__stex_structures_clist
6440 \tl_if_empty:nTF{#1}{
6441   \cs_set:Npn \l__stex_structures_cs ##1 ##2 ##3 {
6442     \tl_if_empty:nF{##2}{
6443       \__stex_structures_present_entry:nn {##1}{##3}
6444     }
6445   }
6446 }{
6447   \cs_set:Npn \l__stex_structures_cs ##1 ##2 ##3 {
6448     \exp_args:Ne \clist_if_in:nnT{\tl_to_str:n{#1}}{##1} {
6449       \__stex_structures_present_entry:nn {##1}{##3}
6450     }
6451   }
6452 }
6453 \prop_map_inline:Nn \l__stex_structures_prop {
6454   \l__stex_structures_cs {##1} ##2
6455 }
6456 \stex_term_oms_or_omv:nnn{}{}{
6457   \exp_args:Nno \use:n{
6458     \bool_set_true:N \l_stex_allow_semantic_bool
6459     \symuse{Metatheory?mathematical-structure}[#2]
6460   }{\l__stex_structures_clist}
6461 }\group_end:
6462 }
6463
6464 \cs_new_protected:Nn \__stex_structures_present_entry:nn {
6465   \seq_if_in:NnTF \l__stex_structures_assigned_seq {#1} {
6466     \clist_put_right:Nn \l__stex_structures_clist {#2!}
6467   }
6468   \exp_args:NNe \clist_put_right:Nn \l__stex_structures_clist {
6469     \stex_next_symbol:n {
6470       \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
6471       \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
6472         \exp_args:No \exp_not:n \l__stex_structures_this_tl
6473       }
6474       \exp_not:n {
6475         \tl_set_eq:NN \this \l__stex_structures_this_tl
6476       }
6477       \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6478         \exp_args:No \exp_not:n \l_stex_return_notation_tl
6479       }
6480     }
6481     \exp_not:n{#2!}
6482   }
6483 }
6484 }
6485
6486 \cs_new_protected:Npn \_thiscomp #1 #2 {
6487   {\tl_set:cn{this}{{}}#1{#2}\c_math_subscript_token{
6488     \group_begin:
6489     \bool_set_true:N \l_stex_allow_semantic_bool
6490     \l__stex_structures_this_tl
6491     \group_end:
6492   }

```

```

6493     }
6494 }
6495
6496 \cs_new_protected:Nn \__stex_structures_set_thiscomp: {
6497     \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_thiscomp {
6498         \edef\maincomp {\_thiscomp\comp}}
6499     }
6500 }
6501
6502 \cs_new_protected:Nn \__stex_structures_set_custom_comp:n {
6503     \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
6504         \cs_set_protected:Npx \_customthiscomp ##1 {
6505             \group_begin:
6506                 \bool_set_true:N \l_stex_allow_semantic_bool
6507                 \exp_not:n{
6508                     \cs_set:Npn \l__stex_structures_comp_cs ##1 {
6509                         #1
6510                     }
6511                     \def\maincomp
6512                         }{\comp}
6513                     \exp_not:N \l__stex_structures_comp_cs{\comp{##1}}
6514                 \group_end:
6515             }
6516             \def\maincomp {\_customthiscomp}
6517         }
6518 }
6519
this (of type structure):
6520
6521 \cs_new_protected:Nn \__stex_structures_invoke_this:n {
6522     \peek_charcode_remove:NTF ! {
6523         \exp_args:Nne\use:nn{
6524             \group_end:\symuse{Metatheory?of~type}[invisible]{#1}
6525         }{
6526             {\__stex_structures_current_type:}
6527         }
6528     }{
6529         \__stex_structures_invoke_maybe_field:nn{#1}
6530     }
6531 }
6532
6533 \cs_new_protected:Nn \__stex_structures_invoke_maybe_field:nn {
6534     \__stex_structures_make_prop:
6535     \__stex_structures_set_this:n{#1}
6536     \tl_if_empty:nTF{#2}{
6537         \__stex_structures_make_prop_assign:
6538         \__stex_structures_present:
6539     }{
6540         \__stex_structures_invoke_field:n{#2}
6541     }
6542 }
6543
6544 \cs_new_protected:Nn \__stex_structures_set_this:n {
6545     \tl_if_empty:nTF{#1}{
```

```

6546   \%tl_put_right:Nn \l_stex_current_redo_tl {
6547     % \tl_clear:N \l_stex_structures_this_tl
6548   %}
6549 }{
6550   \tl_set:Nx \l_stex_structures_this_tl {{
6551     \bool_set_true:N \l_stex_allow_semantic_bool
6552     \tl_set:Nn \exp_not:N \this {
6553       \exp_args:No \exp_not:n \this
6554     }
6555     \exp_not:n{#1}
6556   }}
6557   \tl_set_eq:NN \this \l_stex_structures_this_tl
6558   % \l_stex_return_notation_tl
6559 }
6560 }
6561
6562 \cs_new_protected:Nn \__stex_structures_get_field_name:n {
6563   \str_set:Nx \l_stex_structures_field_name_str {
6564     \exp_args:Nne \use:n {\exp_after:wN \use_i:nn \use:n}
6565     {\prop_item:Nn \l_stex_structures_prop {#1}}
6566   }
6567   \str_if_empty:NT \l_stex_structures_field_name_str {
6568     \str_set:Nn \l_stex_structures_field_name_str {#1}
6569   }
6570 }
6571
6572 \cs_new_protected:Nn \__stex_structures_invoke_field:n {
6573   \prop_if_in:NnTF \l_stex_structures_prop {#1} {
6574     \__stex_structures_get_field_name:n{#1}
6575     \tl_clear:N \l_stex_structures_more_nextsymbol_tl
6576     \%exp_args:NNe \seq_if_in:Nnf \l_stex_structures_assigned_seq {\tl_to_str:n{#1}}{
6577       \tl_set:Nx \l_stex_structures_more_nextsymbol_tl {
6578         \tl_set:Nn \exp_not:N \l_stex_structures_this_tl {
6579           \exp_args:No \exp_not:n \l_stex_structures_this_tl
6580         }
6581         \exp_not:n {
6582           \tl_set_eq:NN \this \l_stex_structures_this_tl
6583         }
6584         \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6585           \exp_args:No \exp_not:n \l_stex_return_notation_tl
6586         }
6587         \exp_args:No \exp_not:n \l_stex_structures_set_comp_tl
6588       }
6589     }
6590   \exp_args:NNx \use:nn \group_end: {
6591     \__stex_next_symbol:n {
6592       \exp_args:No \exp_not:n \l_stex_structures_redo_tl
6593       \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6594         \symuse{Metatheory?record-field}{%
6595           \symuse{Metatheory?of~type}{%
6596             \exp_args:No \exp_not:n \l_stex_structures_this_tl
6597           }{ \__stex_structures_current_type: }
6598         }
6599         \stex_annotation:nn{shml:term=OML,shml:head={\l_stex_structures_field_name_str}}
```

```

6600         }
6601     }
6602     \exp_args:No \exp_not:n \l__stex_structures_more_nextsymbol_tl
6603   }
6604   \exp_not:N \use_ii:nn
6605   \prop_item:Nn \l__stex_structures_prop {#1}
6606 }
6607 }{
6608   \msg_error:nnn{stex}{error/unknownfield}{#1}
6609 }
6610 }
6611
6612 \cs_new_protected:Nn \__stex_structures_make_prop: {
6613   \prop_clear:N \l__stex_structures_prop
6614   \seq_clear:N \l__stex_structures_seq
6615   \seq_clear:N \l__stex_structures_assigned_seq
6616   \tl_clear:N \l__stex_structures_redo_tl
6617   \__stex_structures_prop_do_decls:
6618   \__stex_structures_prop_do_notations:
6619 }
6620
6621 \cs_new_protected:Nn \__stex_structures_make_prop_assign: {
6622   \clist_if_empty:NF \l__stex_structures_fields_clist {
6623     \clist_map_inline:Nn \l__stex_structures_fields_clist {
6624       \__stex_structures_make_prop_assign:nn ##1
6625     }
6626   }
6627 }
6628
6629 \cs_new_protected:Nn \__stex_structures_make_prop_assign:nn {
6630   \prop_if_in:NnTF \l__stex_structures_prop {#1} {
6631     \exp_args:NNe \seq_put_right:Nn \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}
6632     \exp_args:Nne \use:nn {\__stex_structures_make_prop_assign_replace:nnnn {#1}{#2}}
6633     {\prop_item:Nn \l__stex_structures_prop {#1}}
6634   }{
6635     \msg_error:nnn{stex}{error/unknownfieldass}{#1}
6636   }
6637 }
6638 \cs_new_protected:Nn \__stex_structures_make_prop_assign_replace:nnnn {
6639   \prop_put:Nnn \l__stex_structures_prop {#1}{##3}{#2}
6640   \tl_if_empty:nF{#3} {
6641     \tl_set:cn{#1}{ #2 }
6642     \tl_put_right:Nn \l__stex_structures_redo_tl {
6643       \tl_set:cn{#1}{ #2 }
6644     }
6645   }
6646 }
6647
6648 \cs_new_protected:Nn \__stex_structures_prop_do_decls: {
6649   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_type_tl {
6650     \tl_if_empty:nTF{##2} {
6651       \__stex_structures_do_decl_nomacro:nnnnnnnn{##3}
6652     }{
6653       \__stex_structures_do_decl:nnnnnnnn{##2}

```

```

6654     }
6655     {##1}{##3}{##4}{##5}{##6}{##7}{##8}{##9}
6656   }
6657 }
6658
6659 \cs_new_protected:Nn \__stex_structures_do_decl_nomacro:nnnnnnnnn {
6660   \prop_if_in:NnF \l__stex_structures_prop {#1} {
6661     \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
6662     \prop_put:Nnn \l__stex_structures_prop {#1} {
6663       {}{
6664         \stex_invoke_symbol:nnnnnnnnN
6665         {#2}
6666         {#3}
6667         {#4}{#5}{#6}{#7}{#8}{#9}
6668       }
6669     }
6670   }
6671 }
6672
6673 \cs_new_protected:Nn \__stex_structures_do_decl:nnnnnnnnn {
6674   \prop_if_in:NnF \l__stex_structures_prop {#1} {
6675     \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
6676     \prop_put:Nnn \l__stex_structures_prop {#1} {
6677       {}{
6678         \stex_invoke_symbol:nnnnnnnnN
6679         {#2}
6680         {#3}
6681         {#4}{#5}{#6}{#7}{#8}{#9}
6682       }
6683     }
6684   }
6685   \%tl_set:cn{#1}{
6686   % \stex_invoke_symbol:nnnnnnnnN
6687   % {#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}
6688   %}
6689   \%tl_put_right:Nn \l__stex_structures_redo_tl {
6690   % \tl_set:cn{#1} {
6691   % \stex_invoke_symbol:nnnnnnnnN
6692   % {#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}
6693   % }
6694   %}
6695 }
6696
6697 \cs_new_protected:Nn \__stex_structures_prop_do_notations: {
6698   \exp_args:No \stex_iterate_notations:nn\l_stex_current_type_tl{
6699     \exp_args:NNe \seq_if_in:NnT \l__stex_structures_seq {\tl_to_str:n{##1}}{
6700       \tl_put_right:Nn \l__stex_structures_redo_tl {
6701         \cs_if_exist:cF{\l_stex_notation_##1 _##2_cs} {
6702           \tl_set:cn{\l_stex_notation_##1 _##2_cs}{##4}
6703         }
6704         \cs_if_exist:cF{\l_stex_notation_##1 __cs} {
6705           \tl_set:cn{\l_stex_notation_##1 __cs}{##4}
6706         }
6707     }

```

```

6708     \cs_if_exist:cF{l_stex_notation_##1 _##2_cs}{
6709         \tl_set:cn{l_stex_notation_##1 _##2_cs}{##4}
6710     }
6711     \cs_if_exist:cF{l_stex_notation_##1 __cs}{
6712         \tl_set:cn{l_stex_notation_##1 __cs}{##4}
6713     }
6714     \tl_if_empty:nF{##5}{
6715         \tl_put_right:Nn \l__stex_structures_redo_tl {
6716             \cs_if_exist:cF{l_stex_notation_##1 _op_##2_cs} {
6717                 \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
6718             }
6719             \cs_if_exist:cF{l_stex_notation_##1 _op__cs} {
6720                 \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
6721             }
6722         }
6723         \cs_if_exist:cF{l_stex_notation_##1 _op_##2_cs} {
6724             \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
6725         }
6726         \cs_if_exist:cF{l_stex_notation_##1 _op__cs} {
6727             \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
6728         }
6729     }
6730 }
6731 }
6732 }

\usestructure

6733 \cs_new_protected:Npn \usestructure #1 {
6734     \stex_get_mathstructure:n{ #1 }
6735     \seq_clear:N \l__stex_structures_imports_seq
6736     \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6737         \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6738             \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6739         }
6740     }
6741     \seq_map_inline:Nn \l__stex_structures_imports_seq {
6742         \stex_if_do_html:T {
6743             \hbox{\stex_annotation_invisible:nn
6744                 {shtml:usemodule=##1} {}}
6745         }
6746         \stex_activate_module:n {##1}
6747     }
6748 }

```

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

## 13.10 Statements

```

6749 <@=stex_statements>
6750
6751 \stex_keys_define:nnnn{statement}{
6752     \str_clear:N \l_stex_key_name_str
6753     \str_clear:N \l_stex_key_mroname_str

```

```

6754   \clist_clear:N \l_stex_key_for_clist
6755   \str_clear:N \l_stex_key_args_str
6756   \tl_clear:N \l_stex_key_type_tl
6757   \tl_clear:N \l_stex_key_def_tl
6758   \tl_clear:N \l_stex_key_return_tl
6759   \clist_clear:N \l_stex_key_argtypes_clist
6760 }{
6761   name .str_set:N = \l_stex_key_name_str ,
6762   for .clist_set:N = \l_stex_key_for_clist ,
6763   macro .str_set:N = \l_stex_key_macroname_str ,
6764   % start .str_set:N = \l_stex_key_title_str , % TODO remove
6765   type .tl_set:N = \l_stex_key_type_tl ,
6766   judgment .code:n = {},
6767   from .code:n= {}, % TODO remove
6768   to .code:n={} % TODO remove
6769 }{id,title,style,symargs}

\stex_new_statement:nn
6770 \cs_new_protected:Npn \stex_do_for_list: {
6771   \seq_clear:N \l_stex_fors_seq
6772   \clist_map_inline:Nn \l_stex_key_for_clist {
6773     \exp_args:N\stex_get_symbol:n{\tl_to_str:n{##1}}
6774     \seq_put_right:Nx \l_stex_fors_seq
6775       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6776   }
6777 }
6778
6779 \cs_new_protected:Nn \__stex_statements_setup:nn {
6780   \str_if_empty:NF \l_stex_key_macroname_str {
6781     \str_if_empty:NT \l_stex_key_name_str {
6782       \str_set_eq:NN \l_stex_key_name_str \l_stex_key_macroname_str
6783     }
6784   }
6785   \stex_do_for_list:
6786   \str_if_empty:NF \l_stex_key_name_str {
6787     \__stex_statements_force_id:
6788     \seq_put_right:Nx \l_stex_fors_seq {
6789       \l_stex_current_module_str ? \l_stex_key_name_str
6790     }
6791     \str_set_eq:NN \l_stex_macroname_str \l_stex_key_macroname_str
6792     \str_set:Nn \l_stex_key_role_str {#2}
6793     \stex_symdecl_do:
6794     \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN} {
6795       {\l_stex_key_macroname_str}{\l_stex_key_name_str}
6796       {int_use:N \l_stex_get_symbol_arity_int}
6797       {\l_stex_get_symbol_args_tl}
6798       {#1}{}{}\stex_invoke_symbol:
6799     }
6800     \stex_if_do_html:T \stex_symdecl_html:
6801   }
6802   \str_clear:N \l__stex_statements_uri_str
6803   \str_if_empty:NTF \l_stex_key_name_str {
6804     \stex_debug:nn{statement}{no~name}
6805     \int_compare:nNnTF {\seq_count:N \l_stex_fors_seq} = 1 {

```

```

6806     \str_set:Nx \l__stex_statements_uri_str {\seq_item:Nn \l_stex_fors_seq 1}
6807     \stex_debug:nn{statement}{for:~\l__stex_statements_uri_str}
6808 }{
6809     \stex_debug:nn{statement}{no~for}
6810 }
6811 }{
6812     \str_set:Nx \l__stex_statements_uri_str {\l_stex_current_module_str ? \l_stex_key_name_s
6813     \stex_debug:nnf{statement}{name:~\l__stex_statements_uri_str}
6814 }
6815 }
6816
6817 \cs_new:Nn \__stex_statements_html_keyvals:nn {
6818     shtml:#1={},
6819     shtml:inline={#2},
6820     \seq_if_empty:NF \l_stex_fors_seq {},
6821     shtml:fors=\seq_use:Nn \l_stex_fors_seq ,
6822 }
6823 \str_if_empty:NF \l_stex_key_id_str {},
6824     shtml:id=\stex_uri_use:N \l_stex_current_doc_uri ? \l_stex_key_id_str
6825 }
6826 \clist_if_empty:NF \l_stex_key_style_clist {},
6827     shtml:styles=\l_stex_key_style_clist
6828 }
6829 }
6830
6831 \cs_new_protected:Nn \stex_new_statement:nnn {
6832     \stex_new_stylable_env:nnnnnnn {#1}{0{}}
6833     \stex_keys_set:nn{statement}{##1}
6834     #3
6835
6836     \stex_if_smsmode:F {
6837         \exp_args:Nne \begin{stex_annotation_env} {
6838             \__stex_statements_html_keyvals:nn{#1}{false}
6839         }
6840         \tl_set_eq:NN \thistitle \l_stex_key_title_tl
6841         \str_set_eq:NN \thisname \l_stex_key_name_str
6842         \clist_set_eq:NN \thisfor \l_stex_key_for_str
6843         \stex_if_html_backend:TF {
6844             \noindent
6845             \stex_annotation:nn{shtml:statementtitle={}}{\stex_annotation_force_break:n\l_stex_key_}
6846         }
6847         \stex_style_apply:
6848     }
6849     \stex_do_id:
6850     \stex_smsmode_do:
6851 }{
6852     \stex_if_smsmode:F {
6853         \stex_if_html_backend:F \stex_style_apply:
6854         \end{stex_annotation_env}
6855     }
6856 }{}{}{s}
6857 \stex_sms_allow_env:n{s#1}
6858
6859 \tl_if_empty:nF{#2}{
```

```

6860 \exp_after:wN \NewDocumentCommand \cs:w inline#2\cs_end: { 0{} m}{
6861   \group_begin:
6862     \stex_keys_set:nn{statement}{##1}
6863     #3
6864     \stex_do_id:
6865     \stex_if_smsmode:F{
6866       \exp_args:Ne \stex_annotation:nn{\_stex_statements_html_keyvals:nn{#1}{true}}{
6867         \stex_annotation_force_break:n{##2}
6868       }
6869     }
6870   \group_end:
6871   \stex_smsmode_do:
6872 }
6873 \exp_after:wN \stex_sms_allow_escape:N\cs:w inline#2\cs_end:
6874 }
6875 }

6876 \cs_new_protected:Nn \_stex_statements_setup_def: {
6877   \stex_if_smsmode:F{
6878     \seq_map_inline:Nn \l_stex_fors_seq {
6879       \stex_ref_new_sym_target:n{##1}
6880     }
6881   }
6882 \stex_reactivate_macro:N \definiendum
6883 \stex_reactivate_macro:N \defnotation
6884 \stex_reactivate_macro:N \defname
6885 \stex_reactivate_macro:N \Defname
6886 \stex_reactivate_macro:N \varbind
6887 }

6888 }

6889 \cs_new_protected:Nn \_stex_statements_force_id: {
6890   \str_if_empty:NT \l_stex_key_id_str {
6891     \stex_ref_new_id:n{}
6892     \str_set_eq:NN \l_stex_key_id_str \l_stex_refs_str
6893   }
6894 }

6895 }

6896 \stex_new_statement:nnn{definition}{def}{
6897   \_stex_statements_force_id:
6898   \_stex_statements_setup:nn{}{}
6899   \_stex_statements_setup_def:
6900   \stex_reactivate_macro:N \definiens
6901 }

6902 }

6903 \stex_new_statement:nnn{assertion}{ass}{
6904   \_stex_statements_setup:nn{}{assertion}
6905   \stex_if_smsmode:F{
6906     \seq_map_inline:Nn \l_stex_fors_seq {
6907       \stex_ref_new_sym_target:n{##1}
6908     }
6909   }
6910   \stex_reactivate_macro:N \varbind
6911   \stex_reactivate_macro:N \conclusion
6912   \stex_reactivate_macro:N \premise
6913   \stex_reactivate_macro:N \definiendum

```

```

6914   \stex_reactivate_macro:N \defnotation
6915   \stex_reactivate_macro:N \defname
6916   \stex_reactivate_macro:N \Defname
6917 }
6918 \stex_new_statement:nnn{example}{ex}{\__stex_statements_setup:nn{}{example}}
6919 \stex_new_statement:nnn{paragraph}{}{
6920   \clist_if_in:NnTF \l_stex_key_style_clist {symdoc} {
6921     \__stex_statements_force_id:
6922     \__stex_statements_setup:nn{}{}
6923     \__stex_statements_setup_def:
6924   }{
6925     \__stex_statements_setup:nn{}{}
6926   }
6927 }

```

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

#### definiendum

```

6928 \cs_new_protected:Nn \__stex_statements_do_deref:nn {
6929   \stex_if_html_backend:T{\ifvmode\indent\fi}
6930   \group_begin:
6931   \stex_get_symbol:n#1
6932   \bool_if:NTF \l_stex_allow_semantic_bool{
6933     \str_set:Nx \l_stex_current_symbol_str
6934       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6935     \str_if_in:NnT \l_stex_get_symbol_name_str / {
6936       \str_set:Nx \l_stex_get_symbol_name_str {
6937         \exp_after:wN \stex_split_slash: \l_stex_get_symbol_name_str
6938         /\_stex_args_end:
6939       }
6940     }
6941     \exp_args:No \stex_ref_new_sym_target:n \l_stex_current_symbol_str
6942     \def\comp{\_defcomp}
6943     \stex_annotation:nn{shtml:definiendum=\l_stex_current_symbol_str}{\comp{#2}}
6944   }{
6945     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
6946   }
6947   \group_end:
6948 }
6949
6950 \NewDocumentCommand \defnotation{ m } {
6951   \stex_next_symbol:n { \def\comp{\_defcomp}}#1
6952 }
6953 \stex_deactivate_macro:Nn \defnotation {definition~environments}
6954
6955 \NewDocumentCommand \definiendum { O{} m m } {
6956   \stex_keys_set:nn{symname}{ #1 }
6957   \__stex_statements_do_deref:nn{#2}{#3}
6958 }
6959 \stex_deactivate_macro:Nn \definiendum {definition~environments}
6960
6961 \NewDocumentCommand \defname { O{} m } {
6962   \stex_keys_set:nn{symname}{#1}
6963   \__stex_statements_do_deref:nn{#2}{

```

```

6964     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
6965   }
6966 }
6967 \stex_deactivate_macro:Nn \definename {definition-environments}
6968
6969 \NewDocumentCommand \Definename { O{} m } {
6970   \stex_keys_set:nn{symname}{#1}
6971   \l_stex_statements_do_deref:nn{#2}{
6972     \l_stex_key_pre_tl\exp_after:wN\l_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
6973   }
6974 }
6975 \stex_deactivate_macro:Nn \Definename {definition-environments}
6976
6977
6978 \NewDocumentCommand \definiens { O{} m }{
6979   \group_begin:
6980   \str_clear:N \l_stex_get_symbol_name_str
6981   \tl_if_empty:nF {#1} {
6982     \stex_get_symbol:n { #1 }
6983     \str_set:Nx \l_stex_statements_uri_str
6984       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6985   }
6986   \str_if_empty:NT \l_stex_statements_uri_str {
6987     \msg_error:nn{stex}{error/definiensfor}
6988   }
6989   \stex_debug:nn{definiens}{Checking~\l_stex_statements_uri_str}
6990
6991   \exp_args:No \_stex_add_definiens:nn \l_stex_statements_uri_str{#2}
6992
6993   \group_end:
6994   \stex_smsmode_do:
6995 }
6996 \stex_deactivate_macro:Nn \definiens {definition-environments}
6997 \stex_sms_allow_escape:N \definiens
6998
6999 \cs_new_protected:Nn \_stex_add_definiens:nn {
7000   \exp_args:Nno \stex_str_if_starts_with:nnT{#1} \l_stex_current_module_str {
7001     \prop_map_inline:cN{c_stex_module_\l_stex_current_module_str_symbols_prop} {
7002       \stex_debug:nn{definiens}{#1 == \l_stex_current_module_str?##1}
7003       \str_if_eq:noT {#1} {\l_stex_current_module_str?##1} {
7004         \prop_map_break:n{\_stex_add_definiens_inner:nnnnnnnn ##2}
7005       }
7006     }
7007   }
7008   \stex_if_smsmode:F{
7009     \stex_annotation:nn{ \shtml:definiens={#1}}{
7010       #2 \%_stex_annotation_force_break:n{ #2 }
7011     }
7012   }
7013 }
7014
7015 \cs_new_protected:Nn \_stex_add_definiens_inner:nnnnnnnn {
7016   \stex_debug:nn{definiens}{Adding~definiens~to~\l_stex_current_module_str?#2}
7017   \prop_gput:cnn{c_stex_module_\l_stex_current_module_str_symbols_prop}

```

```

7018     {#2}{#1}{#2}{#3}{#4}{defed}{#6}{#7}{#8}}
7019 }
7020
7021 \NewDocumentCommand \varbind {m} {
7022     \clist_map_inline:nn {#1} {
7023         \stex_get_var:n {##1}
7024         \stex_if_do_html:T {
7025             \stex_annotate_invisible:nn {shtml:bind=\l_stex_get_symbol_name_str{}}
7026         }
7027     }
7028 }
7029 \stex_deactivate_macro:Nn \varbind {definition-or-assertion-environments}
7030
7031 \NewDocumentCommand \conclusion { O{} m} {
7032     \group_begin:
7033     \str_clear:N \l_stex_get_symbol_name_str
7034     \tl_if_empty:nF {#1} {
7035         \stex_get_symbol:n { #1 }
7036         \str_set:Nx \l__stex_statements_uri_str
7037             {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7038     }
7039     \str_if_empty:NT \l__stex_statements_uri_str {
7040         \msg_error:nn{stex}{error/conclusionfor}
7041     }
7042     \stex_annotate:nn{ shtml:conclusion=\l__stex_statements_uri_str{}{
7043         #2 \%_stex_annotate_force_break:n{ #2 }
7044     }
7045     \group_end:
7046 }
7047 \stex_deactivate_macro:Nn \conclusion {assertion-environments}
7048
7049 \NewDocumentCommand \premise {O{} m} {
7050     \tl_if_empty:nF {#1} {
7051         \stex_debug:nn{Here:}{Variable~#1}
7052         \exp_args:NNx\exp_args:Nnx\vardef{v#1}{#1}
7053     }
7054     \stex_annotate:nn{shtml:premise={#1}}{#2}
7055 }
7056 \stex_deactivate_macro:Nn \premise {assertion-environments}

```

(End definition for *definiendum*. This function is documented on page 89.)

### 13.11 Proofs

We first define some keys for the *sproof* environment.

```

7057 <@@=stex_proof>
7058 \stex_keys_define:nnnn{ spf }{
7059 \tl_clear:N \l_stex_key_for_clist
7060 \tl_clear:N \l_stex_key_from_tl
7061 \tl_set_eq:NN \l_stex_key_proofend_tl \__stex_proof_proof_box_tl
7062 \tl_clear:N \l_stex_key_continues_tl
7063 \tl_clear:N \l_stex_key_term_tl
7064 \tl_clear:N \l_stex_key_functions_tl

```

```

7065 \tl_clear:N \l_stex_key_method_tl
7066   \bool_set_false:N \l_stex_key_hide_bool
7067 }{
7068   for .clist_set:N = \l_stex_key_for_clist ,
7069   from .tl_set:N = \l_stex_key_from_tl ,
7070   proofend .tl_set:N = \l_stex_key_proofend_tl,
7071   continues .tl_set:N = \l_stex_key_continues_tl,
7072   functions .tl_set:N = \l_stex_key_functions_tl,
7073   term .tl_set:N = \l_stex_key_term_tl,
7074   method .tl_set:N = \l_stex_key_method_tl,
7075   hide .bool_set:N = \l_stex_key_hide_bool
7076 }{id,style,title}
7077
7078 \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.

```

7079 \intarray_new:Nn\l__stex_proof_counter_intarray{50}
7080 \cs_new_protected:Npn \__stex_proof_insert_number: {
7081   \int_set:Nn \l_tmpa_int {1}
7082   \bool_while_do:nn {
7083     \int_compare_p:nNn {
7084       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7085     } > 0
7086   }{
7087     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .
7088     \int_incr:N \l_tmpa_int
7089   }
7090 }
7091 \cs_new_protected:Nn \__stex_proof_number_as_string:N {
7092   \str_clear:N #1
7093   \int_set:Nn \l_tmpa_int {1}
7094   \bool_while_do:nn {
7095     \int_compare_p:nNn {
7096       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7097     } > 0
7098   }{
7099     \str_put_right:Nx #1 {\intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .}
7100     \int_incr:N \l_tmpa_int
7101   }
7102 }
7103
7104 \cs_new_protected:Npn \__stex_proof_inc_counter: {
7105   \int_set:Nn \l_tmpa_int {1}
7106   \bool_while_do:nn {
7107     \int_compare_p:nNn {
7108       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7109     } > 0

```

```

7110 }{
7111   \int_incr:N \l_tmpa_int
7112 }
7113 \int_compare:nNnF \l_tmpa_int = 1 {
7114   \int_decr:N \l_tmpa_int
7115 }
7116 \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int {
7117   \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int + 1
7118 }
7119 }
7120
7121 \cs_new_protected:Npn \__stex_proof_add_counter: {
7122   \int_set:Nn \l_tmpa_int {1}
7123   \bool_while_do:nn {
7124     \int_compare_p:nNn {
7125       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7126     } > 0
7127   }{
7128     \int_incr:N \l_tmpa_int
7129   }
7130   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 1 }
7131 }
7132
7133 \cs_new_protected:Npn \__stex_proof_remove_counter: {
7134   \int_set:Nn \l_tmpa_int {1}
7135   \bool_while_do:nn {
7136     \int_compare_p:nNn {
7137       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7138     } > 0
7139   }{
7140     \int_incr:N \l_tmpa_int
7141   }
7142   \int_decr:N \l_tmpa_int
7143   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 0 }
7144 }

spfsketch
7145 \newenvironment{spfsketchenv}{}{}
7146 \stex_new_stylable_cmd:nnnn{spfsketch}{0}{m}{\par
7147   \begin{spfsketchenv}
7148   \stex_keys_set:nn{spf}{#1}
7149   \stex_do_for_list:
7150   \stex_do_id:
7151   \exp_args:Ne \stex_annotation:nn{
7152     shtml:proofsketch={
7153       \seq_if_empty:NF \l_stex_fors_seq {
7154         \seq_use:Nn \l_stex_fors_seq ,
7155       }
7156     }
7157   }{
7158     \stex_style_apply:
7159     #2
7160   }
7161 \end{spfsketchenv}

```

```

7162 }{
7163   \noindent\emph{\spfsketchenvautorefname :}-
7164 }

```

(End 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

```

7165 \tl_set:Nn \__stex_proof_box_tl {
7166   \ltx@ifpackageloaded{amssymb}{$\square$}-
7167   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
7168 }
7169 }
7170
7171 \tl_set:Nn \sproofend {
7172   \tl_if_empty:NF \l_stex_key_proofend_tl {
7173     \hfil\null\nobreak\hfill\l_stex_key_proofend_tl\par\smallskip
7174   }
7175 }

```

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

#### `\stexcommentfont`

```

7176 \cs_new_protected:Npn \stexcommentfont {
7177   \small\itshape
7178 }

```

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

#### `sproof (env.)`

```

7179 \cs_new_protected:Nn \__stex_proof_start_list:n {
7180   \begin{list}{}{
7181     \setlength\topsep{0pt}
7182     \setlength\parsep{0pt}
7183     \setlength\rightmargin{0pt}
7184   }\item[#1]
7185 }
7186 \cs_new_protected:Nn \__stex_proof_end_list: {
7187   \end{list}
7188 }
7189
7190 \cs_new_protected:Nn \__stex_proof_html: {
7191   \stex_annotation_invisible:n{\hbox{
7192     \tl_if_empty:NF \l_stex_key_term_tl {
7193       $\stex_annotation:nn{shtml:proofterm={}}{\l_stex_key_term_tl}$-
7194     }
7195     \tl_if_empty:NF \l_stex_key_method_tl {
7196       \stex_annotation:nn{shtml:proofmethod={}}{\l_stex_key_method_tl}-
7197     }
7198   }}
7199 }
7200
7201 \cs_new_protected:Nn \__stex_proof_html_env:n {
7202   \exp_args:Nne \begin{stex_annotation_env}{-
7203     shtml:#1=-

```

```

7204     \seq_if_empty:NF \l_stex_fors_seq {
7205         \seq_use:Nn \l_stex_fors_seq ,
7206     }
7207     \bool_if:NT \l_stex_key_hide_bool {
7208         shtml:proofhide=true
7209     }
7210 }
7211 }
7212 \__stex_proof_html:
7213 }
7214
7215 \bool_set_false:N \l__stex_proof_in_spfblock_bool
7216 \cs_new_protected:Nn \__stex_proof_begin_proof:nn {\par
7217     \intarray_gzero:N \l_stex_proof_counter_intarray
7218     \intarray_gset:Nnn \l_stex_proof_counter_intarray 1 1
7219     \stex_keys_set:nn{spfsteps}{#1}
7220     \stex_do_for_list:
7221     \stex_if_do_html:T {
7222         \__stex_proof_html_env:n{proof}
7223     }
7224     \seq_map_inline:Nn \l_stex_fors_seq {
7225         \stex_debug:nn{definiens}{Adding-definiens-to-##1}
7226         \stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7227     }
7228     \stex_style_apply:
7229     \stex_do_id:
7230     \stex_reactivate_macro:N \subproof
7231     \stex_reactivate_macro:N \spfstep
7232     \stex_reactivate_macro:N \conclude
7233     \stex_reactivate_macro:N \assumption
7234     \stex_reactivate_macro:N \eqstep
7235     \stex_reactivate_macro:N \yield
7236     \stex_reactivate_macro:N \spfblock
7237     \stex_reactivate_macro:N \spfjust
7238     \stex_annotation:nn{shtml:prooftitle={}}{#2}
7239     \stex_if_do_html:T{
7240         \begin{stex_annotation_env}{shtml:proofbody={}}
7241     }
7242 }
7243 \stex_new_stylable_env:nnnnnnn{proof}{0{} m}{
7244     \__stex_proof_begin_proof:nn{#1}{#2}
7245     \bool_set_true:N \l__stex_proof_in_spfblock_bool \__stex_proof_start_list:n{}
7246     \group_begin:\stexcommentfont
7247 }{
7248     \stex_style_apply:
7249     \stex_if_do_html:T{\end{stex_annotation_env}\end{stex_annotation_env}}
7250 }{
7251     \emph{\sproofautorefname :}-
7252 }{
7253     \sproofend
7254 }{s}
7255 \AddToHook{env/sproof/end}{
7256     \bool_if:NT \l__stex_proof_in_spfblock_bool {
7257         \group_end:\__stex_proof_end_list:

```

```

7258     }
7259 }
7260
7261 \stex_new_stylable_env:nnnnnnn{proof*}{0{}}
7262   \__stex_proof_begin_proof:nn{#1}{}
7263   \bool_set_false:N\l__stex_proof_in_spfblock_bool
7264 }{
7265   \stex_style_apply:
7266   \stex_if_do_html:T{\end{stex_annotation_env}\end{stex_annotation_env}}
7267 }{
7268   \emph{Proof:}-
7269 }{
7270   \sproofend
7271 }{s}

subproof (env.)
7272 \str_set_eq:NN \subproofautorefname \spfstepautorefname
7273 \stex_new_stylable_env:nnnnnnn{subproof}{s 0{} m}{\par
7274   \stex_keys_set:nn{spf}{#2}
7275   \stex_do_for_list:
7276   \stex_if_do_html:T {
7277     \__stex_proof_html_env:n{subproof}
7278   }
7279   \seq_map_inline:Nn \l_stex_fors_seq {
7280     \stex_debug:nn{definiens}{Adding-definiens-to~##1}
7281     \stex_add_definiens:nn {##1}{\STEXinvisibl{proven}}
7282   }
7283
7284 \IfBooleanTF #1 {
7285   \stex_style_apply:
7286   \str_if_empty:NF \l_stex_key_id_str {
7287     \__stex_proof_number_as_string:N \@currentlabel
7288     \str_set:Nx \@currentHref{subproof.\@currentlabel}
7289     \stex_do_id:
7290   }
7291   \bool_set_false:N \l__stex_proof_in_spfblock_bool
7292   \stex_annotation:nn{shtml:prooftitle={}}{#3}
7293 }{
7294   \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7295     \str_if_empty:NF \l_stex_key_id_str {
7296       \__stex_proof_number_as_string:N \@currentlabel
7297       \str_set:Nx \@currentHref{subproof.\@currentlabel}
7298       \stex_do_id:
7299     }
7300     \__stex_proof_start_list:n\__stex_proof_insert_number:
7301       \stex_annotation:nn{shtml:prooftitle={}}{#3}
7302       \__stex_proof_add_counter:
7303       \stex_style_apply:
7304     }{
7305       \stex_annotation:nn{shtml:prooftitle={}}{#3}
7306       \stex_style_apply:
7307       \stex_do_id:
7308     }
7309 }

```

```

7310  \stex_if_do_html:T{
7311      \begin{stex_annotation_env}{shtml:proofbody={}}
7312  }
7313  \bool_if:NT \l__stex_proof_in_spfblock_bool {\group_begin:\stexcommentfont}
7314 }{
7315  \stex_style_apply:
7316  \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_inc_counter:
7317  \stex_if_do_html:T{\end{stex_annotation_env}}
7318  \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
7319  \stex_if_do_html:T{\end{stex_annotation_env}}
7320  \aftergroup \__stex_proof_inblock_restore:
7321 }{}{}{}
7322 \AddToHook{env/subproof/before}{
7323  \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7324 }
7325 \AddToHook{env/subproof/end}{
7326  \bool_if:NT \l__stex_proof_in_spfblock_bool {
7327      \group_end:\__stex_proof_remove_counter:
7328      \%__stex_proof_end_list:
7329  }
7330 }
7331 \stex_deactivate_macro:Nn \subproof {sproof~environments}
7332
7333 \cs_new_protected:Nn \__stex_proof_inblock_restore: {
7334  \bool_if:NT \l__stex_proof_in_spfblock_bool {
7335      \group_begin:\stexcommentfont
7336  }
7337 }

\spfstep
\conclude
7338
\assumption
7339 \stex_keys_define:nnnn { spfsteps } {
7340  \clist_clear:N \l_stex_key_for_clist
7341  \str_clear:N \l_stex_key_name_str
7342  \tl_clear:N \l_stex_key_method_tl
7343  \tl_clear:N \l_stex_key_term_tl
7344 }{
7345  for .clist_set:N = \l_stex_key_for_clist ,
7346  method .tl_set:N = \l_stex_key_method_tl,
7347  term .tl_set:N = \l_stex_key_term_tl,
7348  name .str_set_x:N = \l_stex_key_name_str
7349  % todo: style=inline
7350 }{id,style,title}
7351
7352 \newenvironment{spfstepenv}{
7353  \str_set_eq:NN \spfstepenvautorefname \spfstepautorefname
7354 }{}
7355
7356 \cs_new_protected:Nn \__stex_proof_step_html:nn {
7357  \stex_if_do_html:TF{
7358      \exp_args:Ne \stex_annotation:n{%
7359          shtml:spf#1=%
7360          \seq_if_empty:NF \l_stex_fors_seq {
7361              \seq_use:Nn \l_stex_fors_seq ,

```

```

7362     }
7363   }
7364   \str_if_empty:NF \l_stex_key_name_str {
7365     shtml:stepname={\l_stex_key_name_str}
7366   }
7367 }{
7368   \__stex_proof_html:
7369   #2
7370 }
7371 }{ #2 }
7372 }
7373
7374 \cs_new_protected:Nn \__stex_proof_make_step_macro:Nnnnn {
7375   \NewDocumentCommand #1 {s O{} +m} {
7376     \bool_if:NT \l_stex_proof_in_spfblock_bool \group_end:
7377     \stex_keys_set:nn{spfsteps}{##2}
7378     \str_if_empty:NF \l_stex_key_name_str {
7379       \stex_debug:nn{Here:}{Variable~\l_stex_key_name_str}
7380       \exp_args:NNx\exp_args:Nnx\vardef{v\l_stex_key_name_str}{\l_stex_key_name_str}
7381     }
7382
7383   \begin{spfstepenv}
7384     \str_if_empty:NF \l_stex_key_id_str {
7385       \__stex_proof_number_as_string:N \@currentlabel
7386       \str_set:Nx \@currentHref{spfstep.\@currentlabel}
7387       \stex_do_id:
7388     }
7389
7390     \bool_if:NTF \l_stex_proof_in_spfblock_bool {
7391       \IfBooleanTF ##1 {
7392         \__stex_proof_step_html:nn{#2}{##3}
7393       }{
7394         \__stex_proof_step_html:nn{#2}{\__stex_proof_start_list:n{#3} ##3 \__stex_proof_end_}
7395         #5
7396       }
7397       \end{spfstepenv}
7398       \group_begin:\stexcommentfont
7399     }{
7400       \__stex_proof_step_html:nn{#2}{##3}
7401       \end{spfstepenv}
7402     }
7403   }
7404   \stex_deactivate_macro:Nn #1 {sproof-environments}
7405 }
7406
7407 \__stex_proof_make_step_macro:Nnnnn \assumption {assumption} \__stex_proof_insert_number: {}
7408 \__stex_proof_make_step_macro:Nnnnn \conclude {conclusion} {$\Rightarrow$} {} {}
7409 \__stex_proof_make_step_macro:Nnnnn \spfstep {step} \__stex_proof_insert_number: {} \__stex_
7410
7411 \NewDocumentCommand \eqstep {s m} {
7412   \bool_if:NTF \l_stex_proof_in_spfblock_bool {
7413     \group_end:
7414     \IfBooleanTF #1 {
7415       \__stex_proof_step_html:nn{eqstep}{$= #2$}

```

```

7416   }{
7417     \__stex_proof_step_html:nn{eqstep}{\__stex_proof_start_list:n{$=$} $$ \__stex_proof_
7418   }
7419   \group_begin:\stexcommentfont
7420   }{
7421     \__stex_proof_step_html:nn{eqstep}{$= #2$}
7422   }
7423 }
7424 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
7425
7426 \NewDocumentCommand \yield {+m}{
7427   \stex_annotation:nn{shtml:proofterm={}}{ #1 }
7428 }
7429 \stex_deactivate_macro:Nn \yield {sproof~environments}
7430
7431 \NewDocumentEnvironment{spfblock}{}{
7432   \bool_set_false:N \l__stex_proof_in_spfblock_bool
7433 }{
7434   \aftergroup\__stex_proof_inblock_restore:
7435 }
7436 \stex_deactivate_macro:Nn \spfblock {sproof~environments}
7437 \AddToHook{env/spfblock/before}{
7438   \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7439 }
7440
7441 \newcommand\spfjust[1]{
7442   \stex_annotation:nn{spfjust={}}{ #1 }
7443 }
7445 \stex_deactivate_macro:Nn \spfjust {sproof~environments}

```

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

### 13.12 Metatheory

```

7446 <@=stex_meta>
7447 \group_begin:
7448   \cs_set:Npn \__stex_modules_persist_module: {}
7449   \cs_set:Npn \stex_check_term:n #1 {}
7450   \cs_set:Npn \_stex_sref_do_aux:n #1 { #1 }
7451   \bool_set_false:N \stex_html_do_output_bool
7452   \bool_set_false:N \c_stex_check_terms_bool
7453   \stex_uri_resolve:Nn \l_stex_current_ns_uri {http://mathhub.info/sTeX/meta}
7454   \stex_module_setup:n{Metatheory}
7455
7456   \symdef{of~type}[args=ii,invisible]{#1}
7457   \notation{of~type}[colon]{#1 \mathbin{\comp{}} #2}
7458
7459   \symdef{apply}[args=ia,prec=0;\infprec x\infprec]{#1\mathopen{\comp{}} #2 \mathclose{\comp{}}}
7460   \notation{apply}[lambda]{#1\lambda\argsep{#2}{\;}}
7461   \notation{apply}[infixop]{\argsep{#2}{\mathbin{#1}}}
7462   \notation{apply}[infixrel]{\argsep{#2}{\mathrel{#1}}}
7463
7464 % structures

```

```

7465 \symdef{module~type}[args=i,op=\mathtt{MOD}]
7466 {\mathopen{\comp{\mathtt{MOD}()}}\#1\mathclose{\comp{}}}}
7467 \symdef{module~type~merge}[args=a,op=\oplus]
7468 {\argsep{\#1}{\mathbin{\comp{\oplus}}}}}
7469 \symdef{anonymous~record}[args=a]
7470 {\mathopen{\comp{[]}}\#1\mathclose{\comp{[]}}}}
7471 \symdef{record~field}[args=2]{\#1\comp{.}\#2}
7472 \symdecl*{record~type}
7473
7474 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
7475 \notation{mathstruct}[angle,prec=nobrackets]
7476 {\mathopen{\comp{\langle}}\#1\mathclose{\comp{\rangle}}}
7477 \notation{mathstruct}[parens,prec=nobrackets]
7478 {\mathopen{\comp{()}}\#1\mathclose{\comp{}}}}
7479
7480 % sequences
7481 \symdef{ellipses}[ldots]{\ldots}
7482 \symdef{sequence-expression}[comma,args=a]{\#1}
7483 \symdef{sequence-type}[args=1]{\#1^{\comp{\ast}}}
7484 \symdef{sequence-map}[args=ia]{\comp{\mathrm{map}}\mathopen{\comp{()}}\#1\mathpunct{\comp{,}}\#2\mathclose{\comp{}}}}
7485
7486 }
7487
7488 \iffalse
7489 % binder (\forall, \Pi, \lambda etc.)
7490 \symdef{pibind}[name=dependent function type,prec=nobrackets,
7491 op=(\cdot\cdot\cdot;\cdot\cdot\cdot),args=Bi,assoc=pre]
7492 {\argmap{\#1}{\mathopen{\comp{()}}\#1\mathclose{\comp{()}}}{\mathbin{\comp{\cdot\cdot\cdot}}}\mathbin{\comp{\cdot\cdot\cdot}}\#2}
7493 \notation{pibind}[\forall]{\comp{\forall}\#1\mathpunct{\comp{.}}\#2}
7494 \notation{pibind}[\Pi]{\mathop{\comp{\prod}}\c_math_subscript_token{\#1}\#2}
7495
7496 \symdef{mapbind}[name=lambda,mapsto,prec=nobrackets,op=\mapsto,args=Bi,assoc=pre]
7497 {\#1\mathrel{\comp{\mapsto}}\#2}
7498 \notation{mapbind}[\lambda,prec=nobrackets,op=\lambda]
7499 {\comp{\lambda}\#1\mathpunct{\comp{.}}\#2}
7500
7501 \fi
7502 \symdecl{bind}[args=Bi,assoc=pre]
7503 \notation{bind}[dfunc,prec=nobrackets,op=(\cdot\cdot\cdot;\cdot\cdot\cdot,\cdot\cdot\cdot)]
7504 {\mathopen{\comp{()}}\#1\mathclose{\comp{()}}\mathbin{\comp{\cdot\cdot\cdot}}\mathbin{\comp{\cdot\cdot\cdot}}\#2}
7505 \notation{bind}[\forall]{\comp{\forall}\#1.\#2}
7506 \notation{bind}[\Pi]{\mathop{\comp{\prod}}\c_math_subscript_token{\#1}\#2}
7507
7508 \symdef{implicit-bind}[args=Bi,assoc=pre]{\mathopen{\comp{\{}}\#1\mathclose{\comp{\}}\c_math_}
7509
7510 \symdecl*{integer-literal}
7511 \notation{integer-literal}{\mathbb{Z}}
7512
7513 \symdecl*{ordinal}
7514 \notation{ordinal}{\mathtt{Ord}}
7515
7516 % propositions
7517 \symdef{prop}[name=proposition]{\mathtt{Prop}}
7518

```

```

7519 \symdef{judgment-holds}[args=i,role=judgment]{\comp\vdash\;#1}
7520
7521 % any object
7522 \symdef{object}{\mathit{Obj}}
7523
7524 % TODO DELETE
7525 \symdef{aseqdots}[args=a,prec=nobrackets]
7526 {#1\comp{,\ldots}}%{##1\comp,##2}
7527 \symdef{aseqfromto}[args=ai,prec=nobrackets]
7528 {#1\comp{,\ldots},#2}%{##1\comp,##2}
7529 \symdef{aseqfromtovia}[args=aai,prec=nobrackets]
7530 {#1\comp{,\ldots},#2\comp{,\ldots},#3}%{##1\comp,##2}
7531
7532
7533 \stex_close_module:
7534 \stex_uri_add_module:Nnn \l_stex_matatheory_uri \l_stex_current_ns_uri {Metatheory}
7535 \global \let \l_stex_matatheory_uri \l_stex_matatheory_uri
7536 \global \let \c_stex_default_matatheory \l_stex_matatheory_uri
7537 \group_end:

```

### 13.13 MMT Interfaces

```

7538 <@=todo>
7539 \cs_new_protected:Npn \MSC #1 {}

\MMTinclude
7540 \stex_new_stylable_cmd:nnnn[MMTinclude]{m}{
7541   \stex_annotation_invisible:nn{shml:import={#1}}{}
7542 }{}}
7543 \stex_deactivate_macro:Nn \MMTinclude {module-environments}
7544 \stex_every_module:n {\stex_reactivate_macro:N \MMTinclude}

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

```

```

\MMTrule
7545 \NewDocumentCommand \MMTrule {m m} {
7546   \tl_if_empty:nTF{#2}{\seq_clear:N \l_tmpa_seq}{%
7547     \seq_set_split:Nnn \l_tmpa_seq , {#2}
7548   }
7549   \int_zero:N \l_tmpa_int
7550   \stex_annotation_invisible:n{
7551     $
7552     \stex_annotation:nn{shml:rule={scala://#1}}{%
7553       \stex_annotation_force_break:n{%
7554         \seq_if_empty:NF \l_tmpa_seq {
7555           \seq_map_inline:Nn \l_tmpa_seq {
7556             \int_incr:N \l_tmpa_int
7557             \stex_annotation:nn{%
7558               shml:argmode=i,
7559               shml:arg={\int_use:N \l_tmpa_int}
7560             }{ ##1 }
7561           }
7562         }
7563       }
7564     }$}

```

```

7565     }
7566 }
7567 \stex_deactivate_macro:Nn \MMTrule {module~environments}
7568 \stex_every_module:n{\stex_reactivate_macro:N \MMTrule}

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

mmtinterface (env.)
7569 \NewDocumentEnvironment { mmtinterface } { O{} m m } {
7570   \stex_module_setup_top_nosig:n { #3 }
7571   \str_set_eq:NN \l_todo_mmt_module_str \l_stex_current_module_str
7572   \str_clear:N \l_stex_current_module_str
7573   \stex_keys_set:nn { smodule }{ #1 }
7574   \stex_module_setup:n{ #2 }
7575   \str_set_eq:NN \l_todo_stex_module_str \l_stex_current_module_str
7576   \stex_debug:nn{mmt}{Interface~\l_todo_stex_module_str^~Jfor~\l_todo_mmt_module_str}
7577
7578   \stex_if_do_html:T {
7579     \exp_args:Nne \begin{stex_annotation_env} {
7580       shtml:theory={\l_stex_current_module_str},
7581       shtml:language={ \l_stex_current_language_str },
7582       shtml:signature={}
7583       \tl_if_empty:NF \l_stex_metatheory_uri {
7584         shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
7585       }
7586     }
7587     \stex_annotation_invisible:n{}
7588     \stex_annotation_invisible:nn
7589       {shtml:import=\l_todo_mmt_module_str} {}
7590   }
7591   \stex_module_add_code:x{
7592     \stex_activate_module:n{ \l_todo_mmt_module_str }
7593   }
7594   \stex_module_add_morphism:nonn
7595   {}{\l_todo_mmt_module_str}{import}{}
7596   \stex_reactivate_macro:N \mmtdef
7597   \stex_smsmode_do:
7598 }{
7599   \str_set_eq:NN \l_stex_current_module_str \l_todo_mmt_module_str
7600   \stex_close_module:
7601   \str_set_eq:NN \l_stex_current_module_str \l_todo_stex_module_str
7602   \stex_close_module:
7603   \stex_if_do_html:T { \end{stex_annotation_env} }
7604 }
7605 \stex_sms_allow_env:n{mmtinterface}

\mmtdef
7606 \NewDocumentCommand \mmtdef {m O{} m} {
7607   \stex_keys_set:nn{symdef}{#2}
7608   \str_set:Nx \l_stex_macroname_str { #1 }
7609   \str_if_empty:NT \l_stex_key_name_str {
7610     \str_set:Nx \l_stex_key_name_str { #1 }
7611   }
7612   \stex_symdecl_do:

```

```

7613   \str_set_eq:NN \l_stex_current_module_str \l__todo_mmt_module_str
7614   \cs_set_eq:NN \l__todo_old_metagroup_cd \stex_metagroup_do_in:nn
7615   \cs_set_protected:Npn \stex_metagroup_do_in:nn ##1 ##2 {##2}
7616   \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN} {
7617     {\l_stex_mroname_str}
7618     {\l_stex_key_name_str}
7619     {\int_use:N \l_stex_get_symbol_arity_int}
7620     {\l_stex_get_symbol_args_tl}
7621     {}
7622     {}
7623     {}
7624     {}
7625     \stex_invoke_symbol:
7626   }
7627   \cs_set_eq:NN \stex_metagroup_do_in:nn \l__todo_old_metagroup_cd
7628   \str_set_eq:NN \l_stex_current_module_str \l__todo_stex_module_str
7629
7630   \str_set_eq:NN \l_stex_get_symbol_mod_str \l__todo_mmt_module_str
7631   \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
7632   \stex_notation_parse:n{#3}
7633   \stex_notation_check:
7634   \stex_notation_add:
7635   \stex_if_do_html:T{
7636     \stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7637   }
7638   \stex_smsmode_do:
7639 }
7640 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
7641 \stex_sms_allow_escape:N \mmtdef

```

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

```

7642 \seq_if_empty:NT \g_stex_current_file {
7643   \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
7644 }
7645 \stex_persist_read_now:
7646 \stex_every_file:
7647 \cs_new_protected:Nn \__todo_newlabel:n {
7648   \exp_args:Ne \__todo_old_newlabel:{\tl_to_str:n{#1}}
7649 }
7650 \AtBeginDocument{
7651   \iow_now:Nn \@auxout {
7652     \ExplSyntaxOn
7653     \let\__todo_old_newlabel:\newlabel
7654     \let\newlabel\__todo_newlabel:n
7655     \ExplSyntaxOff
7656   }
7657 }
7658 
```

# 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.

```
7659 <*cls>
7660 @@=notesslides
7661 \ProvidesExplClass{notesslides}{2023/03/19}{3.3.0}{notesslides Class}
7662 \RequirePackage{l3keys2e}
7663
7664 \str_const:Nn \c__notesslides_class_str {article}
7665
7666 \keys_define:nn{notesslides / cls}{
7667   class .str_set_x:N = \c__notesslides_class_str,
7668   notes .bool_set:N = \c__notesslides_notes_bool ,
7669   slides .code:n    = { \bool_set_false:N \c__notesslides_notes_bool },
7670   %docopt .str_set_x:N = \c__notesslides_docopt_str,
7671   unknown .code:n   = {
7672     \PassOptionsToClass{\CurrentOption}{beamer}
7673     \PassOptionsToClass{\CurrentOption}{\c__notesslides_class_str}
7674     \PassOptionsToPackage{\CurrentOption}{notesslides}
7675     \PassOptionsToPackage{\CurrentOption}{stex}
7676   }
7677 }
7678 \ProcessKeysOptions{ notesslides / cls }
7679
7680 \RequirePackage{stex}
7681 \stex_if_html_backend:T {
7682   \bool_set_true:N \c__notesslides_notes_bool
7683 }
7684
7685 \bool_if:NTF \c__notesslides_notes_bool {
7686   \PassOptionsToPackage{notes=true}{notesslides}
7687   \message{notesslides.cls:-Formatting-document-in-notes-mode}
```

```

7688 }{
7689   \PassOptionsToPackage{notes=false}{notesslides}
7690   \message{notesslides.cls:~Formatting~document~in~slides~mode}
7691 }
7692
7693 \bool_if:NTF \c_notesslides_notes_bool {
7694   \LoadClass{\c__notesslides_class_str}
7695 }{
7696   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7697   \%newcounter{Item}
7698   \%newcounter{paragraph}
7699   \%newcounter{subparagraph}
7700   \%newcounter{Hfootnote}
7701 }
7702 \RequirePackage{notesslides}
7703 
```

now we do the same for the `notesslides` package.

```

7704 <*package>
7705 \ProvidesExplPackage{notesslides}{2023/03/19}{3.3.0}{notesslides Package}
7706 \RequirePackage{l3keys2e}
7707
7708 \keys_define:nn{notesslides / pkg}{
7709   notes          .bool_set:N  = \c_notesslides_notes_bool ,
7710   slides         .code:n    = { \bool_set_false:N \c_notesslides_notes_bool },
7711   sectocframes  .bool_set:N  = \c_notesslides_sectocframes_bool ,
7712   topsect        .str_set_x:N = \c_notesslides_topsect_str,
7713   unknown        .code:n    = {
7714     \PassOptionsToPackage{\CurrentOption}{stex}
7715     \PassOptionsToPackage{\CurrentOption}{tikzinput}
7716   }
7717 }
7718 \ProcessKeysOptions{ notesslides / pkg }
7719
7720 \RequirePackage{stex}
7721 \stex_if_html_backend:T {
7722   \bool_set_true:N\c_notesslides_notes_bool
7723 }
7724
7725 \cs_set:Npn \sectiontitleemph #1 {
7726   \textbf{\Large #1}
7727 }
7728
7729 \newif\ifnotes
7730 \bool_if:NTF \c_notesslides_notes_bool {
7731   \notestrue
7732   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7733   \RequirePackage[noamsthm,hyperref]{beamerarticle}
7734   \RequirePackage{mdframed}
7735   \str_if_empty:NTF \c_notesslides_topsect_str{
7736     \%setsectionlevel{section}
7737   }{
7738     \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7739   }

```

```

7740 }{
7741   \notesfalse
7742
7743   \cs_new_protected:Nn \__notesslides_do_sectocframes: {
7744     \cs_set_protected:Nn \__notesslides_do_label:n {
7745       \str_case:nnF{##1} {
7746         {part} {
7747           \tl_set:Nx\l__notesslides_num{\the part}
7748           \tl_set:cx{@ @ label}{%
7749             \cs_if_exist:NTF\parttitlename{\exp_not:N\parttitlename}{\exp_not:N\partname}{-}}
7750         }
7751         {chapter} {
7752           \tl_set:Nx\l__notesslides_num{\the chapter}
7753           \tl_set:cx{@ @ label}{%
7754             \cs_if_exist:NTF\chapertitlename{\exp_not:N\chapertitlename}{\exp_not:N\chaptername}{-}}
7755         }
7756         {section} {
7757           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesection.}
7758           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7759         }
7760         {subsection} {
7761           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsection.}
7762           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7763         }
7764         {subsubsection} {
7765           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsubsection.}
7766           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7767         }
7768         {paragraph} {
7769           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubparagraph.}
7770           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7771         }
7772       }{
7773         \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsubsubsection.}
7774         \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7775       }
7776     }
7777     \cs_set_protected:Nn \_sfragment_do_level:nn {
7778       \tl_if_exist:cT{c@##1}{\stepcounter{##1}}
7779       \addcontentsline{toc}{##1}{\protect\numberline{\use:c{the##1}}##2}
7780       \__notesslides_do_label:n{##1}
7781       \pdfbookmark[\int_use:N \l_stex_docheader_sect]{\l__notesslides_num##2}{##1.\l__note}
7782       \begin{frame}[noframenumbering]
7783         \vfill\centering
7784         \sectiontitleemph{%
7785           \use:c{@ @ label} ##2
7786         }
7787         \end{frame}
7788         \int_incr:N \l_stex_docheader_sect
7789         \tl_set:Nn \stex_current_section_level{##1}
7790       }
7791     }
7792
7793   \AtBeginDocument{

```

```

7794 \str_if_empty:NTF \c_notesslides_topsect_str {
7795   \setsectionlevel{section}
7796 } {
7797   \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7798   \exp_args:No \str_if_eq:nnTF \c_notesslides_topsect_str {chapter} {
7799     \__notesslides_define_chapter:
7800   }{
7801     \exp_args:No \str_if_eq:nnT \c_notesslides_topsect_str {part} {
7802       \__notesslides_define_chapter:
7803       \__notesslides_define_part:
7804     }
7805   }
7806 }
7807 }
7808
7809 \bool_if:NT \c_notesslides_sectocframes_bool {
7810   \__notesslides_do_sectocframes:
7811 }
7812 }

7813 \cs_new_protected:Nn \__notesslides_define_chapter: {
7814   \cs_if_exist:NF \chaptername {
7815     \cs_set_protected:Npn \chaptername {Chapter}
7816   }
7817   \cs_if_exist:NF \chapter {
7818     \cs_set_protected:Npn \chapter {INVALID}
7819   }
7820   \cs_if_exist:NF \c@chapter {
7821     \newcounter{chapter}\counterwithin*{section}{chapter}
7822   }
7823 }
7824 }

7825 \cs_new_protected:Nn \__notesslides_define_part: {
7826   \cs_if_exist:NF \partname {
7827     \cs_set_protected:Npn \partname {Part}
7828   }
7829   \cs_if_exist:NF \part {
7830     \cs_set_protected:Npn \part {INVALID}
7831   }
7832   \cs_if_exist:NF \c@part {
7833     \newcounter{part}\counterwithin*{chapter}{part}
7834   }
7835 }
7836 }

```

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

```

7837 \def \c_notesslides_document_str{document}
7838 \newcommand\afterprematurestop{}
7839 \def\prematurestop@endsfragment{
7840   \unless\ifx\@currenvir\c_notesslides_document_str
7841     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7842       \expandafter\prematurestop@endsfragment
7843     \fi
7844 }

```

```

7845 \providecommand\prematurestop{
7846   \stex_if_html_backend:F{
7847     \message{Stopping-sTeX-processing-prematurely}
7848     \prematurestop@endsfragment
7849   \afterprematurestop
7850   \end{document}
7851 }
7852 }

```

(End definition for `\prematurestop`. This function is documented on page 97.)

### 14.1.2 Notes and Slides

For the notes case, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class.

```

7853 \bool_if:NT \c_notesslides_notes_bool {
7854   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamertheme#2}}
7855 }
7856 \NewDocumentCommand \libusetheme {O{} m} {
7857   \libusepackage[#1]{beamertheme#2}
7858 }

```

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

```

7859 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7860 \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.

```

7861 \ifnotes
7862
7863 \newlength{\slideframewidth}
7864 \setlength{\slideframewidth}{1.5pt}

```

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

```

7865 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7866   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7867     \bool_set_true:N #1
7868   }{
7869     \bool_set_false:N #1
7870   }
7871 }
7872
7873 \stex_keys_define:nnnn{notesslides / frame}{
7874   \str_clear:N \l__notesslides_frame_label_str
7875   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7876   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7877   \bool_set_true:N \l__notesslides_frame_fragile_bool
7878   \bool_set_true:N \l__notesslides_frame_shrink_bool
7879   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7880   \bool_set_true:N \l__notesslides_frame_t_bool
7881 }
7882   label .str_set_x:N = \l__notesslides_frame_label_str,
7883   allowframebreaks .code:n = {

```

```

7884     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7885 },
7886 allowdisplaybreaks .code:n      = {
7887     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7888 },
7889 fragile          .code:n      = {
7890     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7891 },
7892 shrink           .code:n      = {
7893     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7894 },
7895 squeeze          .code:n      = {
7896     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7897 },
7898 t                .code:n      = {
7899     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7900 },
7901 unknown         .code:n      = {}
7902 }{}
```

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

```

7903 \cs_new_protected:Nn \__notesslides_setup_itemize: {
7904     \def\itemize@level{outer}
7905     \def\itemize@outer{outer}
7906     \def\itemize@inner{inner}
7907     \%newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
7908     \renewenvironment{itemize} {
7909         \ifx\itemize@level\itemize@outer
7910             \def\itemize@label{$\rhd$}
7911         \fi
7912         \ifx\itemize@level\itemize@inner
7913             \def\itemize@label{$\scriptstyle\rhd$}
7914         \fi
7915         \begin{list}
7916             {\itemize@label}
7917             {\setlength{\labelsep}{.3em}}
7918             {\setlength{\labelwidth}{.5em}}
7919             {\setlength{\leftmargin}{1.5em}}
7920         }
7921         \edef\itemize@level{\itemize@inner}
7922     }{
7923         \end{list}
7924     }
7925 }
```

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

```

7926 \stex_if_html_backend:TF {
7927     \cs_new_protected:Nn \__notesslides_frame_box_begin: {
7928         \vbox\bgroup
7929         \begin{stex_annotation_env}[shtml:frame={}]
7930             \mdf@patchamsthm\notesslidesfont
7931         }
7932     \cs_new_protected:Nn \__notesslides_frame_box_end: {
7933         %^A \notesslides@slidelabel
```

```

7934     \medskip\par\noindent\tiny\notesslidesfooter
7935     \end{stex_annotation_env}\egroup
7936   }
7937 }{
7938   \cs_new_protected:Nn \__notesslides_frame_box_begin: {
7939     \begin{mdframed}[
7940       linewidth=\slideframewidth,
7941       skipabove=1ex,
7942       skipbelow=1ex,
7943       userdefinedwidth=\slidewidth,
7944       align=center
7945     ]\notesslidesfont
7946   }
7947   \cs_new_protected:Nn \__notesslides_frame_box_end: {
7948     \medskip\par\noindent\tiny\notesslidesfooter%^^A\notesslides@slidelabel
7949     \end{mdframed}
7950   }
7951 }

```

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

```

7952 \renewenvironment{frame}[1][]{
7953   \stex_keys_set:nn{notesslides / frame}{#1}
7954   \stepcounter{framenumber}
7955   \renewcommand{\newpage}{\addtocounter{framenumber}{1}}
7956   \def\@currentlabel{\theframenumber}
7957   \str_if_empty:NF \l__notesslides_frame_label_str {
7958     \label{\l__notesslides_frame_label_str}
7959   }
7960   \__notesslides_setup_itemize:
7961   \__notesslides_frame_box_begin:
7962 }{
7963   \__notesslides_frame_box_end:
7964 }

```

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

```

\frametitle
7965 \renewcommand{\frametitle}[1]{
7966   \stexdoctitle { #1 }
7967   \notesslidestitlenameph{#1}\medskip
7968 }

```

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

```

\pause
7969 \newcommand{\pause}{}

```

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

We redefine the `columns` and `column` environments:

```

7970 \renewenvironment{columns}[1][]{
7971   \par\noindent
7972   \begin{minipage}
7973     \slidewidth\centering\leavevmode
7974   % \stex_if_html_backend:T{

```

```

7975 %     \cs_if_exist:NT \rustex_if:T {
7976 %         \rustex_if:T {\par
7977 %             \rustex_direct_HTML:n{<table><tr><td>}
7978 %         }
7979 %     }
7980 % }
7981 }{
7982 % \stex_if_html_backend:T{
7983 %     \cs_if_exist:NT \rustex_if:T {
7984 %         \rustex_if:T {\par
7985 %             \rustex_direct_HTML:n{</td></tr></table>}
7986 %         }
7987 %     }
7988 % }
7989     \end{minipage}\par\noindent
7990 }
7991 \newsavebox\columnbox
7992 \renewenvironment<>{column}[2] []{
7993     \begin{lrbox}{\columnbox}
7994 %     \stex_if_html_backend:T{
7995 %         \cs_if_exist:NT \rustex_if:T {
7996 %             \rustex_if:T {\par
7997 %                 \rustex_direct_HTML:n{</td><td>}
7998 %             }
7999 %         }
8000 %     }
8001     \begin{minipage}{#2}
8002 }{
8003     \end{minipage}
8004 % \stex_if_html_backend:T{
8005 %     \cs_if_exist:NT \rustex_if:T {
8006 %         \rustex_if:T {\par
8007 %             \rustex_direct_HTML:n{</td><td>}
8008 %         }
8009 %     }
8010 % }
8011     \end{lrbox}\usebox\columnbox
8012 }
8013 \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.

```

8014 \bool_if:NTF \c_notesslides_notes_bool {
8015     \renewenvironment{note}{\ignorespaces}{}
8016 }{
8017     \renewenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup\egroup}{\egroup}
8018 }

```

For other environments we introduce variants prefixed with `n`, which are excluded in `slides` mode.

```

8019 \cs_new_protected:Nn \__notesslides_notes_env:nnnn {
8020   \bool_if:NTF \c_notesslides_notes_bool {
8021     \newenvironment{#1}#2{#3}{#4}
8022   }{
8023     \newenvironment{#1}#2{
8024       \cs_set:Npn \__notesslides_eat: #####1 \end #####2 {
8025         \str_if_eq:nnTF{#1}{#####2} {
8026           \end{#1}
8027         }{
8028           \__notesslides_eat:
8029         }
8030       }
8031       \__notesslides_eat:
8032       \%setbox\l_tmpa_box\vbox\bgroup#3
8033     }{
8034       %#4\egroup
8035     }
8036   }
8037 }

8038 \__notesslides_notes_env:nnnn{nparagraph}{[1][]}{\begin{sparagraph}{#1}}{\end{sparagraph}}
8039 \__notesslides_notes_env:nnnn{nfragment}{[2][]}{\begin{sfragment}{#1}{#2}}{\end{sfragment}}
8040 \__notesslides_notes_env:nnnn{ndefinition}{[1][]}{\begin{sdefinition}{#1}}{\end{sdefinition}}
8041 \__notesslides_notes_env:nnnn{nassertion}{[1][]}{\begin{sassertion}{#1}}{\end{sassertion}}
8042 \__notesslides_notes_env:nnnn{nproof}{[2][]}{\begin{sproof}{#1}{#2}}{\end{sproof}}
8043 \__notesslides_notes_env:nnnn{nexample}{[1][]}{\begin{sexample}{#1}}{\end{sexample}}
8044 \__notesslides_notes_env:nnnn{nexample}{[1][]}{\begin{sexample}{#1}}{\end{sexample}}
8045
8046 \RequirePackage{graphicx}
8047
8048 \NewDocumentCommand\frameimage{s O{} m}{
8049   \IfBooleanTF #1 {
8050     \begin{frame}[plain]
8051   }{
8052     \begin{frame}
8053   }
8054   \bool_if:NTF \c_notesslides_notes_bool {
8055     \slidewidth=\dimexpr\slidewidth-(2\slideframewidth)\relax
8056   }{
8057     \slidewidth=\textwidth\relax
8058   }
8059   \def\Gin@ewidth{}\setkeys{Gin}{#2}
8060   \tl_if_empty:NTF \Gin@ewidth {
8061     \mhgraphics[width=\slidewidth,#2]{#3}
8062   }{
8063     \mhgraphics[#2]{#3}
8064   }
8065   \end{frame}
8066 }

```

hacking inputref:

\inputref\*

```

8067 \cs_set_eq:NN \__notesslides_inputref:\inputref
8068 \cs_set_protected:Npn \inputref{@ifstar\ninputref\__notesslides_inputref:}

```

```

8069 \bool_if:NTF \c_notesslides_notes_bool {
8070   \newcommand\nininputref[2][]{%
8071     \__notesslides_inputref:[#1]{#2}%
8072   }%
8073 }{%
8074   \newcommand\nininputref[2]{}%
8075 }

```

(End definition for `\inputref*`. This function is documented on page 96.)

#### 14.1.4 Styling Across Notes/Slides

```

8076 \def\notesslidestitleemph#1{%
8077   {\Large\bf\sf#1}%
8078   \vskip0.1\baselineskip
8079   \leaders\vrule width \textwidth
8080   \vskip0.4pt%
8081   \nointerlineskip
8082 }%
8083
8084 \def\notesslidesfooter{}%
8085
8086 \let\notesslidesfont\sffamily

```

#### 14.1.5 Beamer Compatibility

All of this should be removed and made part of a template

```

8087
8088 \bool_if:NT \c_notesslides_notes_bool {%
8089   \def\author{\@dblarg\ns@author}%
8090   \long\def\ns@author[#1]#2{%
8091     \tl_if_empty:nTF{#1}{%
8092       \def\beamer@shortauthor{#2}%
8093     }{%
8094       \def\beamer@shortauthor{#1}%
8095     }%
8096   \def\@author{#2}%
8097 }%
8098   \def\title{\@dblarg\ns@title}%
8099   \long\def\ns@title[#1]#2{%
8100     \tl_if_empty:nTF{#1}{%
8101       \def\beamer@shorttitle{#2}%
8102     }{%
8103       \def\beamer@shorttitle{#1}%
8104     }%
8105   \def\@title{#2}%
8106   \stexdoctitle{#2}%
8107 }%
8108   \def\insertshortauthor{%
8109     \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shortauthor\beamer@shortauthor\egroup
8110   }%
8111   \def\insertshorttitle{%
8112     \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shorttitle\beamer@shorttitle\egroup
8113   }%
8114 \stex_if_html_backend:TF{%

```

```

8115     \def\insertframenumber{\stex_annotate:nn{shtml:framenumber={}}{}}
8116   }{
8117     \def\insertframenumber{\@arabic\c@framenumber}
8118   }
8119   \def\insertshortdate{\today}
8120 }

```

#### 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.

```

8121 \gdef\printexcursions{}
8122 \newcommand\excursionref[2]{% label, text
8123   \bool_if:NT \c_notesslides_notes_bool {
8124     \begin{sparagraph}[title=Excursion]
8125       #2 \sref[fallback=the appendix]{#1}.
8126     \end{sparagraph}
8127   }
8128 }
8129 \newcommand\activate@excursion[2][]{%
8130   \tl_gput_right:Nn\printexcursions{\inputref[#1]{#2}}
8131 }
8132 \newcommand\excursion[4][]{% repos, label, path, text
8133   \bool_if:NT \c_notesslides_notes_bool {
8134     \activate@excursion[#1]{#3}
8135     \excursionref{#2}{#4}
8136   }
8137 }

```

(End definition for `\excursion`. This function is documented on page 98.)

**\excursiongroup**

```

8138 \keys_define:nn{notesslides / excursiongroup }{
8139   id      .str_set_x:N = \l__notesslides_excursion_id_str,
8140   intro    .tl_set:N    = \l__notesslides_excursion_intro_tl,
8141   archive  .str_set_x:N = \l__notesslides_excursion_mrepos_str
8142 }
8143 \cs_new_protected:Nn \__notesslides_excursion_args:n {
8144   \tl_clear:N \l__notesslides_excursion_intro_tl
8145   \str_clear:N \l__notesslides_excursion_id_str
8146   \str_clear:N \l__notesslides_excursion_mrepos_str
8147   \keys_set:nn {notesslides / excursiongroup }{ #1 }
8148 }
8149 \newcommand\excursiongroup[1][]{%
8150   \__notesslides_excursion_args:n{ #1 }
8151   \tl_if_empty:NF\printexcursions
8152   {\IfInputref{}{\begin{note}
8153     \begin{sfragment}{Excursions}% TODO pass on id
8154     \ifdefempty{\l__notesslides_excursion_intro_tl}{}{
8155       \exp_args:NNe \use:nn \inputref{[\l__notesslides_excursion_mrepos_str]}{
8156         \l__notesslides_excursion_intro_tl
8157       }
8158     }
8159   }}
```

```

8159      \printexcursions%
8160      \end{sfragment}
8161      \end{note}}}
8162 }
8163 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
(End definition for \excursiongroup. This function is documented on page 98.)
8164 \prop_new:N \g__notesslides_variables_prop
8165 \cs_set_protected:Npn \setSGvar #1 #2 {
8166   \prop_gput:Nnn \g__notesslides_variables_prop {#1}{#2}
8167 }
8168 \cs_set_protected:Npn \useSGvar #1 {
8169   \prop_item:Nn \g__notesslides_variables_prop {#1}
8170 }
8171 \cs_set_protected:Npn \ifSGvar #1 #2 #3 {
8172   \prop_get:NnNF \g__notesslides_variables_prop {#1} \l__notesslides_tmp {
8173     \PackageError{document-structure}
8174     {The sTeX Global variable #1 is undefined}
8175     {set it with \protect\setSGvar}\TODO better error
8176   }
8177   \tl_if_eq:NnT \l__notesslides_tmp {#2}{ #3 }
8178 }
8179
8180
8181 </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.

```

8182 <*package>
8183 <@@=problems>
8184 \ProvidesExplPackage{problem}{2023/03/19}{3.3.0}{Semantic Markup for Problems}
8185 \RequirePackage{l3keys2e}
8186
8187 \keys_define:nn { problem / pkg }{
8188   notes      .default:n    = { true },
8189   notes      .bool_set:N  = \c__problems_notes_bool,
8190   gnotes     .default:n    = { true },
8191   gnotes     .bool_set:N  = \c__problems_gnotes_bool,
8192   hints      .default:n    = { true },
8193   hints      .bool_set:N  = \c__problems_hints_bool,
8194   solutions   .default:n    = { true },
8195   solutions   .bool_set:N  = \c__problems_solutions_bool,
8196   pts        .default:n    = { true },
8197   pts        .bool_set:N  = \c__problems_pts_bool,
8198   min        .default:n    = { true },
8199   min        .bool_set:N  = \c__problems_min_bool,
8200   %boxed     .default:n    = { true },
8201   %boxed     .bool_set:N  = \c__problems_boxed_bool,

```

```

8202 test .default:n = { true },
8203 test .bool_set:N = \c__problems_test_bool,
8204 unknown .code:n = {
8205   \PassOptionsToPackage{\CurrentOption}{stex}
8206 }
8207 }
8208 \newif\ifsolutions
8209
8210 \ProcessKeysOptions{ problem / pkg }
8211 \bool_if:NTF \c__problems_solutions_bool {
8212   \solutionstrue
8213 }{
8214   \solutionsfalse
8215 }
8216 \RequirePackage{stex}

\problem@kw@* For multilinguality, we define internal macros for keywords that can be specialized in *.ldf files.
8217 \AddToHook{begindocument}{

  \ExplSyntaxOn\makeatletter
  \input{problem-english.ldf}
  \ltx@ifpackageloaded{babel}{

    \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
    \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
      \input{problem-ngerman.ldf}
    }
    \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
      \input{problem-finnish.ldf}
    }
    \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
      \input{problem-french.ldf}
    }
    \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
      \input{problem-russian.ldf}
    }
  }{}}
  \makeatother\ExplSyntaxOff
}

```

(End 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.

```

8237 \stex_keys_define:nnnn{ problem }{
8238   \tl_set:Nn \l_stex_key_pts_tl 0
8239   \tl_set:Nn \l_stex_key_min_tl 0
8240   \str_clear:N \l_stex_key_name_str
8241   \str_clear:N \l_stex_key_mhrepos_str
8242 }{
8243   pts .tl_set:N = \l_stex_key_pts_tl,
8244   min .tl_set:N = \l_stex_key_min_tl,

```

```

8245   name .str_set:N = \l_stex_key_name_str,
8246   archive .str_set:N = \l_stex_key_mhrepos_str,
8247   creators .code:n = {}
8248   %imports .tl_set:N = \l__problems_prob_imports_tl,
8249   %refnum .int_set:N = \l__problems_prob_refnum_int,
8250 }{id,title,style,uses}

```

Then we set up a counter for problems.

```
\numberproblemsin
8251 \newcounter{sproblem}[section]
8252 \newcommand\numberproblemsin[1]{
8253   \@addtoreset{sproblem}{#1}
8254   \def\thesproblem{\arabic{#1}.\arabic{sproblem}}
8255 }
8256 \numberproblemsin{section}
8257 \%def\theplainsproblem{\arabic{sproblem}}
8258 \%def\thesproblem{\thesection.\theplainsproblem}
```

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

```
sproblem (env.)
8259 \newcounter{pts}
8260 \newcounter{min}
8261 \stex_new_stylable_env:nnnnnnn {problem} {0{}} {
8262   \cs_if_exist:NTF \l_problem_inputproblem_keys_tl {
8263     \tl_put_left:Nn \l_problem_inputproblem_keys_tl {#1,}
8264     \exp_args:Nno \stex_keys_set:nn{problem} {
8265       \l_problem_inputproblem_keys_tl
8266     }
8267   }{
8268     \stex_keys_set:nn{problem}{#1}
8269   }
8270   \refstepcounter{sproblem}
8271   \str_if_empty:NT \l_stex_key_name_str {
8272     \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8273     \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8274   }
8275
8276   \exp_args:No \stex_module_setup:n \l_stex_key_name_str
8277
8278   \stex_if_do_html:T {
8279     \exp_args:Nne \begin{stex_annotation_env} {
8280       shtml:problem={\l_stex_current_module_str},
8281       shtml:language={ \l_stex_current_language_str},
8282       shtml:signature={\l_stex_key_sig_str}
8283       \tl_if_empty:N \l_stex_metatheory_uri {
8284         shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
8285       }
8286     }
8287     \stex_annotation_invisible:n{}
8288     \tl_if_empty:N \l_stex_key_title_tl {
8289       \exp_args:No \stexdoctitle \l_stex_key_title_tl
8290     }
8291   }
```

```

8292 \stex_if_smsmode:F {
8293   \str_set_eq:NN \thismoduleuri \l_stex_current_module_str
8294   \tl_set_eq:NN \thismodulename \l_stex_key_name_str
8295   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
8296   \stex_style_apply:
8297   \addtocounter{pts}{\l_stex_key_pts_tl}
8298   \addtocounter{min}{\l_stex_key_min_tl}
8299   \_stex_do_id:
8300   \__problems_record_problem:
8301 }
8302 \stex_reactivate_macro:N \solution
8303 \stex_reactivate_macro:N \mcb
8304 \stex_reactivate_macro:N \fillinsol
8305 \stex_smsmode_do:
8306 }{
8307   \stex_close_module:
8308   \stex_if_smsmode:F \stex_style_apply:
8309   \stex_if_do_html:T{ \end{stex_annotation_env} }
8310 }{
8311   \par\noindent\problemheader
8312   \bool_if:NT \c__problems_pts_bool {
8313     \marginpar{\l_stex_key_pts_tl{}~\problem@kw@points\smallskip}
8314   }
8315   \bool_if:NT \c__problems_min_bool {
8316     \marginpar{\l_stex_key_min_tl{}~\problem@kw@minutes\smallskip}
8317   }
8318 \\
8319   \stex_ignore_spaces_and_pars:
8320 }{
8321   \par\bigskip
8322 % \bool_if:NT \c__problems_test_bool \pagebreak
8323 }{s}
8324 \stex_sms_allow_env:n{sproblem}
8325
8326 \tl_set:Nn \problemheader {
8327   \textbf{\sproblemname{\~}\thesproblem}
8328   \tl_if_empty:NF \thistitle {
8329     {\~}(\thistitle)
8330   }
8331 }
8332 }
8333
8334 \cs_new_protected:Nn \__problems_record_problem: {
8335   \exp_args:NNN \iow_now:Nn \auxout {
8336     \problem@restore {\thesproblem}{\l_stex_key_pts_tl}{\l_stex_key_min_tl}
8337   }
8338 }
8339
8340 \cs_new_protected:Npn \problem@restore #1 #2 #3 {}

\includeproblem
8341 \stex_keys_define:nnnn{ includeproblem }{
8342   \str_clear:N \l_stex_key_mhrepos_str
8343 }{

```

```

8344     archive .str_set:N      = \l_stex_key_mhrepos_str,
8345     unknown .code:n = {}
8346 }{}
8347
8348 \NewDocumentCommand\includeproblem{O{} m}{
8349     \group_begin:
8350     \tl_set:Nn \l_problem_inputproblem_keys_tl {#1}
8351     \stex_keys_set:nn{includeproblem}{#1}
8352     \exp_args:Nno \use:nn{\inputref[]}\l_stex_key_mhrepos_str{#2}
8353     \group_end:
8354 }
8355

```

(End definition for `\includeproblem`. This function is documented on page 104.)

`solution (env.)`

```

8356 \int_new:N \g_problem_id_counter
8357
8358 \cs_new_protected:Nn \__problems_solution_start:n {
8359     \str_set:Nn \l_stex_key_title_tl {#1}
8360     \str_set:Nn \l_stex_key_id_str {#1}
8361     \str_if_empty:NT \l_stex_key_id_str {
8362         \int_gincr:N \g_problem_id_counter
8363         \str_set:Nx \l_stex_key_id_str {
8364             SOLUTION_\int_use:N \g_problem_id_counter
8365         }
8366     }
8367     \stex_if_do_html:T{
8368         \begin{stex_annotation_env}[
8369             shtml:solution=\l_stex_key_id_str
8370         ]
8371     }
8372     \stex_style_apply:
8373 }
8374
8375 \stex_new_stylable_env:nnnnnnn { solution }{ 0{} }{
8376     \stex_if_do_html:TF{
8377         \__problems_solution_start:n{#1}
8378     }{
8379         \ifsolutions
8380             \__problems_solution_start:n{#1}
8381         \else
8382             \setbox\l_tmpa_box\vbox\bgroup
8383         \fi
8384     }
8385 }{
8386     \stex_if_do_html:TF{
8387         \stex_style_apply:
8388         \end{stex_annotation_env}
8389     }{
8390         \ifsolutions
8391             \stex_style_apply:
8392             \stex_if_do_html:T{
8393                 \end{stex_annotation_env}

```

```

8394     }
8395     \else
8396         \egroup
8397     \fi
8398 }
8399 }{
8400     \par\smallskip\hrule\smallskip
8401     \noindent\emph{Solution}\str_if_empty:NF \l_stex_key_title_tl{
8402         \{} \l_stex_key_title_tl
8403     \} :-
8404 }{
8405     \par\smallskip\hrule
8406 }{{}
8407
8408 \stex_deactivate_macro:Nn \solution {sproblem~environments}

\startsolution
\stopsolution
8409 \cs_new_protected:Npn \startsolution{
8410     \global\solutionstrue
8411 }
8412 \cs_new_protected:Npn \stopsolution{
8413     \global\solutionsfalse
8414 }

(End definition for \startsolution and \stopsolution. These functions are documented on page
100.)
```

**hint (env.)**

```

8415 \cs_new_protected:Nn \__problems_hint_start:n {
8416     \str_set:Nn \l_stex_key_title_tl {\#1}
8417     \str_set:Nn \l_stex_key_id_str {\#1}
8418     \str_if_empty:NT \l_stex_key_id_str {
8419         \int_gincr:N \g_problem_id_counter
8420         \str_set:Nx \l_stex_key_id_str {
8421             HINT_\int_use:N \g_problem_id_counter
8422         }
8423     }
8424     \stex_if_do_html:T{
8425         \begin{stex_annotation_env}{}
8426             shtml:problemhint=\l_stex_key_id_str
8427         {}
8428     }
8429     \stex_style_apply:
8430 }

8431 \stex_new_stylable_env:nnnnnnn { hint }{ 0{} }{
8432     \stex_if_do_html:TF{
8433         \__problems_hint_start:n{\#1}
8434     }{
8435         \bool_if:NTF \c__problems_hints_bool {
8436             \__problems_hint_start:n{\#1}
8437         }{
8438             \setbox\l_tmpa_box\vbox\bgroup
8439         }
8440 }
```

```

8441     }
8442 }{
8443     \stex_if_do_html:TF{
8444         \stex_style_apply:
8445         \end{stex_annotation_env}
8446 }{
8447     \bool_if:NTF \c__problems_hints_bool {
8448         \stex_style_apply:
8449         \stex_if_do_html:T{
8450             \end{stex_annotation_env}
8451         }
8452     }{
8453         \egroup
8454     }
8455 }
8456 }{
8457     \par\smallskip\hrule\smallskip
8458     \noindent\emph{Hint}\str_if_empty:N \l_stex_key_title_tl{
8459         \{} \l_stex_key_title_tl
8460     \} : \{}
8461 }{
8462     \par\smallskip\hrule
8463 }{}}

exnote (env.)
8464 \cs_new_protected:Nn \__problems_exnote_start:n {
8465     \str_set:Nn \l_stex_key_title_tl {\#1}
8466     \str_set:Nn \l_stex_key_id_str {\#1}
8467     \str_if_empty:NT \l_stex_key_id_str {
8468         \int_gincr:N \g_problem_id_counter
8469         \str_set:Nx \l_stex_key_id_str {
8470             EXNOTE_\int_use:N \g_problem_id_counter
8471         }
8472     }
8473     \stex_if_do_html:T{
8474         \begin{stex_annotation_env} {
8475             \shtml:problemnote=\l_stex_key_id_str
8476         }
8477     }
8478     \stex_style_apply:
8479 }
8480
8481 \stex_new_stylable_env:nnnnnnn { exnote }{ 0{} }{
8482     \stex_if_do_html:TF{
8483         \__problems_exnote_start:n{\#1}
8484     }{
8485         \bool_if:NTF \c__problems_notes_bool {
8486             \__problems_exnote_start:n{\#1}
8487         }{
8488             \setbox\l_tmpa_box\vbox\bgroup
8489         }
8490     }
8491 }{
8492     \stex_if_do_html:TF{

```

```

8493     \stex_style_apply:
8494     \end{stex_annotation_env}
8495 }{
8496     \bool_if:NTF \c__problems_notes_bool {
8497         \stex_style_apply:
8498         \stex_if_do_html:T{
8499             \end{stex_annotation_env}
8500         }
8501     }{
8502         \egroup
8503     }
8504 }
8505 }{
8506     \par\smallskip\hrule\smallskip
8507     \noindent\emph{Note}\str_if_empty:N \l_stex_key_title_tl{
8508         \l_stex_key_title_tl
8509     } :~}
8510 }{
8511     \par\smallskip\hrule
8512 }{}}

gnote (env.)
8513 \cs_new_protected:Nn \__problems_gnote_start:n {
8514     \str_set:Nn \l_stex_key_title_tl {\#1}
8515     \str_set:Nn \l_stex_key_id_str {\#1}
8516     \str_if_empty:NT \l_stex_key_id_str {
8517         \int_gincr:N \g_problem_id_counter
8518         \str_set:Nx \l_stex_key_id_str {
8519             GNOTE_\int_use:N \g_problem_id_counter
8520         }
8521     }
8522     \stex_if_do_html:T{
8523         \begin{stex_annotation_env} {
8524             \shtml:problemgnote=\l_stex_key_id_str
8525         }
8526     }
8527     \stex_style_apply:
8528 }

8529
8530 \stex_new_stylable_env:nnnnnnn { gnote }{ 0{} }{
8531     \stex_if_do_html:TF{
8532         \__problems_gnote_start:n{\#1}
8533     }{
8534         \bool_if:NTF \c__problems_gnotes_bool {
8535             \__problems_gnote_start:n{\#1}
8536         }{
8537             \setbox\l_tmpa_box\vbox\bgroup
8538         }
8539     }
8540 }{
8541     \stex_if_do_html:TF{
8542         \stex_style_apply:
8543         \end{stex_annotation_env}
8544     }{

```

```

8545     \bool_if:NTF \c__problems_gnotes_bool {
8546         \stex_style_apply:
8547         \stex_if_do_html:T{
8548             \end{stex_annotation_env}
8549         }
8550     }{
8551         \egroup
8552     }
8553 }
8554 }{
8555     \par\smallskip\hrule\smallskip
8556     \noindent\emph{Grading}\str_if_empty:N \l_stex_key_title_tl{
8557         \{} \l_stex_key_title_tl
8558         \} : \{
8559     }{
8560         \par\smallskip\hrule
8561     }{ }

```

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

```

8562 \def\pts#1{
8563     \bool_if:NT \c__problems_pts_bool {
8564         \marginpar{\#1~\problem@kw@points}
8565     }
8566 }
8567 \def\min#1{
8568     \bool_if:NT \c__problems_min_bool {
8569         \marginpar{\#1~\problem@kw@minutes}
8570     }
8571 }

mcb (env.)
8572 \newenvironment{mcb}{\par
8573     \stex_if_do_html:T{
8574         \begin{stex_annotation_env}{shtml:multiple-choice-block={}}
8575     }
8576     \stex_deactivate_macro:Nn \mcb {sproblem~environments}
8577     \stex_deactivate_macro:Nn \solution {sproblem~environments}
8578     \stex_reactivate_macro:N \mcc
8579     \begin{enumerate}
8580    }{
8581     \end{enumerate}
8582     \stex_if_do_html:T{
8583         \end{stex_annotation_env}
8584     }
8585 }
8586 \stex_deactivate_macro:Nn \mcb {sproblem~environments}

we define the keys for the \mcc macro

8587 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8588     \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8589         \bool_set_true:N #1
8590     }{
8591         \bool_set_false:N #1
8592     }

```

```

8593 }
8594 \stex_keys_define:nnnn{mcc}{
8595   \tl_clear:N \l_stex_key_feedback_tl
8596   \bool_set_false:N \l_stex_key_T_bool
8597   \tl_clear:N \l_stex_key_Ttext_tl
8598   \tl_clear:N \l_stex_key_Ftext_tl
8599 }{
8600   feedback .tl_set:N      = \l_stex_key_feedback_tl ,
8601   T        .default:n    = { false } ,
8602   T        .bool_set:N   = \l_stex_key_T_bool ,
8603   F        .default:n    = { false } ,
8604   F        .code:n       = {\bool_set_false:N \l_stex_key_T_bool} ,
8605   Ttext    .tl_set:N     = \l_stex_key_Ttext_tl ,
8606   Ftext    .tl_set:N     = \l_stex_key_Ftext_tl ,
8607 }{id}
8608

\mcc

8609 \tl_set:Nn \problem_mcc_box_tl {
8610   \ltx@ifpackageloaded{amssymb}{$\square$}){
8611     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
8612   }
8613 }
8614 \newcommand\mcc[2][]{
8615   \stex_keys_set:nn{mcc}{#1}\par
8616   \tl_set:Nn \l_tmpb_tl {~~~
8617     \bool_if:NTF \l_stex_key_T_bool {
8618       \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
8619     }{
8620       \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
8621     }
8622     \tl_if_empty:NF \l_stex_key_feedback_tl {
8623       \\\emph{\l_stex_key_feedback_tl}
8624     }
8625   }
8626   \tl_set:Nn \l_tmpa_tl {
8627     #2
8628     \stex_if_do_html:TF{
8629       \stex_annotation:nn{shtml:mcc-solution={}}{\l_tmpb_tl}
8630     }{
8631       \if solutions \l_tmpb_tl \fi
8632     }
8633   }
8634   \item[\problem_mcc_box_tl]{}
8635   \stex_if_do_html:TF{
8636     \stex_annotation:nn{shtml:mcc=}
8637       \bool_if:NTF \l_stex_key_T_bool {true}{false}
8638     }{\l_tmpa_tl}
8639   }{\l_tmpa_tl}
8640 }
8641 \stex_deactivate_macro:Nn \mcc {mcb~environments}

```

(End definition for `\mcc`. This function is documented on page [101](#).)

```

8642 \newcommand\fillinsol[2][]{%
8643   \quad
8644   \stex_if_do_html:TF{%
8645     \stex_annotation:n{<html>\fillinsol={}}{ \_stex_annotation_force_break:n{#2} }%
8646   }{%
8647     \ifxolutions
8648       \textcolor{red}{\fbox{#2}}%
8649     \else
8650       \fbox{%
8651         \tl_if_empty:nTF{#1}{%
8652           \phantom{\huge{#2}}%
8653         }{%
8654           \hspace{#1}%
8655         }%
8656       }%
8657     \fi
8658   }%
8659   \quad
8660 }%
8661 \stex_deactivate_macro:Nn \fillinsol {sproblem-environments}%

```

(End definition for `\fillinsol`. This function is documented on page 103.)

```

8662 </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.

```

8663 <*package>
8664 \ProvidesExplPackage{hwexam}{2023/03/19}{3.3.0}{homework assignments and exams}
8665 \RequirePackage{l3keys2e}
8666
8667 \keys_define:nn {hwexam / pkg}{%
8668   multiple .default:n = { false },
8669   multiple .bool_set:N = \c_hwexam_multiple_bool,
8670   unknown .code:n = {
8671     \PassOptionsToPackage{\CurrentOption}{problem}
8672   }
8673 }
8674 \ProcessKeysOptions{ hwexam /pkg }
8675 \RequirePackage{problem}

```

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

```

8676 \AddToHook{begindocument}{%
8677 \ExplSyntaxOn\makeatletter
8678 \input{hwexam-english.ldf}
8679 \ltx@ifpackageloaded{babel}{%

```

```

8680 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8681 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8682 \input{hwexam-ngerman.ldf}
8683 }
8684 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8685 \input{hwexam-finnish.ldf}
8686 }
8687 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8688 \input{hwexam-french.ldf}
8689 }
8690 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8691 \input{hwexam-russian.ldf}
8692 }
8693 }{}
8694 \makeatother\ExplSyntaxOff
8695 }

```

(End 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.)`

```

8696 \stex_keys_define:nnnn{ assignment }{
8697   \tl_clear:N \l_stex_key_number_tl
8698   \tl_clear:N \l_stex_key_given_tl
8699   \tl_clear:N \l_stex_key_due_tl
8700 }{
8701   number .tl_set:N      = \l_stex_key_number_tl,
8702   given  .tl_set:N      = \l_stex_key_given_tl,
8703   due    .tl_set:N      = \l_stex_key_due_tl,
8704   unknown .code:n = f}
8705 }{id,title,style}
8706
8707 \newcounter{assignment}
8708 \stex_new_stylable_env:nnnnnn {assignment}{0}{ }{
8709   \cs_if_exist:NTF \l_hwexam_includeassignment_keys_tl {
8710     \tl_put_left:Nn \l_hwexam_includeassignment_keys_tl {#1,}
8711     \exp_args:Nno \stex_keys_set:nn{assignment}{}{
8712       \l_hwexam_includeassignment_keys_tl
8713     }
8714   }{
8715     \stex_keys_set:nn{assignment}{#1}
8716   }
8717   \tl_if_empty:NF \l_stex_key_number_tl {
8718     \global\setcounter{assignment}{\int_eval:n{\l_stex_key_number_tl-1}}
8719   }
8720   \global\refstepcounter{assignment}
8721   \setcounter{sproblem}{0}
8722   \def\thesproblem{\theassignment.\arabic{sproblem}}
8723   \stex_style_apply:

```

```

8724 \_stex_do_id:
8725 }{
8726 \stex_style_apply:
8727 }{
8728 \par\begin{center}
8729 \textbf{\Large\assignmentautorefname~\theassignment}
8730 \tl_if_empty:NF \l_stex_key_title_tl {
8731 {\~}---\l_stex_key_title_tl
8732 }
8733 }\par\smallskip
8734 \textbf{
8735 \tl_if_empty:NF \l_stex_key_given_tl {
8736 \hwexam@kw@given:~\l_stex_key_given_tl\quad
8737 }
8738 \tl_if_empty:NF \l_stex_key_due_tl {
8739 \hwexam@kw@due:~\l_stex_key_due_tl\quad
8740 }
8741 }
8742 \end{center}
8743 \par\bigskip
8744 }{
8745 \par\pagebreak
8746 }{}}

\includeassignment
8747 \NewDocumentCommand\includeassignment{O{} m}{
8748 \group_begin:
8749 \tl_set:Nn \l_hwexam_includeassignment_keys_tl {#1}
8750 \stex_keys_set:nn{includeproblem}{#1}
8751 \exp_args:Nno \use:nn{\inputref[]}\l_stex_key_mhrepos_str]{#2}
8752 \group_end:
8753 }

```

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

Restoring information about problems:

```

8754 \prop_new:N \c_@@_problems_prop
8755 \tl_set:Nn \c_@@_total_mins_tl {0}
8756 \tl_set:Nn \c_@@_total_pts_tl {0}
8757 \int_new:N \c_@@_total_problems_int
8758 \cs_set_protected:Npn \problem@restore #1 #2 #3 {
8759 \int_gincr:N \c_@@_total_problems_int
8760 \prop_gput:Nnn \c_@@_problems_prop {#1}{#2}{#3}
8761 \tl_gset:Nx \c_@@_total_pts_tl { \int_eval:n { \c_@@_total_pts_tl + #2 } }
8762 \tl_gset:Nx \c_@@_total_mins_tl { \int_eval:n { \c_@@_total_mins_tl + #2 } }
8763 }

```

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

```

8764 \newcommand\correction@table{
8765 \int_compare:nNnT \c_@@_total_problems_int = 0 {
8766 \int_incr:N \c_@@_total_problems_int
8767 \prop_put:Nnn \c_@@_problems_prop {\~}{\~}{\~}
8768 }
8769 \tl_clear:N \l_tmpa_tl
8770 \tl_clear:N \l_tmpb_tl

```

```

8771 \tl_clear:N \l_tmpc_tl
8772 \prop_map_inline:Nn \c_@@_problems_prop {
8773 \tl_put_right:Nn \l_tmpa_tl { ##1 & }
8774 \tl_put_right:Nx \l_tmpb_tl { \use_i:nn ##2 & }
8775 \tl_put_right:Nn \l_tmpc_tl { & }
8776 }
8777 \resizebox{\textwidth}{!}{%
8778 \exp_args:Nne \begin{tabular}{|l|*{\int_use:N \c_@@_total_problems_int}{c|}c||l|}\hline
8779 &\exp_args:Ne \multicolumn{\int_eval:n{ \c_@@_total_problems_int + 1}}{c||}
8780 {\footnotesize\hwexam@kw@forgrading} &\hline
8781 \hwexam@kw@probs & \l_tmpa_tl \hwexam@kw@sum & \hwexam@kw@grade\\ \hline
8782 \hwexam@kw@pts & \l_tmpb_tl \c_@@_total_pts_tl & \\ \hline
8783 \hwexam@kw@reached & \l_tmpc_tl & \hline
8784 \end{tabular}}}

```

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

`\testheading`

```

8785 \def\hwexamheader{\input{hwexam-default.header}}
8786
8787 \def\hwexamminutes{
8788 \tl_if_empty:NTF \hwexam@duration {
8789 {\hwexam@min}~\hwexam@minutes@kw
8790 }{
8791 \hwexam@duration
8792 }
8793 }
8794
8795 \stex_keys_define:nnnn{ hwexam / testheading }{
8796 \tl_clear:N \hwexam@min
8797 \tl_clear:N \hwexam@duration
8798 \tl_clear:N \hwexam@reqpts
8799 \tl_clear:N \hwexam@tools
8800 }{
8801 min .tl_set:N = \hwexam@min,
8802 duration .tl_set:N = \hwexam@duration,
8803 reqpts .tl_set:N = \hwexam@reqpts,
8804 tools .tl_set:N = \hwexam@tools
8805 }{}}
8806
8807 \newenvironment{testheading}[1][]{%
8808 \stex_keys_set:nn { hwexam / testheading}{#1}
8809
8810 \tl_set_eq:NN \hwexam@totalpts \c_@@_total_pts_tl
8811 \tl_set_eq:NN \hwexam@totalmin \c_@@_total_mins_tl
8812 \tl_set:Nx \hwexam@checktime {\int_eval:n { \hwexam@min - \hwexam@totalmin }}}
8813
8814 \newif\if@bonuspoints
8815 \tl_if_empty:NTF \hwexam@reqpts {
8816 \if@bonuspointsfalse
8817 }{
8818 \tl_set:Nx \hwexam@bonuspts {
8819 \int_eval:n{\hwexam@totalpts - \hwexam@reqpts}
8820 }

```

```

8821 \@bonuspointstrue
8822 }
8823
8824 \makeatletter\hwexamheader\makeatother
8825 }{
8826 \newpage
8827 }

```

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

#### `\testemptypage`

```

8828 \newcommand\testemptypage[1][]{%
8829 \bool_if:NT \c__problems_test_bool {\vfill\begin{center}\hwexam@kw@testemptypage\end{center}}
8830 }

```

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

#### `\testspace`

```

8831 \newcommand\testspace[1]{\bool_if:NT \c__problems_test_bool {\vspace*{#1}}}

```

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

#### `\testnewpage`

```

8832 \newcommand\testnewpage{\bool_if:NT \c__problems_test_bool {\newpage}}

```

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

```

8833 </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

```
8834 <@=tikzinput>
8835 {*package}
8836
8837 %%%%%%%%%% tikzinput.dtx %%%%%%
8838
8839 \ProvidesExplPackage{tikzinput}{2023/03/19}{3.3.0}{tikzinput package}
8840 \RequirePackage{l3keys2e}
8841
8842 \keys_define:nn { tikzinput } {
8843   image .bool_set:N = \c_tikzinput_image_bool,
8844   image .default:n = false ,
8845   unknown .code:n = {}
8846 }
8847
8848 \ProcessKeysOptions { tikzinput }
8849
8850 \bool_if:NTF \c_tikzinput_image_bool {
8851   \RequirePackage{graphicx}
8852
8853   \providecommand\usetikzlibrary[]{}
8854   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
8855 }
8856 \RequirePackage{tikz}
8857 \RequirePackage{standalone}
8858
8859 \newcommand\tikzinput [2] [] {
8860   \setkeys{Gin}{#1}
8861   \ifx \Gin@ewidth \Gin@exclamation
8862     \ifx \Gin@eheight \Gin@exclamation
8863       \input { #2 }
8864     \else
8865       \resizebox{!}{\Gin@eheight }{
8866         \input { #2 }
8867       }
8868     \fi
8869   \else
8870     \ifx \Gin@eheight \Gin@exclamation
8871       \resizebox{ \Gin@ewidth }{!}{
8872         \input { #2 }
8873       }
8874     \else
8875       \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
8876         \input { #2 }
8877       }
8878     \fi
8879   \fi
8880 }
8881
8882
8883 \newcommand\ctikzinput [2] [] {
8884   \begin{center}
8885     \tikzinput [#1] {#2}
8886   
8887 }
```

```
8886     \end{center}  
8887 }  
8888 </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	
\\$ .....	2070, 2073, 2077
\; .....	7460, 7491, 7504, 7506, 7519
@@ commands:	
\c_@@_problems_prop .....	8754, 8760, 8767, 8772
\c_@@_total_mins_tl .	8755, 8762, 8811
\c_@@_total_problems_int .....	8757, 8759, 8765, 8766, 8778, 8779
\c_@@_total_pts_tl .....	8756, 8761, 8782, 8810
\\ 46, 70, 1048, 7749, 7754, 8109, 8112,	8318, 8623, 8780, 8781, 8782, 8783
\{ .....	7509
\} .....	7509
\_ 37, 633, 7781, 8829	
\_comp .....	3922, 4124, 4465, 5161, 5980, 6004, 6084
\_customthiscomp .....	6503, 6504, 6516
\_defcomp .....	6012, 6942, 6951
\_stex_html_do_output_bool .....	272, 273, 276, 281, 285, 316, 319, 2131, 7451
\_thiscomp .....	6486, 6497, 6498
\_varcomp 4446, 4745, 4754, 4911, 4949,	5195, 5637, 5936, 5949, 5962, 6008
\  .....	2069, 2072, 2076
<b>A</b>	
\activateexcursion .....	98
\addbibresource .....	69, 1943
\addcontentsline .....	7779
\addmhbibresource .....	69, 1937
\addtocounter .....	7955, 8297, 8298
\AddToHook .....	1030, 1033, 7255, 7322, 7325, 7437, 8217, 8676
\aftergroup .....	132, 5149, 7320, 7434
\afterprematurestop .....	97, 7838, 7849
\apply .....	77
\arabic .....	8254, 8257, 8722
\arg .....	45, 82, 5374
\argarraymap .....	81, 4185, 4685
\argmap .....	81, 4184, 4326, 4661, 7492
\argsep .....	39, 81, 4183, 4321, 4645, 7460, 7461, 7462, 7468
\assign .....	63, 118, 2967, 3323, 3450
assignment (env.) .....	105, 8696
<b>B</b>	
\assignmentautorefname .....	8729
\assignMorphism .....	118, 2968, 3452
\assumption .....	7233, 7338
\ast .....	7483
\AtBeginDocument .....	1000, 1253, 1439, 1442, 1488, 7650, 7793
\AtEndDocument .....	641, 1489
\AtEndOfFile .....	2020, 2034, 2047
\author .....	8089
\autoref .....	73, 1633
<b>B</b>	
\backmatter .....	1453, 1454, 1455
\baselineskip .....	8078
\beamernitemnestingprefix .....	8163
\begin ... 85, 124, 125, 130, 190, 1296, 1376, 2031, 2044, 2062, 2211, 2233, 2250, 2500, 2758, 2804, 4708, 6837, 7147, 7180, 7202, 7240, 7311, 7383, 7579, 7782, 7915, 7929, 7939, 7972, 7993, 8001, 8039, 8040, 8041, 8042, 8043, 8044, 8050, 8052, 8124, 8152, 8153, 8279, 8368, 8425, 8474, 8523, 8574, 8579, 8728, 8778, 8829, 8884	
\begingroup .....	352, 1915
\bf .....	8077
\bgroup .....	3765, 4791, 5003, 7928, 8017, 8032, 8109, 8112, 8382, 8439, 8488, 8537
\bigskip .....	8321, 8743
blindfragment (env.) .....	71
bool commands:	
\bool_if:NTF .....	190, 612, 664, 665, 671, 1138, 1148, 1150, 1209, 1230, 2114, 2369, 2719, 2752, 2936, 3432, 3638, 4806, 4909, 5159, 5193, 5735, 5829, 5835, 5970, 6932, 7208, 7256, 7294, 7313, 7316, 7318, 7323, 7326, 7334, 7376, 7390, 7412, 7438, 7685, 7693, 7730, 7809, 7853, 8014, 8020, 8054, 8069, 8088, 8123, 8133, 8211, 8312, 8315, 8322, 8436, 8447, 8485, 8496, 8534, 8545, 8563, 8568, 8617, 8637, 8829, 8831, 8832, 8850
\bool_if:nTF .....	276
\bool_if_exist:NTF .....	249, 265

```

\bbool_lazy_any:nTF ..... 4017, 4057
\bbool_lazy_any_p:n ..... 767
\bbool_new:N ..... 272, 2111,
    2354, 2777, 4717, 5129, 5704, 5826
\bbool_not_p:n ..... 766, 770
\bbool_set_false:N ..... .
    . 175, 281, 319, 1135, 1159,
    2112, 2131, 2355, 2732, 2799, 3405,
    4577, 4720, 5133, 5705, 5830, 7066,
    7215, 7263, 7291, 7432, 7451, 7452,
    7669, 7710, 7869, 8591, 8596, 8604
\bbool_set_true:N ..... .
    . 180, 251, 267, 273, 285,
    316, 601, 607, 1132, 2130, 2716,
    2796, 3417, 3427, 3635, 3766, 4811,
    5051, 5130, 5359, 5368, 5484, 5495,
    5593, 5678, 5736, 5747, 5780, 5809,
    5855, 6051, 6458, 6489, 6506, 6551,
    7078, 7245, 7682, 7722, 7867, 7875,
    7876, 7877, 7878, 7879, 7880, 8589
\bbool_while_do:Nn ..... 1133
\bbool_while_do:nn ..... .
    . 765, 1975, 7082, 7094, 7106, 7123, 7135
    \l_tmpa_bool .. 3405, 3417, 3427, 3432
box commands:
\box_clear:N ..... 2134
\c_empty_box ..... 3947, 3954
\l_tmpa_box .. 2129, 2134, 3650, 4176,
    8017, 8032, 8382, 8439, 8488, 8537
boxed ..... 99

C
\catcode ..... 37, 46, 352, 633, 1048,
    1686, 2068, 2069, 2070, 2071, 2072,
    2073, 2075, 2076, 2077, 2411, 2412
\cdot ..... 7491, 7504
\centering ..... 7783, 7973
\chapter ..... 26, 71, 7818, 7819
\chaptername ..... 7754, 7815, 7816
\chapertitlename ..... 7754
\circ ..... 53
\clearpage ..... 1448, 1458, 1469, 1480
clist commands:
\clist_clear:N .. 87, 399, 3678, 4106,
    5052, 6340, 6439, 6754, 6759, 7340
\clist_count:N ..... 6320, 6330
\clist_count:n ..... 4693, 6422
\clist_get:NN ..... 455, 506
\clist_if_empty:NTF .. 174, 451,
    502, 3782, 4274, 6319, 6365, 6622, 6826
\clist_if_in:NnTF ..... .
    . 59, 62, 93, 6920, 8222, 8225,
    8228, 8231, 8681, 8684, 8687, 8690
\clist_if_in:nnTF ..... 6448
\clist_item:Nn ..... 4283
\clist_item:nn ..... 4704
\clist_map_function:NN .. 5053, 6368
\clist_map_function:nN ..... .
    . 2669, 2694, 3569, 3589, 3610
\clist_map_inline:Nn ..... .
    . 88, 97, 177, 2821, 3784, 4829,
    5022, 6147, 6197, 6623, 6736, 6772
\clist_map_inline:nn ..... .
    . 363, 412, 5525, 5624, 5653, 6145, 7022
\clist_pop:NN ..... 4285
\clist_put_right:Nn ..... .
    . 89, 5066, 5070, 6347, 6466, 6468
\clist_set:Nn .. 31, 85, 4930, 8221, 8680
\clist_set_eq:NN ..... 83, 6842
\l_tmpa_clist .. 8221, 8222, 8225, 8228,
    8231, 8680, 8681, 8684, 8687, 8690
\clstinputmhlisting ..... 72, 2019
\cmhgraphics ..... 72, 2019
\cmhtikzinput ..... 107, 2019
\columnbox ..... 7991, 7993, 8011
\comp ..... 31, 32, 35, 45, 79,
    82, 92, 120, 3922, 4124, 4297, 4446,
    4465, 4507, 4513, 4515, 4745, 4754,
    4949, 5178, 5179, 5513, 5637, 5857,
    5862, 5877, 5936, 5949, 5962, 5980,
    5982, 5989, 6084, 6307, 6315, 6498,
    6512, 6513, 6942, 6943, 6951, 7457,
    7459, 7466, 7468, 7470, 7471, 7476,
    7478, 7483, 7485, 7486, 7493, 7494,
    7495, 7496, 7499, 7501, 7505, 7506,
    7507, 7509, 7519, 7526, 7528, 7530
\compemph ..... 92, 5989
\conclude ..... 7232, 7338
\conclusion .. 49–51, 88, 90, 6911, 7031, 7047
\copymod ..... 61, 62, 3579, 3581, 3583
\copymodule ..... 3469, 3471
\counterwithin ..... 7822, 7834
cs commands:
\cs:w .. 438, 441, 473, 634, 2144,
    2373, 2374, 2375, 2382, 2386, 2392,
    5257, 6188, 6238, 6240, 6860, 6873
\cs_argument_spec:N ..... 3971
\cs_end: .. 438, 441, 473, 634, 2144,
    2373, 2374, 2375, 2384, 2388, 2392,
    5257, 6188, 6238, 6240, 6860, 6873
\cs_generate_from_arg_count:NNn .. .
    . 3775,
    . 4573, 4820, 5013, 5097, 5298, 5330
\cs_generate_variant:Nn ..... .
    . 80, 150, 224, 237, 551, 575,
    584, 596, 620, 680, 709, 864, 869,
    873, 957, 1119, 1144, 1546, 1871,

```

2176, 2534, 2538, 2543, 2561, 2610,  
 2632, 2645, 3100, 5154, 5171, 5488  
`\cs_gset:Npn` ..... 2407  
`\cs_if_eq:NNTF` ..... 133,  
 879, 933, 1415, 2200, 2211, 2214,  
 2233, 2236, 3973, 4646, 5569, 5571  
`\cs_if_exist:NTF` .....  
 302, 1087, 1321, 1325, 1363, 1366,  
 1390, 1394, 1423, 1429, 1443, 1453,  
 1466, 1477, 1632, 1780, 1821, 1827,  
 1831, 2081, 2622, 2627, 3949, 3956,  
 3968, 4395, 4403, 4844, 4851, 5038,  
 5084, 5248, 5274, 5618, 5857, 5862,  
 6404, 6701, 6704, 6708, 6711, 6716,  
 6719, 6723, 6726, 7749, 7754, 7757,  
 7761, 7765, 7769, 7773, 7815, 7818,  
 7821, 7827, 7830, 7833, 7975, 7983,  
 7995, 8005, 8109, 8112, 8262, 8709  
`\cs_new:Nn` ..... 202,  
 425, 539, 545, 552, 561, 577, 586,  
 758, 878, 2185, 2368, 2416, 2927,  
 3745, 4076, 4089, 4289, 4356, 4373,  
 4378, 4522, 5093, 5306, 5403, 6125,  
 6180, 6376, 6382, 6385, 6395, 6817  
`\cs_new:Npn` 537, 614, 615, 623, 624,  
 874, 2868, 2899, 4082, 4095, 5549, 5922  
`\cs_new_nopar:Nn` ..... 359, 374  
`\cs_new_protected:Nn` ..... 36,  
 45, 53, 58, 68, 74, 77, 129, 151, 209,  
 220, 230, 239, 257, 279, 322, 328,  
 342, 348, 430, 437, 450, 464, 472,  
 491, 501, 516, 538, 617, 627, 639,  
 644, 663, 675, 686, 702, 711, 726,  
 784, 791, 821, 856, 866, 870, 892,  
 909, 922, 928, 932, 959, 963, 979,  
 995, 1001, 1014, 1079, 1086, 1109,  
 1120, 1129, 1146, 1156, 1165, 1192,  
 1239, 1247, 1293, 1301, 1305, 1315,  
 1361, 1374, 1384, 1421, 1499, 1509,  
 1535, 1575, 1609, 1621, 1631, 1639,  
 1679, 1737, 1748, 1779, 1785, 1789,  
 1803, 1807, 1849, 1874, 1890, 1914,  
 1925, 1937, 1964, 1990, 2064, 2095,  
 2099, 2103, 2118, 2122, 2128, 2139,  
 2147, 2177, 2199, 2205, 2223, 2246,  
 2255, 2271, 2278, 2291, 2304, 2319,  
 2327, 2336, 2357, 2379, 2400, 2422,  
 2426, 2434, 2446, 2531, 2535, 2539,  
 2548, 2556, 2562, 2580, 2587, 2599,  
 2613, 2620, 2633, 2647, 2664, 2673,  
 2684, 2689, 2698, 2709, 2714, 2738,  
 2756, 2770, 2778, 2819, 2828, 2840,  
 2844, 2856, 2864, 2872, 2887, 2900,  
 2946, 2952, 2973, 3016, 3049, 3116,  
 3135, 3155, 3161, 3166, 3193, 3224,  
 3249, 3275, 3325, 3369, 3436, 3519,  
 3544, 3554, 3649, 3657, 3718, 3730,  
 3749, 3791, 3799, 3853, 3862, 3944,  
 3953, 3958, 3965, 3986, 4003, 4016,  
 4027, 4029, 4042, 4053, 4141, 4157,  
 4208, 4220, 4247, 4268, 4273, 4294,  
 4305, 4314, 4424, 4455, 4497, 4526,  
 4533, 4554, 4568, 4604, 4609, 4614,  
 4645, 4718, 4762, 4776, 4789, 4839,  
 4857, 4870, 4876, 4886, 4894, 4957,  
 4971, 4984, 4985, 4987, 5031, 5064,  
 5083, 5095, 5107, 5136, 5158, 5173,  
 5192, 5217, 5222, 5227, 5294, 5310,  
 5316, 5351, 5357, 5365, 5420, 5481,  
 5490, 5506, 5519, 5553, 5564, 5585,  
 5617, 5649, 5695, 5698, 5708, 5721,  
 5734, 5757, 5758, 5759, 5763, 5789,  
 5792, 5818, 5828, 5916, 5968, 5989,  
 6079, 6104, 6111, 6128, 6134, 6183,  
 6243, 6251, 6258, 6263, 6317, 6339,  
 6346, 6350, 6413, 6438, 6464, 6496,  
 6502, 6521, 6533, 6544, 6562, 6572,  
 6612, 6621, 6629, 6638, 6648, 6659,  
 6673, 6697, 6779, 6831, 6877, 6890,  
 6928, 6999, 7015, 7091, 7179, 7186,  
 7190, 7201, 7216, 7333, 7356, 7374,  
 7647, 7743, 7814, 7826, 7865, 7903,  
 7927, 7932, 7938, 7947, 8019, 8143,  
 8334, 8358, 8415, 8464, 8513, 8587  
`\cs_new_protected:Npn` .....  
 ... 199, 331, 334, 338, 339, 422,  
 682, 885, 888, 1235, 1319, 1388,  
 1404, 1490, 1492, 1820, 1826, 1830,  
 1845, 2080, 2191, 2274, 2350, 2833,  
 2893, 2931, 2959, 2990, 3004, 3087,  
 3102, 3236, 3562, 4393, 4539, 4546,  
 4596, 4652, 4661, 4686, 5037, 5050,  
 5115, 5121, 5125, 5155, 5205, 5246,  
 5272, 5397, 5533, 5847, 5867, 5874,  
 6004, 6008, 6012, 6018, 6022, 6026,  
 6030, 6034, 6038, 6042, 6046, 6050,  
 6284, 6306, 6311, 6402, 6421, 6434,  
 6486, 6733, 6770, 7080, 7104, 7121,  
 7133, 7176, 7539, 8340, 8409, 8412  
`\cs_parameter_spec:N` ..... 541, 555  
`\cs_prefix_spec:N` ..... 553  
`\cs_set:Npn` ..... 442, 477, 482,  
 630, 631, 813, 814, 815, 816, 937,  
 944, 1088, 1091, 1104, 2153, 2402,  
 2435, 2649, 2651, 2654, 2667, 2692,  
 2717, 2731, 3776, 3989, 4178, 4179,  
 4183, 4184, 4185, 4297, 4327, 4557,  
 4573, 4667, 4692, 4821, 4859, 5014,

5097, 5299, 5331, 5651, 6441, 6447,	46
6508, 7448, 7449, 7450, 7725, 8024	56, 63, 88–90, 6901, 6978, 6996, 6997
\cs_set:Npx . . . . .	2591
\cs_set_eq:NN . . . . .	34, 211, 215,
807, 808, 809, 1255, 1262, 1380,	2628, 2970, 2971, 3969, 4571, 4624,
4637, 4852, 5085, 5088, 5162, 5164,	5196, 5198, 5374, 5377, 5405, 5406,
5619, 5622, 5761, 7615, 7627, 8067	5619, 5622, 5761, 7615, 7627, 8067
\cs_set_protected:Nn . . . . .	804, 811, 1559, 7744, 7777
\cs_set_protected:Npn . . . . .	1274, 1283, 3945, 3946, 4321, 4326,
4332, 6016, 7616, 7816, 7819, 7828,	7831, 8068, 8165, 8168, 8171, 8758
\cs_set_protected:Npx . . . . .	6504
\cs_undefine:N . . . . .	206, 216, 263, 1200
\l_tmpa_cs . . . . .	4178, 4188, 5651, 5669
\csname . . . . .	261,
290, 349, 2066, 2067, 2068, 2069,	2070, 2075, 2076, 2077, 2372, 2635
\ctikzinput . . . . .	107, 8883
\CurrentFile . . . . .	1017, 1019, 1021
\CurrentFilePath . . . . .	1017, 1018, 1021
\currentgrouplevel . . . . .	221, 243, 244, 245,
247, 249, 251, 259, 261, 263, 265, 267	276, 281, 283, 285, 287, 289, 291,
\CurrentOption . . . . .	7, 7672, 7673,
7674, 7675, 7714, 7715, 8205, 8671	8206, 8207, 8208, 8209, 8210, 8211,
\Currentsectionlevel . . . . .	71, 1283
\currentsectionlevel . . . . .	71, 1274
<b>D</b>	
\DeclareOption . . . . .	7
\def . . . . .	80, 130,
296, 300, 303, 305, 351, 554, 1237,	1241, 1258, 1263, 2036, 2039, 2050,
2054, 3922, 4124, 4145, 4444, 4446,	4463, 4465, 4745, 4752, 4754, 4947,
4949, 5178, 5179, 5637, 5877, 5906,	5920, 5936, 5949, 5962, 5980, 5991,
5920, 5936, 5949, 5962, 5980, 5991,	6084, 6511, 6516, 6942, 6951, 7837,
6084, 6511, 6516, 6942, 6951, 7837,	7839, 7904, 7905, 7906, 7910, 7913,
7839, 7904, 7905, 7906, 7910, 7913,	7956, 8059, 8076, 8084, 8089, 8090,
7956, 8059, 8076, 8084, 8089, 8090,	8092, 8094, 8096, 8098, 8099, 8101,
8092, 8094, 8096, 8098, 8099, 8101,	8103, 8105, 8108, 8109, 8111, 8112,
8103, 8105, 8108, 8109, 8111, 8112,	8115, 8117, 8119, 8163, 8254, 8257,
8115, 8117, 8119, 8163, 8254, 8257,	8258, 8562, 8567, 8722, 8785, 8787
\defemph . . . . .	92, 5989
\Definame . . . . .	89, 6886, 6916, 6969, 6975
\definename . . . . .	23, 35,
44, 88, 89, 92, 6885, 6915, 6961, 6967	\definiendum . . . . .
44, 88, 89, 92, 6883, 6913, 6955, 6959	23, 35,
definiendum . . . . .	6928
<b>E</b>	
\edef . . . . .	109, 2068, 2069, 2070, 6498, 7921
\egroup . . . . .	3789, 4836, 5029, 7935, 8017, 8034,
8109, 8112, 8396, 8453, 8502, 8551	\elgroup . . . . .
\eject . . . . .	8829
\ellipses . . . . .	84,
85, 4671, 4672, 5065, 5067, 5625, 5663	\else . . . . .
8381, 8395, 8649, 8864, 8869, 8874	289, 301,
\emph . . . . .	1878, 1961, 2187, 5852, 6031, 8163,
7268, 8401, 8458, 8507, 8556, 8623	8381, 8395, 8649, 8864, 8869, 8874
\end . . . . .	125, 130, 1302, 1356, 2031, 2044,
2062, 2214, 2236, 2259, 2519, 2774,	2852, 4710, 6854, 7161, 7187, 7249,
7266, 7317, 7319, 7397, 7401, 7603,	7787, 7841, 7850, 7923, 7935, 7949,
7787, 7841, 7850, 7923, 7935, 7949,	7989, 8003, 8011, 8024, 8026, 8039,
7989, 8003, 8011, 8024, 8026, 8039,	8040, 8041, 8042, 8043, 8044, 8065,
8040, 8041, 8042, 8043, 8044, 8065,	8126, 8160, 8161, 8309, 8388, 8393,
8126, 8160, 8161, 8309, 8388, 8393,	8445, 8450, 8494, 8499, 8543, 8548,
8445, 8450, 8494, 8499, 8543, 8548,	8581, 8583, 8742, 8784, 8829, 8886
\endcsname . . . . .	261, 289, 290,
349, 2066, 2067, 2068, 2069, 2070,	349, 2066, 2067, 2068, 2069, 2070,
2075, 2076, 2077, 2372, 2635, 8163	2075, 2076, 2077, 2372, 2635, 8163
\endgroup . . . . .	354, 356, 1922
\endinput . . . . .	1760, 2261
\ensuremath . . . . .	5737, 5748
environments:	
assignment . . . . .	105, 8696
blindfragment . . . . .	71
exnote . . . . .	1, 8464
extstructure . . . . .	85, 6143
extstructure* . . . . .	85
frame . . . . .	1, 7865
gnote . . . . .	1, 8513

hint ..... 1, 8415  
 mathstructure ..... 85, 6061  
 mcb ..... 1, 8572  
 mminterface ..... 7569  
 nassertion ..... 1  
 ndefinition ..... 1  
 nexample ..... 1  
 note ..... 1  
 nparagraph ..... 1, 1  
 nsproof ..... 1  
 problem ..... 1  
 sassertion ..... 88  
 sdefinition ..... 88  
 sexample ..... 88  
 sfragment ..... 71  
 smodule ..... 75, 2491  
 solution ..... 1, 8356  
 sparagraph ..... 88  
 sproblem ..... 8259  
 sproof ..... 90, 7179  
 stex\_annotate\_env ..... 130  
 subproof ..... 7272  
 testheading ..... 105  
 \eq ..... 31, 41  
 \eqstep ..... 7234, 7338  
 \equal ..... 30  
 \escapechar ..... 46, 1048  
 \everyeof ..... 2140  
 \excursion ..... 98, 8121  
 \excursiongroup ..... 98, 8138  
 \excursionref ..... 98, 8122, 8135  
 exnote (env.) ..... 1, 8464  
 exp commands:  
     \exp\_after:wN ..... 349, 438,  
         441, 473, 540, 691, 880, 2142, 2186,  
         2187, 2188, 2328, 2373, 2374, 2375,  
         2382, 2386, 2390, 2391, 2913, 2934,  
         3547, 3779, 4078, 4091, 4675, 4703,  
         4824, 5017, 5254, 5256, 5264, 5266,  
         5312, 5319, 5320, 5321, 5340, 5373,  
         5536, 5538, 5640, 5641, 5642, 5655,  
         5850, 5913, 5959, 5975, 6187, 6237,  
         6240, 6564, 6860, 6873, 6937, 6972  
     \exp\_args:N ..... 528, 533, 547, 548,  
         879, 933, 1021, 1094, 1121, 1643,  
         1792, 1804, 1816, 1832, 1835, 1838,  
         1840, 2322, 2816, 2822, 2912, 2915,  
         3344, 3372, 3378, 3741, 3750, 3907,  
         3971, 3973, 4646, 4791, 4988, 5423,  
         5569, 5571, 5580, 6148, 6198, 6285,  
         6289, 6367, 6448, 6497, 6503, 6737,  
         6773, 6866, 7151, 7358, 7648, 8779  
     \exp\_args:NNe .....  
         ... 47, 191, 194, 588, 1066, 1071,  
             1077, 2436, 2875, 3523, 4010, 4192,  
             4290, 5455, 5458, 5542, 5589, 5674,  
             6468, 6576, 6631, 6699, 8155, 8335  
     \exp\_args:Nne ..... 200,  
         540, 2012, 2086, 2460, 2500, 2557,  
         2758, 2836, 4148, 4299, 4338, 4446,  
         4465, 4754, 4949, 6086, 6523, 6564,  
         6632, 6837, 7202, 7579, 8279, 8778  
     \exp\_args:NNNo .....  
         825, 1169, 3531, 4527, 4529, 4763, 4958  
     \exp\_args:NNno ..... 260, 677, 825  
     \exp\_args:Nnno ..... 1719, 1727  
     \exp\_args:NNNx ..... 1169, 1708  
     \exp\_args:NNnx .....  
         ... 830, 3051, 4763, 4958, 5434, 5440  
     \exp\_args:NNo .....  
         ... 31, 41, 59, 62, 89, 93, 158,  
         166, 689, 798, 830, 834, 839, 844,  
         1009, 1172, 1650, 3184, 3210, 3528,  
         3534, 3538, 4240, 4253, 4585, 5668  
     \exp\_args:Nno ..... 245, 247, 260,  
         372, 2403, 2938, 2941, 3774, 4010,  
         4819, 5012, 5118, 5123, 5207, 5998,  
         6457, 7000, 8264, 8352, 8711, 8751  
     \exp\_args:NNx ..... 48, 103,  
         115, 904, 991, 1887, 1935, 2104,  
         2123, 6590, 7052, 7380, 8222, 8225,  
         8228, 8231, 8681, 8684, 8687, 8690  
     \exp\_args:Nnx .....  
         ... 240, 1310, 1584, 1652, 1887,  
         1935, 2791, 2992, 2997, 3184, 3210,  
         3731, 3898, 5096, 5297, 5329, 5339,  
         5849, 6118, 6794, 7052, 7380, 7617  
     \exp\_args:No ..... 135,  
         155, 156, 157, 204, 295, 433, 600,  
         606, 744, 1019, 1105, 1205, 1257,  
         1265, 1287, 1310, 1523, 1708, 1713,  
         1718, 1750, 1755, 1757, 2058, 2343,  
         2440, 2495, 2532, 2861, 2873, 2910,  
         3097, 3112, 3354, 3355, 3545, 3550,  
         3634, 3736, 3737, 3738, 4008, 4160,  
         4518, 4693, 4769, 4770, 4771, 4782,  
         4783, 4784, 4964, 4965, 4966, 4977,  
         4978, 4979, 5071, 5140, 5143, 5147,  
         5188, 5327, 5332, 5337, 5342, 5344,  
         5432, 5460, 5569, 5571, 5598, 5683,  
         6096, 6106, 6138, 6211, 6323, 6327,  
         6334, 6361, 6396, 6398, 6470, 6472,  
         6478, 6553, 6579, 6585, 6587, 6592,  
         6596, 6602, 6649, 6698, 6941, 6991,  
         7738, 7797, 7798, 7801, 8276, 8289  
     \exp\_args:Nx .....  
         ... 242, 690, 745, 753, 1527, 1633,  
         1635, 2192, 2497, 2976, 7866, 8588

\exp_not:N	49, 204, 206, 554, 939, 946, 1104, 1215, 1218, 1940, 1993, 2140, 2186, 2396, 2792, 3347, 3381, 4159, 4535, 5099, 5102, 5138, 5139, 5142, 5146, 5150, 5307, 5382, 5388, 5590, 5594, 5597, 5675, 5679, 5681, 5851, 6205, 6397, 6471, 6477, 6513, 6552, 6578, 6584, 6593, 6604, 7749, 7754	
\exp_not:n	204, 537, 1105, 1310, 1523, 2390, 2417, 2440, 2570, 2594, 3353, 3354, 3355, 3736, 3737, 3738, 3927, 4160, 4201, 4202, 4518, 4675, 4703, 4769, 4770, 4771, 4782, 4783, 4784, 4964, 4965, 4966, 4977, 4978, 4979, 5093, 5140, 5143, 5147, 5149, 5151, 5188, 5320, 5327, 5332, 5337, 5342, 5344, 5556, 5598, 5635, 5640, 5683, 6181, 6323, 6361, 6391, 6396, 6398, 6470, 6472, 6474, 6478, 6481, 6507, 6553, 6555, 6579, 6581, 6585, 6587, 6592, 6596, 6602	
\expandafter	290, 354, 2066, 2067, 2068, 2069, 2070, 2372, 2635, 6031, 7841, 7842	
\ExplSyntaxOff	2109, 2549, 7655, 8235, 8694	
\ExplSyntaxOn	2108, 2545, 7652, 8218, 8677	
\extref	74, 1547	
\extstructure (env.)	85, 6143	
\extstructure	6175, 6177	
\extstructure* (env.)	85	
<b>F</b>		
\fbox	8648, 8650	
\fi	292, 307, 1882, 1897, 1961, 2188, 5160, 5194, 5865, 5969, 6031, 6929, 7502, 7843, 7911, 7914, 8013, 8163, 8383, 8397, 8631, 8657, 8868, 8878, 8879	
\fiboxed	94	
file commands:		
\file_if_exist:nTF	628, 645, 1158	
\filepath	127, 128	
\fillinsol	103, 8304, 8642	
\first	126	
\fn	53	
\foo	14, 53, 77, 130, 132	
\fooname	14	
\footnotesize	8780	
\foral	45, 50	
\forallall	45, 7489, 7495, 7506	
\frame (env.)	1, 7865	
\frameimage	97, 8048	
\frameimages	94	
\frametitle	7965	
<b>G</b>		
\frontmatter	1443, 1444, 1445	
\fun	44	
\funspace	36	
<b>H</b>		
\gdef	8121	
\given	105	
\global	1237, 1241, 1258, 1263, 2067, 7535, 7536, 8410, 8413, 8718, 8720	
\gnote (env.)	1, 8513	
\gnotes	99, 105	
group commands:		
\group_begin:	36, 45, 629, 1047, 1686, 2151, 2410, 2447, 2766, 2786, 2810, 3041, 3265, 3306, 3651, 3703, 3865, 3867, 3869, 3871, 3931, 4133, 4440, 4486, 4748, 4903, 4943, 5072, 5076, 5174, 5327, 5483, 5491, 5565, 5586, 5634, 5650, 5853, 5890, 5898, 5909, 5931, 5942, 5955, 6488, 6505, 6861, 6930, 6979, 7032, 7246, 7313, 7335, 7398, 7419, 7447, 8349, 8748	
\group_end:	40, 48, 635, 1051, 1733, 2174, 2428, 2461, 2775, 2791, 2853, 3046, 3271, 3312, 3653, 3711, 3865, 3867, 3869, 3871, 3936, 3956, 4136, 4451, 4489, 4759, 4915, 4954, 5043, 5061, 5073, 5077, 5118, 5251, 5262, 5279, 5290, 5312, 5340, 5345, 5361, 5394, 5486, 5503, 5565, 5589, 5642, 5674, 5861, 5986, 6405, 6461, 6491, 6514, 6524, 6590, 6870, 6947, 6993, 7045, 7257, 7323, 7327, 7376, 7413, 7438, 7537, 8353, 8752	
\group_insert_after:N	250, 266	
<b>I</b>		
\have	7338	
\hbox	1249, 3037, 3340, 3765, 3919, 3948, 4315, 4791, 5003, 6159, 6743, 7167, 7191, 8109, 8112, 8611	
hbox commands:		
\hbox_set:Nn	3650, 4176	
\hbox_unpack:N	3947, 3954	
\HCode	302	
\hfil	7173	
\hfill	7173	
\hint (env.)	1, 8415	
\hints	99, 105	
\hline	8778, 8780, 8781, 8782, 8783	
\href	1827	
\hrule	7167, 8400, 8405, 8457, 8462, 8506, 8511, 8555, 8560, 8611	
\hspace	8654	

```

\HTML ..... 14, 25
\huge ..... 8652
hwexam commands:
  \l_hwexam_includeassignment_-
    keys_t1 ..... 8709, 8710, 8712, 8749
  \hwexam_kw_* ..... 8676
  \c_hwexam_multiple_bool ..... 8669
hwexam@kw@due commands:
  \hwexam@kw@due: ..... 8739
hwexam@kw@given commands:
  \hwexam@kw@given: ..... 8736
\hwexamheader ..... 8785, 8824
\hwexamminutes ..... 8787
\hyperref ..... 1821, 1822
I
\id ..... 112
\if ..... 2186, 5848
\IfBooleanTF ..... 3695, 4127,
  4437, 5411, 7284, 7391, 7414, 8049
\ifcsname ..... 289, 8163
\ifdefempty ..... 8154
\iffalse ..... 7488
\IfFileExists ..... 1713, 1896,
  1904, 1977, 1985, 3213, 3218, 3238
\IfInputref ..... 26, 27, 72, 1954, 8152
\ifinputref ..... 26, 72, 1872, 1961
\ifmmode ..... 6031
\ifnotes ..... 95, 7729, 7861
\ifSGvar ..... 98, 8171
\ifsolutions ..... 100, 8208, 8379, 8390, 8631, 8647
\ifstexhtml ..... 27, 70, 129, 289
\ifvmode ..... 1897, 5160, 5194, 5969, 6929
\ifx 2066, 7840, 7909, 7912, 8861, 8862, 8870
\ignorespaces ..... 4759, 4954, 8015
image ..... 106
\imply ..... 50
\importmodule ..... 37, 48, 54, 82–84,
  119, 122, 124, 125, 132, 201, 2964, 3243
\includeassignment ..... 8747
\includegraphics ..... 72, 2029, 8854
\includeproblem ..... 104, 8341
\indent ..... 5160, 5194, 5969, 6929
\infprec 79, 80, 4240, 5820, 5823, 5856, 7459
\inline* ..... 89
\inlineass ..... 49, 54, 57, 89
\inlinedef ..... 49, 89
\inlineex ..... 89
\input ..... 26, 69, 124, 33, 75, 78, 311,
  1024, 1909, 1996, 8219, 8223, 8226,
  8229, 8232, 8678, 8682, 8685, 8688,
  8691, 8785, 8863, 8866, 8872, 8876
\inputassignment ..... 105
\inputref ..... 26, 27, 71, 72, 91, 1872,
  8067, 8068, 8130, 8155, 8352, 8751
\inputref* ..... 96, 8067
\inputreffalse ..... 1872, 1881
\inputreftrue ..... 1879, 1916
\insertframenumber ..... 8115, 8117
\insertshortauthor ..... 8108
\insertshortdate ..... 8119
\insertshorttitle ..... 8111
\inset ..... 44
int commands:
  \int_case:nn ..... 1362
  \int_case:nnTF ..... 1320, 1389, 1422
  \int_compare:nNnTF ..... 242, 2010, 2085, 3524, 4165, 4200,
  4699, 5281, 5568, 5579, 5628, 5834,
  6320, 6330, 6422, 6805, 7113, 8765
  \int_compare_p:nNn ..... 7083, 7095, 7107, 7124, 7136
\int_decr:N ..... 7114, 7142
\int_eval:n ..... 259, 261, 263, 265, 267,
  5437, 5443, 5833, 6119, 6126, 6253,
  8718, 8761, 8762, 8779, 8812, 8819
\int_gincr:N ..... 1501,
  5422, 8362, 8419, 8468, 8517, 8759
\int_gset:Nn ..... 5386
\int_gzero:N ..... 5371
\int_incr:N ..... 1312, 1322,
  1326, 1350, 1364, 1367, 1370, 1391,
  1395, 1426, 1432, 3841, 3842, 3843,
  3844, 4615, 4698, 5534, 7088, 7100,
  7111, 7128, 7140, 7556, 7788, 8766
\int_new:N . 1270, 1497, 3797, 3985,
  4685, 5364, 5518, 5822, 8356, 8757
\int_set:Nn ..... 1406, 1407, 1408, 1409,
  1410, 1411, 1413, 3021, 3806, 3810,
  3814, 3818, 3822, 3826, 3830, 3834,
  3838, 3913, 3992, 4033, 4065, 4588,
  4693, 4700, 4738, 4862, 5494, 5823,
  5856, 7081, 7093, 7105, 7122, 7134
\int_step_function:nN ..... 5100, 5300
\int_step_inline:nn ..... 3855, 4232, 4239, 4259, 4509
\int_use:N ..... 221, 1296,
  1376, 1416, 1436, 1502, 3351, 3734,
  4309, 4426, 4588, 4767, 4780, 4962,
  4975, 5210, 5386, 5423, 5555, 5592,
  5677, 5820, 5821, 6796, 7559, 7620,
  7781, 8364, 8421, 8470, 8519, 8778
\int_zero:N ..... 3800, 3801, 4570, 4696, 5524, 7549
\c_max_int ..... 5820, 5821
\l_tmpa_int ..... 4570, 4573, 4588, 4615, 4696, 4698,

```

intarray commands:	
\intarray_gset:Nnn	7116, 7130, 7143, 7218
\intarray_gzero:N	7217
\intarray_item:Nn	7084, 7087, 7096, 7099, 7100, 7105, 7108, 7111, 7113, 7114, 7116, 7117, 7122, 7125, 7128, 7130, 7134, 7137, 7140, 7142, 7143, 7549, 7556, 7559
intarray commands:	
\intarray_new:Nn	7079
\interpretmod	3600, 3604
\interpretmodule	3490, 3492
invokation commands:	
\invokation_macro	112, 114, 120
ior commands:	
\ior_close:N	652, 658, 1052, 1183
\ior_map_inline:Nn	1168
\ior_new:N	1164
\ior_open:Nn	646, 654, 1166
\ior_open:NnTF	1046
\ior_str_get:NN	1049
\ior_str_map_inline:Nn	648, 655
\g_tmpa_ior	646, 648, 652, 654, 655, 658, 1046, 1049, 1052
iow commands:	
\iow_close:N	641, 651, 1489
\iow_new:N	610, 1487
\iow_now:Nn	618, 649, 656, 1515, 1786, 7651, 8335
\iow_open:Nn	640, 647, 1488
\g_tmpa_iow	647, 649, 651
\isassociative	48
\iscommutative	48
\item	7184, 8634
\interpretmod	3602
\itshape	7177
J	
\jobname	66, 130, 640, 645, 646, 647, 654, 659, 668, 974, 1488, 1708
\join	62
K	
keys commands:	
\l_keys_choice_tl	3668
\keys_define:nn	21, 372, 7666, 7708, 8138, 8187, 8667, 8842
\l_keys_key_str	4114, 6075
\keys_set:nn	376, 8147
keyval commands:	
\keyval_parse>NNn	6342
L	
\label	73, 110, 121, 1527, 1804, 7958
\labelsep	7917
\labelwidth	7918
\lambda	7489, 7500, 7501
\langle	7476
\Large	7726, 8077, 8729
\LaTeX	22
\latext	22
\ldots	7481, 7526, 7528, 7530
\leaders	8079
\leavevmode	7973
\left	5850
\leftmargin	7919
\let	349, 1444, 1445, 1454, 1455, 1716, 2067, 2141, 2152, 2159, 2170, 2182, 3289, 3948, 3955, 4445, 4464, 4753, 4948, 5177, 5895, 5905, 5919, 5981, 6060, 7535, 7536, 7653, 7654, 8086
\libinput	19, 69, 1964
\libusepackage	69, 70, 2008, 7857
\libusetHEME	7856
\libusetikZlibrary	69, 107, 2064
\LoadClass	12, 7694, 7696
\long	130, 581, 8090, 8099
\lstinputlisting	72, 2043
\lstinputmhlisting	72, 2019
M	
\macro	124–126, 131
\macroname	31
\magma	52
\maincomp	31, 32, 35, 53, 79, 86, 92, 3955, 4159, 4168, 4179, 4180, 4201, 4332, 4535, 5179, 5360, 5877, 6056, 6497, 6498, 6503, 6511, 6516
\mainmatter	1466, 1467, 1477, 1478
\makeatletter	2106, 8218, 8677, 8824
\makeatother	2107, 8235, 8694, 8824
\maketitle	1262, 1263
\mapsto	7498, 7499
\marginnote	7907
\marginpar	8313, 8316, 8564, 8569
\mathbb	7512
\mathbin	32, 7457, 7461, 7468, 7494, 7505
\mathclose	32, 5862, 7459, 7466, 7470, 7476, 7478, 7486, 7493, 7505, 7509
\mathhub	67, 127, 159, 24, 1035
\mathop	32, 7496, 7507
\mathopen	32, 5857, 7459, 7466, 7470, 7476, 7478, 7485, 7493, 7505, 7509
\mathord	32
\mathpunct	32, 4513, 5513, 7485, 7495, 7501
\mathrel	31, 32, 7462, 7499

\mathrm . . . . .	4535, 7485
mathstructure (env.) . . . . .	85, 6061
\mathstructure . . . . .	6101, 6102
\mathhtt . . . . .	7465, 7466, 7515, 7518, 7522
\mcb (env.) . . . . .	1, 8572
\mcb . . . . .	8303, 8576, 8586
\mcc . . . . .	101, 8578, 8609
\meaning . . . . .	1017, 2372, 2635, 4205, 4479, 4564, 4590, 5117, 5122, 5325, 5335, 5336
\medskip . . . . .	7934, 7948, 7967
\meet . . . . .	62
\message . . . . .	20, 7687, 7690, 7847
\mhframeimage . . . . .	97
\mhgraphics . . . . .	72, 2019, 8061, 8063
\mhinput . . . . .	72, 1872
\mhtikzinput . . . . .	107, 2019
\min . . . . .	105
\min . . . . .	8567
\min . . . . .	99, 105
\mmlarg . . . . .	330
\mmlintent . . . . .	330
\mmtdef . . . . .	7596, 7606
\MMTinclude . . . . .	7540
\mmtinterface (env.) . . . . .	7569
\MMTrule . . . . .	7545
\mname . . . . .	76, 85
mode commands:	
\mode_if_math:TF . . . . .	3340, 3948, 3949, 3956, 5219
\mode_if_vertical:TF . . . . .	3947, 3954
\MSC . . . . .	7539
msg commands:	
\msg_error: . . . . .	131
\msg_error:nn . . . . .	75, 1775, 6987, 7040
\msg_error:nnn . . . . .	78, 147, 186, 234, 805, 941, 948, 1966, 1969, 2082, 3176, 3232, 3403, 3433, 3961, 4648, 4890, 6131, 6608, 6635
\msg_error:nnnn . . . . .	345, 823, 828, 847, 882, 954, 1940, 1993, 2895, 2983, 3328, 3846, 4418, 4913, 5168, 5202, 5448, 5475, 5984, 6245, 6945
\msg_none:nn . . . . .	72
\msg_redirect_module:nnn . . . . .	94
\msg_redirect_name:nnn . . . . .	98
\msg_set:nnn . . . . .	69
\msg_warning:nn . . . . .	1065
\msg_warning:nnn . . . . .	1685, 1727
\msg_warning:nnnn . . . . .	427, 1719
\mult . . . . .	39, 40, 80, 81
\multicolumn . . . . .	8779
\multiple . . . . .	105
	N
nassertion (env.) . . . . .	1
\Nat . . . . .	35
ndefinition (env.) . . . . .	1
\neginfpref 40, 79, 80, 4210, 4213, 4222, 4225, 4238, 5039, 5275, 5286, 5820	
\newcommand . . . . .	438, 473, 1569, 1947, 1955, 1960, 2001, 2008, 2038, 2044, 2052, 2062, 7442, 7838, 7907, 7969, 8070, 8074, 8122, 8129, 8132, 8149, 8252, 8614, 8642, 8764, 8828, 8831, 8832, 8854, 8859, 8883
\newcounter ... 7697, 7698, 7699, 7700, 7822, 7834, 8251, 8259, 8260, 8707	
\NewDocumentCommand . . . . .	
... 441, 1533, 1765, 1770, 1814, 1886, 1934, 2465, 2544, 4902, 5410, 5889, 5897, 5908, 5930, 5941, 5954, 6860, 6950, 6955, 6961, 6969, 6978, 7021, 7031, 7049, 7375, 7411, 7426, 7545, 7606, 7856, 8048, 8348, 8747	
\NewDocumentEnvironment . . . . .	
... 476, 1342, 1351, 7431, 7569	
\newenvironment . . . . .	1463, 1474, 7145, 7352, 8021, 8023, 8572, 8807
\newif . . . . .	290, 313, 1872, 7729, 8208, 8814
\newlabel . . . . .	7653, 7654
\newlength . . . . .	7859, 7860, 7863
\newpage . . . . .	7955, 8826, 8832
\newsavebox . . . . .	7991
nexample (env.) . . . . .	1
\ninputref . . . . .	8068, 8070, 8074
\nobreak . . . . .	7173
\noindent . . . . .	1297, 1897, 6844, 7163, 7934, 7948, 7971, 7989, 8311, 8401, 8458, 8507, 8556
\nointerlineskip . . . . .	8081
\notation . . . . .	31, 32, 40, 77, 79, 84, 115, 118, 124, 2963, 4117, 7457, 7460, 7461, 7462, 7475, 7477, 7495, 7496, 7500, 7504, 7506, 7507, 7512, 7515
note (env.) . . . . .	1
notes . . . . .	94, 99, 105
\notesfalse . . . . .	7741
notesslides commands:	
\c_notesslides_notes_bool . . . . .	
... 7668, 7669, 7682, 7685, 7693, 7709, 7710, 7722, 7730, 7853, 8014, 8020, 8054, 8069, 8088, 8123, 8133	
\c_notesslides_sectocframes_bool . . . . .	7711, 7809
\c_notesslides_topsect_str . . . . .	7712, 7735, 7738, 7794, 7797, 7798, 7801

notesslides internal commands:

- \c\_notesslides\_class\_str ..... 7664, 7667, 7673, 7694
- \\_notesslides\_define\_chapter: ... 7799, 7802, 7814
- \\_notesslides\_define\_part: .... 7803, 7826
- \\_notesslides\_do\_label:n 7744, 7780
- \\_notesslides\_do\_sectocframes: . 7743, 7810
- \\_notesslides\_do\_yes\_param:Nn .. 7865, 7884, 7887, 7890, 7893, 7896, 7899
- \c\_notesslides\_docopt\_str ... 7670
- \c\_notesslides\_document\_str ... 7837, 7840
- \\_notesslides\_eat: . 8024, 8028, 8031
- \\_notesslides\_excursion\_args:n . 8143, 8150
- \l\_notesslides\_excursion\_id\_str ..... 8139, 8145
- \l\_notesslides\_excursion\_intro\_- tl ..... 8140, 8144, 8154, 8156
- \l\_notesslides\_excursion\_- mhrepos\_str ..... 8141, 8146, 8155
- \l\_notesslides\_frame\_allowdisplaybreaks\_- bool ..... 7876, 7887
- \l\_notesslides\_frame\_allowframebreaks\_- bool ..... 7875, 7884
- \\_notesslides\_frame\_box\_begin: . 7927, 7938, 7961
- \\_notesslides\_frame\_box\_end: ... 7932, 7947, 7963
- \l\_notesslides\_frame\_fragile\_- bool ..... 7877, 7890
- \l\_notesslides\_frame\_label\_str . 7874, 7882, 7957, 7958
- \l\_notesslides\_frame\_shrink\_- bool ..... 7878, 7893
- \l\_notesslides\_frame\_squeeze\_- bool ..... 7879, 7896
- \l\_notesslides\_frame\_t\_bool ... 7880, 7899
- \\_notesslides\_inputref: ..... 8067, 8068, 8071
- \\_notesslides\_notes\_env:nmm ... 8019, 8039, 8040, 8041, 8042, 8043, 8044
- \l\_notesslides\_num ..... 7747, 7749, 7752, 7754, 7757, 7758, 7761, 7762, 7765, 7766, 7769, 7770, 7773, 7774, 7781
- \\_notesslides\_setup\_itemize: ... 7903, 7960

- \l\_notesslides\_tmp ..... 8172, 8177
- \g\_notesslides\_variables\_prop .. 8164, 8166, 8169, 8172
- \notesslidesfont ..... 7930, 7945, 8086
- \notesslidesfooter ..... 7934, 7948, 8084
- \notesslidestitleemph ..... 7967, 8076
- \notestrue ..... 7731
- nparagraph (env.) ..... 1, 1
- nsproof (env.) ..... 1
- \null ..... 7173
- \num ..... 132
- \number ..... 105
- \numberline ..... 7779
- \numberproblemsin ..... 8251

## O

- \OMDoc ..... 25
- \omdoc ..... 14
- \oplus ..... 7467, 7468

## P

- \PackageError ..... 8173
- \pagebreak ..... 8322, 8745
- \pagenumbering ... 1450, 1460, 1471, 1482
- \par ..... 131, 353, 1295, 1299, 1905, 7146, 7173, 7216, 7273, 7934, 7948, 7971, 7976, 7984, 7989, 7996, 8006, 8311, 8321, 8400, 8405, 8457, 8462, 8506, 8511, 8555, 8560, 8572, 8615, 8728, 8733, 8743, 8745
- \paragraph ..... 26, 71
- \parsep ..... 7182
- \part ..... 26, 71, 7830, 7831
- \partname ..... 7749, 7827, 7828
- \parttitlename ..... 7749
- \PassOptionsToClass ..... 7672, 7673
- \PassOptionsToPackage ..... 7, 7674, 7675, 7686, 7689, 7714, 7715, 7732, 8205, 8671
- \pause ..... 7969
- \PDF ..... 14, 25
- \pdfbookmark ..... 7781
- \pdfdest ..... 1529

peek commands:

- \peek\_charcode:NTF ..... 3555, 5033, 5046, 5109, 5111, 5233, 5240, 6267, 6273, 6414, 6425
- \peek\_charcode\_remove:NTF ..... 5032, 5108, 5224, 5229, 5231, 5238, 5311, 5699, 6272, 6522
- \phantom ..... 8652
- \Pi ..... 7489
- \plus ..... 38–40, 43, 79–81
- \prematurestop ..... 97, 7837
- \premise ..... 50, 51, 90, 6912, 7049, 7056

prg commands:  
   \prg\_new\_conditional:Nnn .....  
     ..... 225, 275, 527,  
       532, 741, 751, 2113, 2523, 2527,  
       3628, 3639, 3643, 5454, 5567, 5578  
   \prg\_new\_protected\_conditional:Nnn  
     ..... 761  
   \prg\_return\_false: .. 227, 277, 530,  
     535, 742, 746, 752, 754, 777, 781,  
     2114, 2525, 2529, 3629, 3644, 5472,  
     5476, 5573, 5574, 5575, 5581, 5582  
   \prg\_return\_true: .. 227, 277,  
     530, 535, 744, 746, 754, 781, 2114,  
     2525, 2529, 3640, 5470, 5573, 5581  
\printexcursion ..... 98  
\printexcursions .. 8121, 8130, 8151, 8159  
problem (env.) ..... 1  
problem commands:  
  \g\_problem\_id\_counter .....  
    ..... 8356, 8362, 8364,  
    8419, 8421, 8468, 8470, 8517, 8519  
  \l\_problem\_inputproblem\_keys\_tl ..  
    ..... 8262, 8263, 8265, 8350  
  \problem\_mcc\_box\_tl ..... 8609, 8634  
\problemheader ..... 8311, 8326  
problems internal commands:  
  \c\_\_problems\_boxed\_bool ..... 8201  
  \\_\_problems\_do\_yes\_param:Nn .. 8587  
  \\_\_problems\_exnote\_start:n ..  
    ..... 8464, 8483, 8486  
  \\_\_problems\_gnote\_start:n ..  
    ..... 8513, 8532, 8535  
  \c\_\_problems\_gnotes\_bool ..  
    ..... 8191, 8534, 8545  
  \\_\_problems\_hint\_start:n ..  
    ..... 8415, 8434, 8437  
  \c\_\_problems\_hints\_bool ..  
    ..... 8193, 8436, 8447  
  \c\_\_problems\_min\_bool 8199, 8315, 8568  
  \c\_\_problems\_notes\_bool ..  
    ..... 8189, 8485, 8496  
  \l\_\_problems\_path\_seq ... 8272, 8273  
  \l\_\_problems\_prob\_imports\_tl .. 8248  
  \l\_\_problems\_prob\_refnum\_int .. 8249  
  \c\_\_problems\_pts\_bool 8197, 8312, 8563  
  \\_\_problems\_record\_problem: ..  
    ..... 8300, 8334  
  \\_\_problems\_solution\_start:n ..  
    ..... 8358, 8377, 8380  
  \c\_\_problems\_solutions\_bool ..  
    ..... 8195, 8211  
  \c\_\_problems\_test\_bool ..  
    ..... 8203, 8322, 8829, 8831, 8832

\ProcessKeysOptions .....  
  ..... 32, 7678, 7718, 8210, 8674, 8848  
\ProcessOptions ..... 8  
\prod ..... 7496, 7507  
\prop ..... 41  
prop commands:  
  \prop\_clear:N .. 1167, 2857, 2858, 6613  
  \prop\_const\_from\_keyval:Nn .. 103, 115  
  \prop\_gclear:N ..  
    ..... 2296, 2297, 2298, 2408, 5369  
  \prop\_get:NnN ..... 1201  
  \prop\_get:NnNTF ..  
    ..... 182, 896, 897, 1791, 1892,  
    1893, 1968, 2911, 2932, 5455, 8172  
\prop\_gput:Nnn ..  
  2436, 2557, 2569, 2571, 2604, 5398,  
  5434, 5440, 5458, 7017, 8166, 8760  
\prop\_gset\_eq:NN .. 1184, 1215, 1218  
\prop\_gset\_from\_keyval:Nn ..  
  ..... 1187, 1212, 2401, 2403, 5382  
\prop\_if\_exist:NTF ..  
  ..... 894, 1093, 1110, 1594,  
  1688, 1790, 1852, 1965, 3056, 5381  
\prop\_if\_in:NnTF ..  
  131, 166, 798, 6573, 6630, 6660, 6674  
\prop\_item:Nn ..  
  ..... 132, 914, 1094, 1207, 1612,  
  1691, 1854, 1857, 1866, 2026, 2040,  
  2055, 2837, 2993, 2998, 3059, 3084,  
  3095, 3110, 6565, 6605, 6633, 8169  
\prop\_map\_break: ..... 2597  
\prop\_map\_break:n 2616, 2652, 2655,  
  2733, 3017, 4030, 4878, 4881, 7004  
\prop\_map\_function:NN .. 2394, 2924  
\prop\_map\_inline:Nn .. 2404, 2581,  
  2614, 2658, 2676, 2679, 2701, 2704,  
  2742, 2888, 2907, 2947, 2975, 2979,  
  4046, 4871, 6227, 6453, 7001, 8772  
\prop\_new:N ..... 8164, 8754  
\prop\_put:Nnn .. 1175, 1176, 1177,  
  1178, 1179, 2875, 2879, 3347, 3381,  
  4764, 4959, 6639, 6662, 6676, 8767  
\prop\_remove:Nn ..... 3443  
\prop\_set\_eq:NN 1082, 1199, 1203, 1603  
\prop\_to\_keyval:N ..  
  ... 1185, 1188, 1213, 2373, 2374,  
  2375, 2382, 2386, 2902, 2905, 5383  
\protect ..... 7779, 8175  
\protected ..... 112,  
  130, 351, 565, 566, 1263, 5906, 5920  
\providecommand .. 2024, 2031, 7845, 8853  
\ProvidesExplClass ..... 5, 7661  
\ProvidesExplPackage ..  
  ..... 16, 7705, 8184, 8664, 8839

\pts . . . . .	8562
pts . . . . .	99, 105
<b>Q</b>	
\quad . . . . .	7758, 7762, 7766, 7770, 7774, 8643, 8659, 8736, 8739
quark commands:	
\quark_new:N . . . . .	2137, 2420
quark internal commands:	
\q__stex_smsmode_break . . . . .	2137, 2140, 2200
<b>R</b>	
\rangleangle . . . . .	7476
\realization . . . . .	3514, 3516
\realize . . . . .	62, 63, 3621, 3623, 3625
\ref . . . . .	73, 1635
\refstepcounter . . . . .	8270, 8720
\relax . . . . .	352, 633, 634, 1445, 1455, 1529, 1686, 2066, 2144, 2182, 2186, 2411, 2412, 3948, 3955, 8055, 8057
\renamedecl . . . . .	63, 118, 2969, 3367, 3451
\renewcommand . . . . .	7854, 7955, 7965
\renewenvironment . . . . .	7908, 7952, 7970, 7992, 8015, 8017
repo commands:	
\repo_prop . . . . .	127, 128
\reqpts . . . . .	105
\requiremodule . . . . .	82, 83, 2965, 3316
\RequirePackage . . . . .	4, 10, 15, 17, 18, 140, 142, 191, 194, 5846, 7662, 7680, 7702, 7706, 7720, 7733, 7734, 8046, 8185, 8216, 8665, 8675, 8840, 8851, 8856, 8857
\resizebox . . . . .	8777, 8865, 8871, 8875
\rhd . . . . .	7910, 7913
\right . . . .	5851
\Rightarrow . . . . .	7408
\rightmargin . . . . .	7183
rustex commands:	
\rustex_direct_HTML:n . . . . .	7977, 7985, 7997, 8007
\rustex_if:TF . . . . .	7975, 7976, 7983, 7984, 7995, 7996, 8005, 8006
\rustexBREAK . . . . .	2110
<b>S</b>	
sassertion (env.) . . . . .	88
\scriptsize . . . . .	7907
\scriptstyle . . . . .	7913
sdefinition (env.) . . . . .	88
\second . . . . .	126
\section . . . . .	25, 26, 71
\sectiontitleemph . . . . .	7725, 7784
sectocframes . . . . .	94
\selectlanguage . . . . .	129, 135
seq commands:	
\seq_clear:N . . . . .	176, 676, 714, 1067, 1971, 2154, 2665, 2690, 2715, 2820, 2859, 4163, 4508, 4669, 5523, 5587, 5652, 5661, 6093, 6144, 6194, 6352, 6614, 6615, 6735, 6771, 7546
\seq_count:N . . . . .	2010, 2085, 3524, 6805
\seq_gclear:N . . . . .	2148, 2149, 5370
\seq_gclear_new:N . . . . .	977, 978
\seq_get:NN . . . . .	1008
\seq_get_right:NN . . . . .	152, 8273
\seq_gpop>NN . . . . .	1003
\seq_gpush:Nn . . . . .	991
\seq_gput_left:Nn . . . . .	1538, 1543
\seq_gput_right:Nn . . . . .	1541, 2104, 2123, 5399
\seq_gset_eq:NN . . . . .	990, 1006, 1010, 2847, 2848, 2849, 7643
\seq_gset_split:Nnn . . . . .	5388
\seq_if_empty:NTF . . . . .	163, 712, 730, 742, 752, 781, 827, 833, 838, 843, 1002, 1005, 1111, 1939, 1992, 4007, 6820, 7153, 7204, 7360, 7554, 7642
\seq_if_empty_p:N . . . . .	768, 769, 1975
\seq_if_exist:NTF . . . . .	1583, 5387
\seq_if_in:NNTF . . . . .	1536, 1537, 1650, 1652, 2248, 2257, 2634, 2674, 2699, 2740, 6465, 6576, 6699
\seq_item:Nn . . . . .	743, 753, 2011, 2086, 4276, 4282, 5456, 6806
\seq_map_break: . . . . .	1585, 1589, 2652
\seq_map_break:n . . . . .	1589, 2655, 4030
\seq_map_function:NN . . . . .	716, 720, 6364
\seq_map_inline:Nn . . . . .	258, 911, 915, 1588, 1623, 1942, 1995, 2163, 2657, 4044, 4670, 4697, 5662, 6157, 6204, 6213, 6741, 6879, 6906, 7224, 7279, 7555
\seq_new:N . . . . .	219, 1496, 1542, 2094, 2117, 2267
\seq_pop>NN . . . . .	713
\seq_pop_left:NN . . . . .	162, 696, 774, 775, 826, 831, 837, 842, 912, 916, 918, 1980, 3525, 3527, 3533, 3537
\seq_pop_left:NNTF . . . . .	1171, 4250, 4252, 4260
\seq_pop_right:NN . . . . .	154, 164, 731, 786, 788, 793, 795, 797, 1139, 2342, 3052, 4006, 4528, 4530, 6355
\seq_push:Nn . . . . .	719
\seq_put_left:Nn . . . . .	697, 717, 2675, 2700, 6661, 6675

\seq_put_right:Nn . . . . .	158, 184, 222, 734, 789, 799, 802, 974, 1978, 1981, 1986, 2301, 2331, 2344, 2636, 2741, 2823, 2954, 4233, 4240, 4261, 4263, 4510, 4672, 4674, 5554, 5590, 5626, 5633, 5658, 5664, 5668, 5675, 6094, 6149, 6153, 6199, 6631, 6738, 6774, 6788	6356	\slidewidth . . . . .	7859, 7943, 7973, 8055, 8057, 8061
\seq_reverse:N . . . . .	722, 762, 763, 785, 792, 910, 973, 1972, 4679, 4707, 5673	19, 23, 78, 5877	\smacro . . . . .	81, 82
\seq_set_eq:NN . . . . .	722, 762, 763, 785, 792, 910, 973, 1972, 4679, 4707, 5673	19, 23, 78, 5877	\small . . . . .	7177
\seq_set_split:Nnn . . . . .	153, 677, 787, 794, 825, 830, 1131, 1170, 1973, 2341, 3051, 3523, 3531, 4005, 4249, 4253, 4527, 4529, 6354, 7547	19, 78, 5877	\smallskip . . . . .	7173, 8313, 8316, 8400, 8405, 8457, 8462, 8506, 8511, 8555, 8560, 8733
\seq_use:Nn . . . . .	189, 192, 195, 759, 789, 802, 830, 1173, 2346, 3053, 3529, 3535, 3539, 4011, 4319, 4513, 4656, 4680, 5389, 5512, 6821, 7154, 7205, 7361	1, 8356	\smodule (env.) . . . . .	75, 2491
\l_tmpa_seq . . . . .	153, 154, 158, 162, 163, 164, 176, 184, 189, 192, 195, 4508, 4510, 4513, 4527, 4528, 4529, 4530, 4669, 4672, 4674, 4679, 4707, 6352, 6354, 6355, 6356, 6364, 7546, 7547, 7554, 7555	2966	\smodule . . . . .	2966
\seqmap . . . . .	85, 5125, 5580	19, 23, 78, 5877	\Sn . . . . .	19, 78, 5877
\setbox . . . . .	124, 8017, 8032, 8382, 8439, 8488, 8537	19, 78, 5877	\sn . . . . .	19, 78, 5877
\setcounter . . . . .	8718, 8721	19, 78, 5877	\Sns . . . . .	19, 78, 5877
\setkeys . . . . .	2028, 2042, 2057, 8059, 8860	19, 78, 5877	\sns . . . . .	19, 78, 5877
\setlength . . . . .	7181, 7182, 7183, 7859, 7860, 7864, 7917, 7918, 7919	1, 8356	\solution (env.) . . . . .	1, 8356
\setlicensing . . . . .	96	8302, 8408, 8577	\solution . . . . .	8302, 8408, 8577
\setmetatheory . . . . .	2446	solutions . . . . .	99, 105	
\setnotation . . . . .	32, 80, 4393	\solutionsfalse . . . . .	8214, 8413	
\setsectionlevel . . . . .	26, 71, 110, 1404, 7736, 7738, 7795, 7797	\solutionstrue . . . . .	8212, 8410	
\setSGvar . . . . .	98, 8165, 8175	\source . . . . .	126	
\setslidelogo . . . . .	96	\sparagraph (env.) . . . . .	88	
\setsource . . . . .	96	\spfblock . . . . .	7236, 7436	
\sexample (env.) . . . . .	88	\spfjust . . . . .	7237, 7442, 7445	
\sf . . . . .	8077	\spfsketch . . . . .	7145	
\sffamily . . . . .	8086	\spfsketchenvautorefname . . . . .	7163	
\sfragment (env.) . . . . .	71	\spfstep . . . . .	7231, 7338	
sfragment commands:		\spfstepautorefname . . . . .	7272, 7353	
\_\sfragment\_do\_level:nm . . . . .	1293, 1305, 1321, 1325, 1329, 1330, 1331, 1332, 1333, 7777	\spfstepenvautorefname . . . . .	7353	
\_\sfragment\_end: . . . . .	1301, 1315, 1347	\sproblem (env.) . . . . .	8259	
\skipfragment . . . . .	71, 1388, 1392, 1396	\sproblemautorefname . . . . .	8327	
\slideframewidth . . . . .	7863, 7864, 7940, 8055	\sproof (env.) . . . . .	90, 7179	
\slideheight . . . . .	7860	\sproofautorefname . . . . .	7251	
slides . . . . .	94	\sproofend . . . . .	7165, 7253, 7270	
		\square . . . . .	7166, 8610	
		\sr . . . . .	19, 23, 77, 5877	
		\sref . . . . .	73, 74, 88, 1547, 8125	
		\sreflabel . . . . .	73, 75, 110, 1496	
		\srefsetin . . . . .	73, 1569	
		\srefsym . . . . .	78, 1814	
		\srefsymuri . . . . .	78, 1845	
		\startsolutions . . . . .	100, 8409	
		\stepcounter . . . . .	1385, 7778, 7954	
		\sTeX . . . . .	13, 14, 70	
		\stex . . . . .	14, 23, 70	
		stex commands:		
		\l_stex_aB_args_seq . . . . .		
		4656, 4670, 4679, 4680, 4697, 4707, 5512, 5523, 5554, 5587, 5590, 5626, 5633, 5652, 5658, 5662, 5673, 5675		
		\stex_activate_module:n . . . . .		
		111, 2312, 2361, 2633, 2633, 2645, 2815, 3158, 3256, 3299, 3499, 6165, 6221, 6746, 7592		
		\stex_add_definiens:nn . . . . .		
		2971, 6991, 6999, 7226, 7281		

```

\stex_add_definiens_inner:nnnnnnnn
    ..... 7004, 7015
\stex_add_module_notation:nnnnn  111
\l_stex_all_modules_seq .....
    ..... 111, 2154, 2267,
    2301, 2331, 2634, 2636, 2657, 4044
\l_stex_allow_semantic_bool .....
    4909, 5051, 5129, 5130, 5133, 5159,
    5193, 5359, 5368, 5484, 5495, 5593,
    5678, 5736, 5747, 5780, 5809, 5970,
    6051, 6458, 6489, 6506, 6551, 6932
\stex_annotation:nn .....
    ..... 332, 335, 1276, 1285, 1297,
    1739, 1956, 1957, 3340, 3768, 3771,
    3778, 3783, 3785, 4322, 4328, 4337,
    4348, 4359, 4360, 4363, 4364, 4368,
    4813, 4816, 4823, 4827, 4830, 5006,
    5009, 5016, 5020, 5023, 5054, 5500,
    5507, 5601, 5710, 5723, 5740, 5765,
    5773, 5794, 5802, 5996, 6378, 6386,
    6391, 6599, 6845, 6866, 6943, 7009,
    7042, 7054, 7151, 7193, 7196, 7238,
    7292, 7301, 7305, 7358, 7427, 7443,
    7552, 7557, 8115, 8629, 8636, 8645
\stex_annotation:nnn ..... 129, 130
\stex_annotation_force_break:n ...
    ..... 321, 322, 328,
    1298, 3339, 3765, 3783, 4810, 4828,
    5004, 5021, 5508, 5606, 5715, 5728,
    5745, 5770, 5778, 5799, 5807, 6390,
    6845, 6867, 7010, 7043, 7553, 8645
\stex_annotation_invisible:n .....
    ..... 130, 323,
    325, 1355, 1377, 2508, 2765, 2809,
    5156, 5412, 7191, 7550, 7587, 8287
\stex_annotation_invisible:nn .....
    ..... 1249, 1416, 1424,
    1430, 1436, 1897, 1905, 3037, 3261,
    3302, 3338, 3372, 3413, 3423, 3750,
    4315, 4791, 4988, 5746, 5779, 5808,
    6159, 6215, 6743, 7025, 7541, 7588
\stex_annotation_invisible:nnn .. 130
\l_stex_argnames_seq ..... 113
\stex_args_end: ..... 5373,
    5397, 5400, 5922, 5926, 5976, 6938
\stex_assign_do:n .....
    ..... 2830, 3320, 3325, 3550
\l_stex_assoc_args_count .....
    ..... 113, 3797, 3801, 3843, 3844
\l_stex_brackets_dones_bool .....
    .. 4577, 5826, 5829, 5830, 5854, 5855
\stex_capitalize:n .....
    ..... 1287, 5913, 5916, 5959, 6972
\stex_check_term:n .....
    ..... 113, 114, 2153, 3332,
    3648, 3649, 3657, 3863, 4295, 7449
\c_stex_check_terms_bool .....
    ..... 27, 3635, 3638, 7452
\stex_close_module: 111, 2368, 2368,
    2517, 2772, 7533, 7600, 7602, 8307
\l_stex_current_archive ..... 1079
\l_stex_current_archive_prop ...
    ..... 123, 128, 914, 924,
    929, 1082, 1093, 1094, 1103, 1192,
    1594, 1595, 1688, 1691, 1790, 1791,
    1852, 1854, 1857, 1892, 1893, 1965,
    1968, 2026, 2040, 2055, 3056, 3059
\l_stex_current_args_tt ...
    ..... 120, 4499, 4503, 5182, 5372, 5373, 5376
\l_stex_current_arity_str .....
    ..... 120, 4509, 5071, 5098,
    5100, 5181, 5281, 5299, 5300, 5331
\l_stex_current_doc_uri 128, 958,
    960, 961, 1506, 1511, 1513, 1517,
    1583, 1585, 1625, 1642, 1643, 6824
\l_stex_current_domain_str .....
    ..... 118, 2779, 2782, 2784,
    2792, 2797, 2806, 2815, 2846, 2861,
    2873, 2910, 2922, 3390, 3438, 3459,
    3479, 3499, 3503, 3573, 3593, 3614
\g_stex_current_file .....
    ..... 126–128, 152, 923, 925,
    930, 978, 981, 983, 985, 986, 987,
    990, 991, 1006, 1009, 1010, 1598,
    1696, 2320, 3066, 3072, 3073, 3121,
    3140, 3184, 3210, 7642, 7643, 8272
\l_stex_current_language_str ...
    ..... 128, 129, 128, 130, 165, 170, 2502,
    3171, 3174, 3198, 3201, 7581, 8281
\l_stex_current_module ..... 112, 114
\l_stex_current_module_str .....
    ..... 111, 132,
    2155, 2266, 2283, 2300, 2301, 2316,
    2329, 2330, 2331, 2332, 2371, 2372,
    2373, 2374, 2375, 2381, 2383, 2387,
    2392, 2394, 2501, 2511, 2524, 2532,
    2536, 2557, 2563, 2569, 2571, 2581,
    2588, 2593, 2602, 2604, 2614, 2638,
    2641, 2760, 2771, 2805, 2816, 2913,
    2916, 3345, 3379, 3406, 3415, 3425,
    3704, 3742, 3751, 3847, 3908, 3917,
    3926, 3932, 4456, 4475, 4476, 6054,
    6088, 6093, 6094, 6114, 6206, 6228,
    6789, 6812, 7000, 7001, 7002, 7003,
    7016, 7017, 7571, 7572, 7575, 7580,
    7599, 7601, 7614, 7628, 8280, 8293
\l_stex_current_ns_uri ..... 128,
```

964, 965, 2338, 2481, 3122, 3127,  
 3128, 3141, 3146, 3147, 7453, 7534  
 $\backslash l_{stex\_current\_redo\_tl}$  . . . . .  
 . . . . . 5163, 5176, 5187, 5190,  
 5197, 5327, 6323, 6361, 6396, 6546  
 $\backslash l_{stex\_current\_return\_tl}$  . . . . .  
 120, 3955, 5183, 5249, 5261, 5282, 5332  
 $\backslash stex\_current\_section\_level$  . . . . .  
 . . . . . 1272, 1278, 1287, 1313, 7789  
 $\backslash l_{stex\_current\_symbol\_str}$  . . . . .  
 . . . . . 120, 4147, 4177,  
 4296, 4336, 4347, 4445, 4464, 4527,  
 4753, 4905, 4907, 4913, 4948, 5038,  
 5041, 5056, 5084, 5085, 5168, 5180,  
 5202, 5218, 5247, 5248, 5251, 5257,  
 5273, 5274, 5277, 5366, 5448, 5475,  
 5492, 5594, 5595, 5638, 5679, 5680,  
 5712, 5725, 5742, 5767, 5775, 5796,  
 5804, 5857, 5862, 5935, 5948, 5961,  
 5971, 5984, 5992, 5996, 5998, 6053,  
 6404, 6405, 6933, 6941, 6943, 6945  
 $\backslash l_{stex\_current\_term\_tl}$  4910, 5131,  
 5185, 5493, 5596, 5597, 5598, 5681,  
 5709, 5722, 5737, 5748, 5764, 5781,  
 5793, 5810, 5979, 6397, 6403, 6593  
 $\backslash stex\_current\_this$ : . 5177, 6050, 6060  
 $\backslash l_{stex\_current\_this\_tl}$  . . . . .  
 . . . . . 6052, 6056, 6070, 6073  
 $\backslash l_{stex\_current\_type\_tl}$  . . . . .  
 . . . . . 120, 5053, 5184, 6265,  
 6320, 6327, 6330, 6334, 6649, 6698  
 $\backslash stex\_deactivate\_macro:Nn$  . . . . .  
 . . . . . 131, 342, 342, 2553,  
 2960, 2961, 2962, 2963, 2964, 2965,  
 2966, 3247, 3316, 3450, 3451, 3452,  
 3469, 3490, 3514, 3579, 3600, 3621,  
 3715, 3940, 4154, 4494, 6101, 6175,  
 6237, 6953, 6959, 6967, 6975, 6996,  
 7029, 7047, 7056, 7331, 7404, 7424,  
 7429, 7436, 7445, 7543, 7567, 7640,  
 8408, 8576, 8577, 8586, 8641, 8661  
 $\backslash stex\_debug:nn$  . . . . . 109, 58, 58,  
 95, 99, 170, 189, 240, 507, 632, 699,  
 706, 853, 923, 926, 961, 965, 976,  
 1017, 1025, 1053, 1058, 1078, 1081,  
 1114, 1157, 1185, 1196, 1206, 1226,  
 1242, 1257, 1512, 1577, 1582, 1622,  
 1626, 1642, 1645, 1651, 1653, 1657,  
 1660, 1666, 1668, 1671, 1680, 1681,  
 1707, 1754, 1756, 1758, 1850, 1856,  
 1861, 1868, 2206, 2208, 2224, 2226,  
 2230, 2247, 2249, 2256, 2258, 2280,  
 2293, 2306, 2308, 2330, 2370, 2448,  
 2450, 2459, 2563, 2588, 2596, 2600,  
 2635, 2901, 2933, 2937, 2940, 3050,  
 3055, 3057, 3065, 3072, 3077, 3081,  
 3120, 3139, 3156, 3162, 3169, 3185,  
 3196, 3211, 3229, 3237, 3239, 3326,  
 3333, 3370, 3391, 3407, 3411, 3421,  
 3439, 3441, 3864, 3866, 3868, 3870,  
 3987, 4004, 4043, 4054, 4056, 4205,  
 4237, 4248, 4298, 4475, 4479, 4555,  
 4564, 4569, 4590, 4739, 4858, 4898,  
 5117, 5122, 5218, 5223, 5228, 5247,  
 5250, 5253, 5273, 5325, 5334, 5358,  
 5366, 5482, 5833, 6112, 6184, 6264,  
 6804, 6807, 6809, 6813, 6989, 7002,  
 7016, 7051, 7225, 7280, 7379, 7576  
 $\backslash c_{stex\_debug\_clist}$  . . . . .  
 . . . . . 22, 31, 59, 62, 83, 87, 89, 93, 97  
 $\backslash c_{stex\_default\_metatheory}$  2152, 7536  
 $\backslash l_{stex\_default\_notation}$  . . . . .  
 . . . . . 4507, 4512, 4515, 4517, 4518,  
 4535, 5088, 5262, 5267, 5288, 5622  
 $\backslash stex\_do\_default\_notation:$  . . . . .  
 . . . . . 4497, 5087, 5260, 5621  
 $\backslash stex\_do\_default\_notation\_op:$  . . . . .  
 . . . . . 4497, 4498, 4533, 5285  
 $\_stex\_do\_deprecation:n$  . . . . .  
 . . . . . 109, 425, 425, 2286, 3691, 3792  
 $\backslash stex\_do\_for\_list:$  . . . . .  
 . . . . . 2970, 6770, 6785, 7149, 7220, 7275  
 $\backslash stex\_do\_id:$  . . . . . 121, 430,  
 430, 1345, 6849, 6864, 7150, 7229,  
 7289, 7298, 7307, 7387, 8299, 8724  
 $\backslash stex\_do\_up\_to\_module:n$  . . . . .  
 . . . . . 112, 2531, 2531, 2534, 2541, 3298, 6252  
 $\backslash l_{stex\_dochader\_sect}$  . . . . .  
 . . . . . 110, 1270, 1296, 1312, 1320, 1322,  
 1326, 1350, 1362, 1364, 1367, 1370,  
 1376, 1389, 1391, 1395, 1406, 1407,  
 1408, 1409, 1410, 1411, 1413, 1416,  
 1422, 1426, 1432, 1436, 7781, 7788  
 $\backslash stex\_eat\_exclamation\_point:$  . . . . .  
 . . . . . 3746, 5251, 5262, 5698, 5700  
 $\backslash stex\_end:$  . . . . . 414, 422, 2868, 2876  
 $\backslash stex\_every\_file:$  . . . . .  
 . . . . . 992, 995, 1000, 1012, 7646  
 $\backslash stex\_every\_module:n$  . . . . .  
 . . . . . 111, 2268, 2271,  
 2555, 3286, 3470, 3491, 3515, 3580,  
 3601, 3622, 3716, 3941, 4155, 4495,  
 6102, 6176, 6239, 6249, 7544, 7568  
 $\backslash g_{stex\_every\_module\_tl}$  . . . . .  
 . . . . . 2269, 2272, 2284  
 $\backslash l_{stex\_every\_symbol\_tl}$  5132, 5137,  
 5139, 5140, 5146, 5147, 5150, 5188

```

\stex_execute_in_module:n . . . .
    . . . . . 111, 2285, 2359, 2539,
    2539, 2543, 2574, 2606, 3255, 3498,
    3925, 6092, 6152, 6164, 6203, 6220
\stex_fatal_error:n 131, 74, 74, 1112
\stex_fatal_error:nnn . . . .
    . . . . . 131, 74, 77, 80, 1123, 2016, 2088
\l_stex_feature_name_str . . . .
    . . . . . 118, 2812, 2816,
    2845, 2916, 2921, 2937, 2938, 2940,
    2941, 2984, 3329, 3345, 3379, 3383
\stex_file_in_smsmode:nn . . . .
    . . . . . 124, 2128, 2147, 2176, 2322, 3225
\stex_file_resolve:Nn . . . . 126,
    127, 681, 686, 702, 709, 871, 972,
    981, 983, 986, 1041, 1069, 1071,
    1597, 1615, 1705, 3073, 3167, 3194
\stex_file_set:Nn .. 126, 127, 675,
    675, 680, 693, 704, 867, 914, 1009, 1066
\stex_file_split_off_ext:NN . . .
    . . . . . 126, 784, 784, 889, 2320, 3066
\stex_file_split_off_lang:NN . . .
    . . . . . 126, 784, 791, 886, 2321, 3067, 8272
\stex_file_use:N . . . . . 126,
    127, 699, 706, 758, 758, 835, 840,
    845, 899, 905, 920, 923, 976, 987,
    991, 1046, 1066, 1072, 1077, 1121,
    1124, 1157, 1158, 1160, 1194, 1598,
    1599, 1611, 1690, 1696, 1701, 1706,
    1708, 1853, 1860, 1865, 1877, 1880,
    1904, 1909, 1920, 1976, 1984, 2022,
    2026, 2036, 2040, 2050, 2055, 2323,
    3069, 3072, 3073, 3075, 3096, 3111,
    3121, 3140, 3168, 3184, 3195, 3210
\stex_filestack_pop: . . . .
    . . . . . 126, 1001, 1001, 1026, 1033, 2173
\stex_filestack_push:n . . . .
    . . . . . 126, 979, 979, 1019, 1021, 2157
\l_stex_fors_seq . . . . . 2820,
    2823, 6771, 6774, 6788, 6805, 6806,
    6820, 6821, 6879, 6906, 7153, 7154,
    7204, 7205, 7224, 7279, 7360, 7361
\stex_get_env:Nn . . . . .
    . . . . . 131, 44, 45, 53, 81, 294, 598,
    604, 968, 970, 1037, 1039, 1044, 3632
\stex_get_in_morphism:n . . . .
    . . . . . 120, 2822, 2973, 2973, 3319, 3363, 3545
\l_stex_get_mathstructure:n . . .
    . . . . . 2781, 6129, 6134
\stex_get_mathstructure:n . . .
    . . . . . 6128, 6128, 6146, 6195, 6734
\l_stex_get_structure_module_str ..
    . . . . . 2782, 6130, 6135, 6139, 6153, 6204
\l_stex_get_symbol:n . . . . .
    . . . . . 3959, 3965, 6136
\stex_get_symbol:n . . . . .
    . . . . . 113, 114, 120, 1815, 3958,
    3958, 4119, 4394, 4899, 5206, 5892,
    5900, 5911, 6773, 6931, 6982, 7035
\l_stex_get_symbol_args_tl . . .
    . . . . . 113, 115,
    223, 3022, 3352, 3735, 3854, 3856,
    3857, 3993, 4034, 4066, 4077, 4079,
    4092, 4503, 4768, 4781, 4863, 4963,
    4976, 5211, 5376, 6117, 6797, 7621
\l_stex_get_symbol_arity_int . . .
    . . . . . 113, 115, 223,
    3021, 3351, 3734, 3776, 3800, 3806,
    3810, 3814, 3818, 3822, 3826, 3830,
    3834, 3838, 3841, 3842, 3843, 3844,
    3855, 3913, 3985, 3992, 4033, 4065,
    4165, 4200, 4232, 4239, 4259, 4309,
    4426, 4738, 4767, 4780, 4821, 4862,
    4962, 4975, 5014, 5210, 6796, 7620
\l_stex_get_symbol_def_tl . . .
    . . . . . 113, 2841, 3023,
    3327, 3994, 4035, 4067, 4864, 5212
\l_stex_get_symbol_invoke_cs . . .
    . . . . . 113, 3026, 3356,
    3997, 4038, 4070, 4867, 5215, 6138
\l_stex_get_symbol_macro_str . . .
    . . . . . 120, 3020, 3350
\l_stex_get_symbol_mod_str . . 113,
    1817, 2824, 2834, 3018, 3334, 3338,
    3348, 3373, 3382, 3917, 3923, 3966,
    3990, 4031, 4063, 4125, 4129, 4144,
    4307, 4396, 4400, 4404, 4408, 4415,
    4419, 4476, 4483, 4860, 5208, 5972,
    6775, 6934, 6984, 7037, 7630, 7636
\l_stex_get_symbol_name_str . . .
    . . . . . 113, 2824, 2835, 2974, 2978,
    2982, 3019, 3326, 3328, 3334, 3338,
    3348, 3370, 3373, 3382, 3383, 3918,
    3923, 3960, 3967, 3991, 4032, 4064,
    4125, 4129, 4143, 4144, 4307, 4396,
    4400, 4404, 4408, 4415, 4419, 4433,
    4438, 4441, 4477, 4483, 4861, 4887,
    4889, 4895, 4897, 5209, 5902, 5913,
    5935, 5946, 5948, 5959, 5961, 5972,
    5973, 5974, 5975, 6137, 6775, 6934,
    6935, 6936, 6937, 6964, 6972, 6980,
    6984, 7025, 7033, 7037, 7631, 7636
\stex_get_symbol_or_var:n . . .
    . . . . . 114, 4857, 4894
\l_stex_get_symbol_return_tl . . .
    . . . . . 113, 3025, 3355, 3996,
    4037, 4069, 4740, 4866, 4929, 5214
\l_stex_get_symbol_type_tl . . .
    . . . . . 113, 3024, 3354, 3995, 4036, 4068,

```

4865, 5213, 6139, 6147, 6197, 6736  
`\l_stex_get_symbol_uri_str` ... 1817  
`\stex_get_var:n` ..... 114, 4432,  
   4857, 4886, 5933, 5944, 5957, 7023  
`\c_stex_home_file` ..... 127, 1035  
`\stex_if_check_terms:` .....  
   ..... 113, 3628, 3639, 3643  
`\stex_if_check_terms:TF` 113, 3627,  
   3648, 4121, 4435, 4742, 4938, 4941  
`\stex_if_check_terms_p:` ... 113, 3627  
`\stex_if_do_html:` ..... 275  
`\stex_if_do_html:TF` ..... 129,  
   272, 284, 1248, 1275, 1284, 1528,  
   2499, 2519, 2757, 2773, 2803, 2851,  
   3036, 3260, 3301, 3337, 3371, 3725,  
   3909, 3921, 4123, 4482, 4736, 4744,  
   4790, 4937, 6158, 6214, 6742, 6800,  
   7024, 7221, 7239, 7249, 7266, 7276,  
   7310, 7317, 7319, 7357, 7578, 7603,  
   7635, 8278, 8309, 8367, 8376, 8386,  
   8392, 8424, 8433, 8443, 8449, 8473,  
   8482, 8492, 8498, 8522, 8531, 8541,  
   8547, 8573, 8582, 8628, 8635, 8644  
`\stex_if_do_html_p:` ..... 129  
`\stex_if_file_absolute:N` 126, 741, 751  
`\stex_if_file_absolute:NTF` .....  
   ..... 126, 739, 985, 1070  
`\stex_if_file_absolute_p:N` . 126, 739  
`\stex_if_file_starts_with:NN` 126, 761  
`\stex_if_file_starts_with:NNTF` ..  
   ..... 126, 761, 1193  
`\stex_if_html_backend:TF` .....  
   ..... 129, 289, 314,  
   321, 330, 613, 1248, 1292, 1354,  
   1373, 1420, 1927, 1954, 3627, 5160,  
   5194, 5706, 5762, 5791, 5969, 5995,  
   6843, 6853, 6929, 7681, 7721, 7846,  
   7926, 7974, 7982, 7994, 8004, 8114  
`\stex_if_html_backend_p:` ... 129, 289  
`\stex_if_in_module:` ..... 2523  
`\stex_if_in_module:TF` .....  
   ..... 111, 2275, 2523, 2539  
`\stex_if_in_module_p:` ... 111, 2523  
`\stex_if_module_exists:n` ..... 2527  
`\stex_if_module_exists:nTF` .....  
   111, 2292, 2305, 2527, 3124, 3127,  
   3143, 3146, 3231, 6148, 6198, 6737  
`\stex_if_module_exists_p:n` 111, 2527  
`\stex_if_smsmode:` ..... 2113  
`\stex_if_smsmode:TF` .....  
   .... 124, 431, 1240, 1514, 2111,  
   2178, 2311, 2510, 2518, 3040, 3183,  
   3209, 3264, 3305, 3457, 3464, 3477,  
   3485, 3501, 3509, 3571, 3591, 3612,

3702, 3930, 4132, 4485, 6836, 6852,  
   6865, 6878, 6905, 7008, 8292, 8308  
`\stex_if_smsmode_p:` ..... 124, 2111  
`\stex_ignore_spaces_and_pars:` ...  
   . 131, 351, 351, 354, 2549, 2550, 8319  
`\l_stex_import_archive_str` . 119,  
   2451, 2456, 2788, 3033, 3058, 3062,  
   3082, 3083, 3084, 3252, 3279, 3295  
`\stex_import_module_uri:nn` .....  
   ..... 119, 2449, 2785,  
   3031, 3049, 3049, 3250, 3276, 3293  
`\l_stex_import_name_str` .....  
   119, 2453, 2458, 2790, 3035, 3043,  
   3052, 3254, 3268, 3281, 3297, 3309  
`\l_stex_import_ns_str` .... 2459,  
   2462, 2792, 3038, 3042, 3125, 3128,  
   3131, 3144, 3147, 3150, 3158, 3256,  
   3259, 3262, 3267, 3299, 3303, 3308  
`\l_stex_import_path_str` .....  
   ..... 119, 2452, 2457,  
   2789, 3034, 3053, 3064, 3068, 3072,  
   3073, 3074, 3077, 3253, 3280, 3296  
`\stex_import_require_module:nnn` .  
   ..... 119, 2455, 2787,  
   3032, 3087, 3087, 3100, 3251, 3294  
`\stex_import_require_module_-`  
   safe:nnn ..... 3102, 3278  
`\l_stex_import_uri_str` .....  
   ..... 119, 3063, 3084, 3090,  
   3095, 3105, 3110, 3118, 3120, 3124,  
   3125, 3131, 3137, 3139, 3143, 3144,  
   3150, 3169, 3176, 3196, 3231, 3232  
`\stex_in_archive:nn` ... 123, 1086,  
   1086, 1875, 1926, 1951, 2005, 2058  
`\stex_in_invisible_html_bool` ...  
   ..... 3766, 4811, 5704, 5705, 5735  
`\l_stex_in_meta_bool` 2354, 2355, 2360  
`\l_stex_inpararray_bool` ..... 5835  
`\stex_input_with_hooks:n` .....  
   .. 1014, 1877, 1880, 1901, 1918, 1920  
`\_stex_invoke_notation:w` .....  
   ..... 5123, 5240, 5241, 5246, 5283  
`\stex_invoke_outer_field:` .... 6122  
`\stex_invoke_sequence:` .....  
   ..... 4967, 4980, 5031, 5031, 5572  
`\stex_invoke_sequence_in:` .... 121  
`\stex_invoke_sequence_range:` ... 121  
`\stex_invoke_structure:` .....  
   ..... 6090, 6138, 6258  
`\stex_invoke_symbol` ..... 120  
`\stex_invoke_symbol:` .....  
   ..... 114, 120, 3739,  
   4772, 4785, 5217, 5217, 6798, 7625

```

\l_stex_invoke_symbol:nnnnnnnn . .
    ..... 112, 120,
2592, 3974, 3988, 3989, 5158, 5158,
5171, 5207, 6664, 6678, 6686, 6691
\l_stex_invoke_symbol:nnnnnnnn\l_-
    stex_current_symbol_str .... 120
\l_stex_invoke_text_symbol: 3905, 3953
\l_stex_invoke_variable:nnnnnn .. 5192
\l_stex_invoke_variable:nnnnnnN ..
    ..... 120, 4778, 4973, 5192, 5570
\l_stex_is_sequentialized:n ....
    ..... 5564, 5591, 5666, 5676
\l_stex_iterate_break: 112, 2648, 2651
\l_stex_iterate_break:n .. 112, 2648,
    2654, 2731, 3412, 3422, 3437, 4062
\l_stex_iterate_morphisms:nn .....
    ..... 2714, 2714, 3390, 3406
\l_stex_iterate_notations:nn .....
    ..... 115, 2689, 2689, 2910, 6698
\l_stex_iterate_symbols:n .....
    ..... 112, 2647, 2647, 4055
\l_stex_iterate_symbols:nn .....
    ..... 112, 2664, 2664, 2873, 3440, 6106, 6649
\l_stex_key_archive_str .....
    ..... 121, 379, 382, 1561, 1579,
1593, 1602, 1603, 1680, 1687, 1701
\l_stex_key_argnames_clist .... 113
\l_stex_key_args_str .....
    .... 113, 3660, 3665, 3674, 3708,
3752, 3802, 3807, 3811, 3815, 3819,
3823, 3827, 3831, 3835, 3839, 3858,
4460, 4793, 4924, 4925, 4990, 6755
\l_stex_key_argtypes_clist .....
    ..... 3678, 3683, 3782, 3784,
3871, 4826, 4829, 5019, 5022, 6759
\l_stex_key_assoc_str 3663, 3668,
3756, 3757, 4797, 4798, 4994, 4995
\l_stex_key_continues_tl . 7062, 7071
\l_stex_key_def_tl 114, 3676, 3686,
3707, 3736, 3770, 3771, 3867, 3878,
3882, 3902, 4459, 4769, 4782, 4815,
4816, 4964, 4977, 5008, 5009, 6757
\l_stex_key_deprecate_str .....
    ..... 405, 407, 426, 427
\l_stex_key_due_tl .....
    ..... 8699, 8703, 8738, 8739
\l_stex_key_feedback_tl .....
    ..... 8595, 8600, 8622, 8623
\l_stex_key_file_str .....
    ..... 121, 380, 383, 1562,
1578, 1581, 1598, 1613, 1641, 1656,
1667, 1680, 1692, 1696, 1702, 1774
\l_stex_key_for_clist 2821, 6754,
6762, 6772, 7059, 7068, 7340, 7345
\l_stex_key_for_str ..... 6842
\l_stex_key_from_tl .... 7060, 7069
\l_stex_key_Ftext_tl 8598, 8606, 8620
\l_stex_key_functions_tl . 7064, 7072
\l_stex_key_given_tl .....
    ..... 8698, 8702, 8735, 8736
\l_stex_key_hide_bool 7066, 7075, 7208
\l_stex_key_id_str .....
    ..... 121, 387, 389, 432,
433, 6823, 6824, 6891, 6893, 7286,
7295, 7384, 8360, 8361, 8363, 8369,
8417, 8418, 8420, 8426, 8466, 8467,
8469, 8475, 8515, 8516, 8518, 8524
\l_stex_key_intent_args_clist ...
    ..... 4106, 4112, 4274, 4283, 4285
\l_stex_key_intent_str .....
    ..... 3915, 4105, 4111, 4197, 4277
\l_stex_key_macroname_str .....
    .. 6753, 6763, 6780, 6782, 6791, 6795
\l_stex_key_method_tl .....
    .. 7065, 7074, 7195, 7196, 7342, 7346
\l_stex_key_mhrepos_str .....
    .. 8241, 8246, 8342, 8344, 8352, 8751
\l_stex_key_min_tl .....
    ..... 8239, 8244, 8298, 8316, 8336
\l_stex_key_name_str .....
    . 113, 114, 3673, 3680, 3704, 3705,
3719, 3720, 3733, 3742, 3751, 3792,
3847, 3876, 3880, 3889, 3890, 3892,
3900, 3908, 3918, 3926, 3932, 3933,
4433, 4456, 4457, 4475, 4477, 4477,
4728, 4729, 4739, 4746, 4749, 4764, 4766,
4779, 4792, 4841, 4844, 4845, 4849,
4851, 4852, 4921, 4922, 4944, 4959,
4961, 4974, 4989, 6752, 6761, 6781,
6782, 6786, 6789, 6795, 6803, 6812,
6841, 7341, 7348, 7364, 7365, 7378,
7379, 7380, 7609, 7610, 7619, 7631,
8240, 8245, 8271, 8273, 8276, 8294
\l_stex_key_number_tl .....
    ..... 8697, 8701, 8717, 8718
\l_stex_key_op_tl 3914, 4104, 4110,
4158, 4159, 4160, 4167, 4168, 4175,
4180, 4311, 4345, 4349, 4407, 4412,
4428, 4848, 4850, 4853, 4931, 4933
\l_stex_key_post_tl .. 5881, 5885,
5902, 5913, 5946, 5959, 6964, 6972
\l_stex_key_pre_tl ... 5880, 5884,
5902, 5913, 5946, 5959, 6964, 6972
\l_stex_key_prec_str .. 115, 3916,
4103, 4109, 4209, 4212, 4215, 4221,
4224, 4227, 4230, 4236, 4248, 4249
\l_stex_key_proofend_tl .....
    ..... 7061, 7070, 7172, 7173

```

```

\l_stex_key_proot_t1 ..... 5882
\l_stex_key_pts_t1 ..... 8238, 8243, 8297, 8313, 8336
\l_stex_key_reorder_str 3662, 3666,
3759, 3760, 4803, 4804, 5000, 5001
\l_stex_key_return_t1 .....
... 3677, 3682, 3738, 3773, 3776,
3869, 4740, 4771, 4784, 4818, 4821,
4929, 4966, 4979, 5011, 5014, 6758
\l_stex_key_role_str .....
.... 3661, 3669, 3762, 3763,
3894, 4800, 4801, 4997, 4998, 6792
\l_stex_key_root_t1 ..... 5886
\l_stex_key_short_t1 .....
.... 1307, 1310, 1337, 1339
\l_stex_key_sig_str .....
111, 2281, 2323, 2471, 2486, 2503, 8282
\l_stex_key_style_clist .....
.... 122, 399, 401,
451, 455, 502, 506, 6826, 6827, 6920
\l_stex_key_T_bool .....
.... 8596, 8602, 8604, 8617, 8637
\l_stex_key_term_t1 .....
.. 7063, 7073, 7192, 7193, 7343, 7347
\l_stex_key_title_str ..... 6764
\l_stex_key_title_t1 .....
393, 395, 1563, 1741, 1742, 2493, 6840,
6845, 8288, 8289, 8295, 8359, 8401,
8402, 8416, 8458, 8459, 8465, 8507,
8508, 8514, 8556, 8557, 8730, 8731
\l_stex_key_Ttext_t1 8597, 8605, 8618
\l_stex_key_type_t1 .....
114, 3675, 3685, 3706, 3737, 3767, 3768,
3865, 3877, 3881, 4458, 4770, 4783,
4812, 4813, 5005, 5006, 6756, 6765
\l_stex_key_variant_str .....
.... 115, 4102, 4108,
4114, 4142, 4195, 4308, 4317, 4349,
4443, 4462, 4751, 4842, 4850, 4946
\stex_keys_define:nnnn .....
.... 121, 359, 359, 378, 386, 392,
398, 404, 410, 1336, 1547, 1553,
2470, 3659, 3672, 3875, 4101, 4453,
4719, 5879, 6069, 6751, 7058, 7339,
7873, 8237, 8341, 8594, 8696, 8795
\stex_keys_set:nn .....
121, 122, 374, 374, 1343, 1565, 1766, 1771,
2492, 3694, 3886, 3887, 4118, 4431,
4472, 4726, 4919, 5891, 5899, 5910,
5932, 5943, 5956, 6080, 6833, 6862,
6956, 6962, 6970, 7148, 7219, 7274,
7377, 7573, 7607, 7953, 8264, 8268,
8351, 8615, 8711, 8715, 8750, 8808
\stex_kpsewhich:Nn 131, 36, 36, 47, 54
\c_stex_language_abbrevs_prop ...
.... 128, 103
\stex_language_from_file: .....
.... 129, 151, 151, 997
\c_stex_languages_clist . 23, 174, 177
\c_stex_languages_prop .....
.... 128, 103, 131, 132, 166, 182, 798
\g_stex_last_feature_str .....
.... 117, 2771, 6106
\stex_macro_body:N . 130, 537, 539, 557
\stex_macro_definition:N 130, 552, 552
\l_stex_macroname_str .....
.... 114, 215, 2761,
2762, 3692, 3696, 3698, 3732, 3753,
3754, 3888, 3899, 4473, 4727, 4765,
4777, 4794, 4795, 4920, 4960, 4972,
4991, 4992, 6091, 6791, 7608, 7618
\stex_main_archive: .... 1192, 1231
\c_stex_main_archive_prop . 123, 1192
\c_stex_main_file .....
.... 126, 127, 967, 1006, 1072, 7643
\stex_map_args:N .....
.... 3777, 4076, 4076, 4174, 4270,
4341, 4505, 4822, 5015, 5378, 6120
\stex_map_notation_args:N .....
.... 4076, 4089, 4374
\stex_map_uri:Nnnnn 127, 804, 811, 2338
\c_stex_mathhub_file 127, 911, 1035,
1111, 1121, 1124, 1134, 1193, 1611,
1690, 1701, 1853, 1865, 1877, 1880,
1904, 1909, 1920, 1972, 2022, 2026,
2036, 2040, 2050, 2055, 3096, 3111
\c_stex_mathhub_main_manifest_-
prop ..... 1199, 1200
\stex_mathml_arg:nn ..... 130
\stex_mathml_intent:nn ..... 130
\stex_maybe_brackets:nn .....
.. 4558, 4575, 5039, 5275, 5286, 5828
\stex_metagroup_do_in:nn .....
.... 112, 132, 225, 230,
237, 2532, 3344, 3378, 7615, 7616, 7627
\stex_metagroup_new:n .....
. 132, 219, 220, 224, 2283, 2332, 2816
\l_stex_metatheory_uri .....
.... 2152, 2358, 2361, 2445,
2461, 2475, 2477, 2504, 2505, 7534,
7535, 7536, 7583, 7584, 8283, 8284
\c_stex_module_ ..... 112, 114
\stex_module_add_code:n .....
.... 112, 2535, 2535, 2538, 2540, 2550, 7591
\stex_module_add_morphism:nnn .. 111
\stex_module_add_morphism:nnnn ..
.... 119, 2556, 2556,
2561, 2920, 3258, 6162, 6218, 7594

```

```

\stex_module_add_notation:nnnn . . . . . 114, 2599,
2599, 2610, 2912, 2915, 4306, 4425
\stex_module_add_symbol:nnnnnnN . . . . . 111, 2562
\stex_module_add_symbol:nnnnnnnN . . . . . 112, 2562, 2934, 2938, 2941,
3731, 3898, 6086, 6118, 6794, 7617
\stex_module_setup:n . . . . . 111, 2274,
2274, 2497, 2767, 7454, 7574, 8276
\_stex_module_setup_top_nosig:n . . . . .
2282, 2291, 2351, 7570
\l_stex_morphism_morphisms_seq . . . . .
118, 2849, 2859, 2954
\l_stex_morphism_renames_prop . . . . .
118, 2848, 2858,
2905, 2911, 2924, 2932, 2979, 3381
\l_stex_morphism_symbols_prop . . . . .
118, 2837,
2847, 2857, 2875, 2879, 2888, 2902,
2907, 2975, 2993, 2998, 3347, 3443
\stex_new_statement:nn . . . . . 6770
\stex_new_statement:nnn . . . . .
6831, 6897, 6903, 6918, 6919
\stex_new_styable_cmd:nnn . . . . .
122, 437, 437,
3030, 3243, 3292, 3318, 3362, 3388,
3566, 3586, 3607, 3693, 3885, 4117,
4430, 4471, 4725, 4918, 7146, 7540
\stex_new_styable_env:nnnnnn . . . . .
122, 437,
472, 2491, 3453, 3475, 3495, 6061,
6143, 6192, 6832, 7243, 7261, 7273,
8261, 8375, 8432, 8481, 8530, 8708
\_stex_next_symbol:n . . . . .
5135, 5136, 5154, 6469, 6591, 6951
\l_stex_notation_. . . . . 115
\_stex_notation_add: . . . . .
3920, 4122, 4294, 4305, 4481, 7634
\l_stex_notation_args_tl . . . . .
4090, 4164, 4198, 4269, 4275, 4281, 4479
\_stex_notation_check: . . . . .
4121, 4294,
4294, 4435, 4480, 4742, 4941, 7633
\_stex_notation_do_html:n . . . . .
3923, 4125, 4294, 4314, 4483, 4746, 7636
\_stex_notation_downprec . . . . .
5494, 5822, 5823, 5833, 5834, 5854, 5856
\l_stex_notation_macrocode_cs . . . . .
114–116, 4148, 4193,
4205, 4299, 4310, 4339, 4399, 4427,
4446, 4465, 4754, 4843, 4846, 4949
\_stex_notation_make_args: . . . . .
4149, 4294, 4300, 4373, 4447, 4466, 4755
\stex_notation_parse:n . . . . .
3919, 4120, 4157, 4157,
4434, 4478, 4741, 4932, 4935, 7632
\stex_notation_parse_and_then:nw . . . . .
115, 117
\stex_notation_set_default:n . . . . .
4128, 4393, 4414, 4424, 4438
\stex_notation_top:nw . . . . . 117
\stex_persist:n . . . . .
111, 130, 610, 614, 615, 617, 620,
623, 624, 630, 631, 1186, 1211, 2380
\c_stex_persist_mode_bool . . . . .
130, 25, 601, 664, 1230
\_stex_persist_read_now: . . . . .
663, 1210, 7645
\c_stex_persist_write_mode_bool . . .
130, 26, 607, 612, 665, 671, 1209, 2369
\stex_pseudogroup:nn . . . . . 131,
132, 199, 199, 280, 1090, 2637, 5367
\stex_pseudogroup_restore:N . . . . .
131, 132, 199, 202, 1103, 2641
\stex_pseudogroup_with:nn . . . . .
131, 209, 209, 812,
934, 2360, 2648, 2666, 2691, 3988,
4654, 4663, 4688, 5854, 5868, 5990
\c_stex_pwd_file . . . . .
127, 967, 987, 1193, 1194, 1708, 1860
\stex_reactivate_macro:N . . . . .
348, 2555, 2967, 2968, 2969, 3286,
3288, 3471, 3492, 3516, 3581, 3602,
3623, 3716, 3941, 4155, 4495, 6102,
6177, 6240, 6883, 6884, 6885, 6886,
6887, 6901, 6910, 6911, 6912, 6913,
6914, 6915, 6916, 7230, 7231, 7232,
7233, 7234, 7235, 7236, 7237, 7544,
7568, 7596, 8302, 8303, 8304, 8578
\stex_ref_new_doc_target:n . . . . .
110, 121, 433, 1496, 1509, 1533
\_stex_ref_new_id:n . . . . .
1499, 1510, 6892
\stex_ref_new_sym_target:n . . . . .
110, 1779, 1803, 1809, 6880, 6907, 6941
\stex_ref_new_sym_target:nn . . . . .
110, 1807
\stex_ref_new_symbol:n . . . . .
110, 1779, 2405, 3741, 3907
\l_stex_ref_url_str . . . . .
1506, 1527, 1529
\_stex_renamedecl_do:nn . . . . .
3364, 3369, 3563
\stex_require_archive:n . . . . .
119, 123, 1080, 1109, 1109,
1119, 1602, 1864, 3083, 3093, 3108
\stex_resolve_path_pair:Nnn . . . . .
123, 1849, 1849, 1871
\stex_return_args:nn . . . . .
3745, 3777, 4822, 5015

```

```

\l_stex_return_notation_tl . . .
    ..... 5175, 5352, 5353, 6307,
    6315, 6477, 6478, 6558, 6584, 6585
\stex_set_current_archive:n . .
    ... 123, 1079, 1079, 1099, 1205, 3227
\stex_set_current_namespace: .
    ..... 128, 963, 963, 998
\stex_set_document_uri: .
    ..... 128, 959, 959, 996
\stex_set_language:n . .
    ..... 129, 129, 129, 150, 167, 2484
\stex_set_notation_macro:nnnn . .
    114, 115, 2599, 2616, 2620, 2620, 2632
\stex_sms_allow:N . .
    124, 2095,
    2095, 2106, 2107, 2108, 2109, 2110
\stex_sms_allow_env:n . .
    125,
    2095, 2103, 2522, 3473, 3494, 3518,
    6100, 6174, 6236, 6857, 7605, 8324
\stex_sms_allow_escape:N . .
    125, 2095, 2099, 2469, 2554, 3285,
    3323, 3367, 3583, 3604, 3625, 3717,
    3942, 4156, 4496, 6873, 6997, 7641
\stex_sms_allow_import:Nn . .
    ..... 125, 2116, 2118, 3287
\stex_sms_allow_import_env:nn . .
    ..... 125, 2116, 2122
\g_stex_sms_import_code . .
    ..... 125, 2127, 2150, 2168, 3277
\stex_smemode_do: . .
    2141, 2143, 2177, 2182, 2467, 2515,
    2551, 3245, 3314, 3321, 3365, 3462,
    3482, 3506, 3577, 3598, 3619, 3713,
    3938, 4138, 4491, 6063, 6168, 6225,
    6850, 6871, 6994, 7597, 7638, 8305
\stex_split_slash: . .
    ..... 5922, 5926, 5975, 6937
\stex_sref_do_aux:n . .
    ..... 1785, 1792, 1796, 1799, 7450
\stex_str_if_ends_with:nn .. 125, 527
\stex_str_if_ends_with:nnTF . .
    ..... 125, 527, 980, 3397, 3420, 4045
\stex_str_if_ends_with_p:nn . .
    ..... 125, 527, 4020, 4060
\stex_str_if_starts_with:nn 125, 532
\stex_str_if_starts_with:nnTF . .
    ..... 125, 413, 532, 2874, 6285, 6289, 7000
\stex_str_if_starts_with_p:nn . .
    ..... 125, 532
\stex_structural_feature_-
    module:nn ... 117, 2756, 2756, 6096
\stex_structural_feature_module_-
    end: 117, 2756, 2770, 6065, 6170, 6232
\stex_structural_feature_-
    morphism:nnn . .... 2777
\stex_structural_feature_-
    morphism:nnnn . .... 119
\stex_structural_feature_-
    morphism:nnnnn . .... 118, 2778,
    3455, 3476, 3497, 3567, 3587, 3608
\stex_structural_feature_-
    morphism_check_total: . ....
    ..... 2887, 3484, 3508, 3596, 3617
\stex_structural_feature_-
    morphism_end: . 118, 2777, 2844,
    3467, 3488, 3512, 3576, 3597, 3618
\stex_style_apply: . .....
    ..... 122, 123, 437, 442, 477,
    482, 2513, 2518, 3045, 3270, 3311,
    3460, 3465, 3480, 3486, 3504, 3510,
    3574, 3594, 3615, 3710, 3935, 4135,
    4450, 4488, 4758, 4953, 6847, 6853,
    7158, 7228, 7248, 7265, 7285, 7303,
    7306, 7315, 8296, 8308, 8372, 8387,
    8391, 8429, 8444, 8448, 8478, 8493,
    8497, 8527, 8542, 8546, 8723, 8726
\stex_suppress_html:n . .....
    ..... 129, 279, 279, 2128, 4188
\_stex_symdecl_check_terms: . .....
    ... 114, 3723, 3862, 3862, 3897, 4733
\stex_symdecl_do: . .....
    ..... 113, 114, 3722, 3791,
    3791, 3896, 4732, 4927, 6793, 7612
\stex_symdecl_html: . .....
    ..... 3726, 3749, 3910, 6800
\stex_symdecl_top:n . .....
    ..... 114, 3700, 3718, 3718, 4474
\stex_symdef_styledefs: . 4455, 4487
\stex_term_arg:nnn . .....
    ..... 5485, 5489, 5496, 5506
\stex_term_arg:nnnn . .....
    ..... 4389, 4619, 4627,
    5489, 5490, 5521, 5555, 5592, 5677
\stex_term_arg_aB:nnnn . .....
    ..... 4632, 4640, 4647, 5511, 5519
\stex_term_do_aB_clist: . .
    ..... 4654,
    4655, 4664, 4668, 4689, 4694, 5516,
    5528, 5557, 5607, 5636, 5684, 5687
\stex_term_oma:nnn . .....
    ..... 4571, 5377, 5762, 5763, 5789
\stex_term_oma_or_omb:nnn . .....
    ..... 4571, 4576, 4624, 4637
\stex_term_omb:nnn . .....
    ..... 4624,
    4637, 5405, 5406, 5791, 5792, 5818
\stex_term_oms:nnn . .....
    ..... 4349, 5162, 5164, 5698,
    5708, 5757, 5761, 5893, 5903, 5914
\stex_term_oms_or_omv:nnn . .....
    ..... 3955, 4559, 5040, 5162,

```

```

    5164, 5196, 5198, 5276, 5287, 5360,
    5698, 5761, 6324, 6362, 6400, 6456
\stex_term_omv:nnn .....
    ..... 4911, 5196, 5198,
    5698, 5721, 5758, 5937, 5950, 5963
\stex_undefined: ..... 34
\stex_uri_add_module:NNn .....
    ..... 128, 932, 932, 957, 1511, 7534
\stex_uri_from_current_file:Nn ..
    ..... 128, 922, 922, 960
\stex_uri_from_current_file_-
    nolang:Nn ..... 128, 922, 928, 964
\stex_uri_from_repo_file ..... 128
\stex_uri_from_repo_file:NNNn ...
    ..... 127, 128, 885, 888, 924, 1616
\stex_uri_from_repo_file_-
    nolang:NNNn ..... 128, 885, 885, 929
\stex_uri_resolve:Nn ..... 127,
    870, 870, 873, 2461, 2477, 2481, 7453
\stex_uri_set:Nn 127, 820, 866, 869, 920
\stex_uri_use:N 127, 853, 874, 878,
    926, 961, 965, 1506, 1512, 1513,
    1517, 1583, 1585, 1618, 1625, 1642,
    1643, 2361, 2505, 3122, 3127, 3128,
    3141, 3146, 3147, 6824, 7584, 8284
\stex_vardecl_notation_macro: ...
    ... 114, 4436, 4743, 4839, 4839, 4942
\stex_variable:nnnnnnnN . 4718, 4859
\l_stex_variables_prop .....
    ..... 114, 4716, 4764, 4871, 4959
stex internal commands:
\stex_annotation_env_str ... 294, 295
\stex_aux_apply_patch:n .. 443, 450
\stex_aux_apply_patch_begin:n ..
    ..... 478, 501
\stex_aux_apply_patch_end:n ...
    ..... 483, 516
\stex_aux_args:n . 541, 545, 548, 551
\stex_aux_end: ..... 537, 538, 541
\stex_aux_params:n 555, 586, 593, 596
\stex_aux_patch:nnn ..... 439, 464
\stex_aux_patch:nnnn ..... 474, 491
\stex_aux_prefix:n .. 553, 561, 575
\stex_aux_prefix_long:n .....
    ..... 567, 571, 577, 584
\stex_aux_split_at_bracket:w ..
    ..... 414, 422
\stex_aux_start: ..... 537, 540
\l_stex_aux_tl .....
    ..... 364, 365, 366, 368, 371, 372
\stex_debug_nn ..... 60, 63, 68
\l_stex_debug_cl ..... 83, 85, 88
\stex_debug_env_str .... 81, 82, 85
\__stex_doc_check_topsect: .....
    ..... 1421, 1427, 1433, 1439
\__stex_doc_do_section:n .....
    ..... 1319, 1323, 1327, 1344
\__stex_doc_maketitle: ... 1262, 1267
\__stex_doc_orig_backmatter .....
    ..... 1454, 1457, 1475
\__stex_doc_orig_frontmatter ...
    ..... 1444, 1447, 1464
\__stex_doc_set_title:n .. 1239, 1255
\__stex_doc_skip_fragment:n .....
    ..... 1384, 1390,
    1394, 1398, 1399, 1400, 1401, 1402
\__stex_doc_skip_section: .....
    ..... 1352, 1374, 1380
\__stex_doc_skip_section_i: .....
    ..... 1361, 1364, 1367, 1375, 1380
\__stex_doc_title_html: .....
    ..... 1244, 1247, 1259
\g_stex_doc_title_tl .....
    .. 1234, 1236, 1243, 1249, 1254, 1257
\__stex_expr_aB_arg:nnnnn 5526, 5533
\__stex_expr_aB_simple_arg:nnnn
    ..... 5544, 5553
\__stex_expr_add_prop_arg:nnw ...
    ..... 5373, 5397, 5400
\__stex_expr_arg:n ..... 5374, 5410
\l__stex_expr_arg_counter_int ...
    ..... 5364, 5371, 5386, 5422, 5423
\__stex_expr_arg_do:nnn .....
    ..... 5424, 5430, 5446, 5481, 5488
\__stex_expr_arg_inner:nn .....
    ..... 5413, 5416, 5420, 5426
\__stex_expr_assoc_make_seq:nnn .
    ..... 5588, 5617, 5696
\__stex_expr_assoc_seq:nnnnnnn ..
    ..... 5537, 5585
\__stex_expr_check:n ..... 5454
\__stex_expr_check:nTF .. 5423, 5429
\__stex_expr_check_b:nn .. 5378, 5403
\l__stex_expr_count_int .....
    .. 5518, 5524, 5534, 5555, 5592, 5677
\l__stex_expr_cs .... 5619, 5622, 5642
\l__stex_expr_customs_prop .....
    ..... 5369, 5381, 5382,
    5383, 5398, 5434, 5440, 5455, 5458
\l__stex_expr_customs_seq .....
    .. 5370, 5387, 5388, 5389, 5399, 5456
\__stex_expr_do_aB_clist: .....
    ..... 5511, 5516, 5557, 5636
\__stex_expr_do_ab_next:nnn .....
    ..... 5377, 5379, 5405, 5406
\__stex_expr_do_headerterm:nn .....
    ..... 5686, 5718, 5731, 5734, 5759

```

```

\__stex_expr_do_ref:nNn ... 5893,
  5901, 5912, 5934, 5945, 5958, 5968
\__stex_expr_do_seqmap:nnnnnn ...
  ..... 5542, 5649
\__stex_expr_end: ..... 5539, 5549
\__stex_expr_gobble:nnnnnnnn ...
  ..... 5539, 5549
\l__stex_expr_iarg_tl 5629, 5631, 5642
\__stex_expr_invoke_custom:n ...
  ..... 5224, 5238, 5365
\__stex_expr_invoke_math: 5219, 5227
\__stex_expr_invoke_op_custom:n .
  ..... 5224, 5231, 5357
\__stex_expr_invoke_op_notation:w
  ..... 5233, 5234, 5272
\__stex_expr_invoke_return: ...
  ..... 5313, 5316
\__stex_expr_invoke_return_-
  maybe:n ..... 5255, 5265, 5294
\__stex_expr_invoke_return_next:
  ..... 5301, 5310
\__stex_expr_invoke_text: 5219, 5222
\__stex_expr_is_seqmap:n ...
  ..... 5578
\__stex_expr_is_seqmap:nTF ...
  ..... 5541
\__stex_expr_is_varseq:n ...
  ..... 5567
\__stex_expr_is_varseq:nTF 5535, 5654
\l__stex_expr_left_bracket_str ..
  ..... 5824, 5850, 5858, 5868, 5869
\l__stex_expr_old_seq ...
  ..... 5661, 5664, 5668, 5673
\l__stex_expr_reset_tl ...
  ..... 5135, 5138, 5142, 5143, 5149
\__stex_expr_ret_cs ...
  ..... 5298, 5303, 5330, 5335, 5340
\__stex_expr_return_arg:n 5300, 5306
\l__stex_expr_return_args_tl ...
  .. 5295, 5307, 5312, 5321, 5337, 5342
\__stex_expr_return_notation:n ...
  ..... 5318, 5351
\l__stex_expr_return_this_tl ...
  ..... 5296, 5312,
  5317, 5321, 5325, 5326, 5336, 5344
\l__stex_expr_right_bracket_str ..
  ..... 5825, 5851, 5863, 5868, 5870
\__stex_expr_setup:nnnnnn ...
  ..... 5161, 5173, 5195
\__stex_expr_varseq_in_map:nnnnnnnn
  ..... 5656, 5695
\__stex_features_add_definiens:nn
  ..... 2828, 2971
\__stex_features_break: ... 2829, 2833
\__stex_features_check_break:nnnnnnnn
  ..... 2992, 2997, 3006, 3009, 3016
\__stex_features_clean:nnw 2868, 2876
\__stex_features_do_decls: 2860, 2872
\__stex_features_do_elaboration:
  ..... 2850, 2900
\__stex_features_do_for_list: ...
  ..... 2819, 2970
\__stex_features_do_morph:nnmn ...
  ..... 2948, 2952
\__stex_features_do_morphisms:n .
  ..... 2861, 2946, 2956
\__stex_features_elab_check: ...
  ..... 2908, 2931
\l__stex_features_feature_str ...
  ..... 2811, 2923
\__stex_features_get_check:nnnn ...
  ..... 2976, 3004
\l__stex_features_implicit_bool .
  ..... 2777, 2796, 2799, 2936
\__stex_features_reactivate: ...
  ..... 2814, 2959
\__stex_features_rename:nn 2924, 2927
\__stex_features_rename_all: . 2864
\__stex_features_renamed_-
  check:nnnn ..... 2980, 2990
\__stex_features_set_definiens_-
  macros: ..... 2829, 2833
\__stex_features_set_definiens_-
  macros_i:nnnnnnn ..... 2836, 2840
\__stex_features_setup: ... 2813, 2856
\__stex_features_split_qm:w ...
  ..... 2899, 2917
\l__stex_features_tmp ...
  ..... 2911, 2913, 2932, 2933, 2934
\__stex_features_total_check: ...
  ..... 2889, 2893
\__stex_groups_do: ... 250, 257, 266
\__stex_groups_do_in:nn 232, 239, 260
\__stex_groups_exists:n ...
  ..... 225
\__stex_groups_exists:nTF ...
  ..... 231
\l__stex_groups_ids_seq 219, 222, 258
\__stex_groups_tmp ... 241, 248, 254
\l__stex_importmodule_archive_-
  str 3089, 3094, 3104, 3109, 3226, 3227
\__stex_importmodule_check_-
  file:nn ..... 3170,
  3171, 3172, 3173, 3174, 3175, 3197,
  3198, 3199, 3200, 3201, 3202, 3236
\__stex_importmodule_get_from_-
  file:nnn ..... 3130, 3166
\__stex_importmodule_get_from_-
  file_safe:nnn ..... 3149, 3193
\__stex_importmodule_get_-
  module:nnn ..... 3091, 3097, 3155
\__stex_importmodule_get_module_-
  safe:nnn ..... 3106, 3112, 3161

```

```

\__stex_importmodule_get_module_-
    uri:nnn ..... 3116, 3157
\__stex_importmodule_get_module_-
    uri_safe:nnn ..... 3135, 3163
\__stex_importmodule_import_-
    module:nn ..... 3244, 3249, 3289
\__stex_importmodule_import_-
    module_presms:nn ..... 3275, 3289
\__stex_importmodule_load_file:n
    ..... 3187, 3190, 3214, 3219, 3224
\l__stex_importmodule_path_seq ..
    ..... 3066, 3067, 3069
\__stex_importmodule_seq .....
    ..... 3051, 3052, 3053
\l__stex_importmodule_seq .....
    .. 3073, 3075, 3167, 3168, 3194, 3195
\l__stex_importmodule_str .....
    ..... 3096, 3097, 3111,
    3112, 3168, 3184, 3195, 3210, 3213,
    3218, 3225, 3229, 3237, 3238, 3240
\__stex_inputs_bibresource:n ...
    ..... 1937, 1949, 1951
\l__stex_inputs_gin_repo_str ...
    ..... 2019, 2049, 2053, 2058
\l__stex_inputs_id_seq .....
    ..... 1973, 1975, 1980
\l__stex_inputs_id_str ... 1968, 1973
\__stex_inputs_inputref:nn 1925, 1935
\__stex_inputs_inputref_html:nn .
    ..... 1890, 1928
\__stex_inputs_inputref_pdf:nn ...
    ..... 1914, 1929
\__stex_inputs_libinput:n .....
    ..... 1990, 2003, 2005
\l__stex_inputs_libinput_files_-
    seq 1939, 1942, 1971, 1978, 1986,
    1992, 1995, 2010, 2011, 2085, 2086
\__stex_inputs_mhinput:nn 1874, 1887
\l__stex_inputs_path_seq .....
    ..... 1972, 1976, 1981, 1984
\l__stex_inputs_path_str .....
    ..... 1976, 1977,
    1978, 1980, 1981, 1984, 1985, 1986
\l__stex_inputs_tmp_str .. 2011, 2013
\__stex_inputs_up_archive:nn ...
    ..... 1938, 1964, 1991, 2009, 2084
\__stex_inputs_usetikzlibrary:nn
    ..... 2064, 2086
\l__stex_iterate_continue_bool ..
    ..... 2716, 2719, 2732, 2752
\__stex_iterate_it_decl_check:nnnn
    ..... 2677, 2684
\__stex_iterate_it_decl_i:n ...
    ..... 2669, 2673, 2686
\__stex_iterate_it_not_check:nnnn
    ..... 2705, 2709
\__stex_iterate_it_not_i:n .....
    ..... 2694, 2698, 2711
\__stex_iterate_iterate_morphism:nn
    ..... 2722, 2726, 2735, 2738
\l__stex_iterate_mods_seq .....
    ..... 2665, 2674, 2675,
    2690, 2699, 2700, 2715, 2740, 2741
\__stex_iterate_morphism_cs:nnnn
    ..... 2717, 2743
\__stex_iterate_not_cs:nnnnn ...
    ..... 2691, 2692, 2702
\__stex_iterate_sym_cs:nnnnnnnn
    .. 2648, 2649, 2659, 2666, 2667, 2680
\l__stex_iterate_todo_tl .....
    ..... 2721, 2725, 2739, 2752
\l__stex_lang_lang_str . 132, 135, 142
\l__stex_lang_str .. 164, 165, 166, 167
\l__stex_lang_turkish_bool .....
    ..... 175, 180, 190
\l__stex_mathhub_bool .... 1132,
    1133, 1135, 1138, 1148, 1150, 1159
\__stex_mathhub_check_manifest: .
    ..... 1137, 1146
\__stex_mathhub_check_manifest:n
    ..... 1147, 1149, 1151, 1156
\l__stex_mathhub_cs .. 1087, 1088,
    1091, 1094, 1096, 1100, 1104, 1105
\__stex_mathhub_do_manifest:n ...
    ..... 1115, 1120
\__stex_mathhub_find_manifest:n .
    ..... 1121, 1129, 1144, 1194
\l__stex_mathhub_key 1171, 1172, 1174
\c__stex_mathhub_manifest_ior ...
    ..... 1164, 1166, 1168, 1183
\l__stex_mathhub_manifest_str ...
    ..... 1122, 1130, 1160, 1166, 1195
\__stex_mathhub_parse_manifest:n
    ..... 1126, 1165, 1198
\l__stex_mathhub_prop .....
    ..... 1167, 1175, 1176,
    1177, 1178, 1179, 1184, 1185, 1188
\l__stex_mathhub_seq .....
    ..... 1131, 1134, 1139,
    1157, 1158, 1160, 1170, 1171, 1173
\l__stex_mathhub_str .....
    1037, 1039, 1041, 1044, 1045, 1049,
    1050, 1055, 1061, 1064, 1069, 1072,
    1202, 1203, 1205, 1212, 1216, 1219
\l__stex_mathhub_tl .....
    .. 1173, 1175, 1176, 1177, 1178, 1179
\__stex_module_setup_end: 2329, 2350

```

```

\__stex_module_setup_get_uri_-
    str:n ..... 2279, 2336
\__stex_module_setup_load_meta: .
    ..... 2288, 2357
\__stex_module_setup_load_sig: ..
    ..... 2309, 2319
\l__stex_module_setup_ns_str ...
    ..... 2280, 2282, 2337, 2339, 2346
\l__stex_module_setup_seg .....
    ..... 2342, 2343, 2344
\l__stex_module_setup_seq .....
    ..... 2341, 2342, 2344, 2346
\__stex_module_setup_setup_-
    nested:n ..... 2275, 2327
\__stex_module_setup_setup_top:n
    ..... 2275, 2278
\__stex_module_setup_setup_top_-
    sig:n ..... 2282, 2304
\l__stex_module_setup_sigfile ...
    ..... 2320, 2321, 2323
\__stex_module_setup_split_-
    module:n ..... 2329, 2350
\__stex_modules_activate_-
    i:nnnnnnnn ..... 2583, 2587
\__stex_modules_activate_not:nn .
    ..... 2607, 2613
\__stex_modules_activate_sym:n ..
    ..... 2575, 2580
\__stex_modules_export:n . 2546, 2548
\__stex_modules_persist_module: .
    ..... 2369, 2379, 7448
\__stex_modules_persist_nots_-
    i:nn ..... 2395, 2416
\l__stex_modules_restore_mod_str
    ..... 2409, 2437
\__stex_modules_restore_module:nnnn
    ..... 2381, 2400
\__stex_modules_restore_nots:n ..
    ..... 2413, 2422
\__stex_modules_restore_nots_i:n
    ..... 2423, 2426, 2443
\__stex_modules_restore_nots_-
    ii:nnnn ..... 2430, 2434
\__stex_modules_set_metatheory:nn
    ..... 2446, 2466
\__stex_modules_tl .....
    ..... 2402, 2403
\l__stex_modules_tl .....
    ..... 2435, 2440
\l__stex_morphisms_ass_tl .....
    ..... 3522, 3535, 3539, 3549, 3550
\__stex_morphisms_do_morph_-
    assign:nnn .....
    ..... 3395, 3398, 3436
\__stex_morphisms_do_parsed_-
    assign: ..... 3541, 3544
\__stex_morphisms_do_parsed_-
    newname: ..... 3547, 3554
\__stex_morphisms_do_parsed_-
    newname:w ..... 3556, 3558, 3562
\__stex_morphisms_end: ... 3547, 3562
\l__stex_morphisms_morphism_dom_-
    str ... 3389, 3402, 3414, 3424, 3438
\l__stex_morphisms_name_str 3520,
    3527, 3528, 3532, 3533, 3534, 3545
\l__stex_morphisms_newname_str ...
    ..... 3521, 3537, 3538, 3546, 3547
\l__stex_morphisms_next_tl .....
    ..... 3525, 3529, 3531
\__stex_morphisms_parse_assign:n
    ..... 3519, 3569, 3589, 3610
\l__stex_morphisms_seq .....
    ..... 3523, 3524, 3525, 3527,
    3529, 3531, 3533, 3535, 3537, 3539
\__stex_notations_add:nnnnn .....
    ..... 4605, 4610, 4614
\__stex_notations_add_last:nnnnn
    ..... 4598, 4609
\__stex_notations_add_missing_-
    args:nn ..... 4174, 4289
\__stex_notations_add_next:nnnnnn
    ..... 4600, 4604
\l__stex_notations_after_tl .....
    ..... 4584, 4611
\__stex_notations_args_end: .....
    ..... 4079, 4082, 4085,
    4092, 4095, 4098, 4593, 4596, 4606
\l__stex_notations_args_tl .....
    ..... 4500, 4504, 4517
\__stex_notations_augment_arg:nn
    ..... 4505, 4522
\__stex_notations_check_aB_-
    arg:Nn ..... 4645, 4653, 4662, 4687
\l__stex_notations_clist_count_-
    int ..... 4685, 4693, 4699
\l__stex_notations_code_tl .....
    ... 4541, 4556, 4564, 4572, 4586,
    4587, 4590, 4618, 4626, 4631, 4639
\__stex_notations_complex:nnnnnn
    ..... 4550, 4568
\__stex_notations_const_precs: ...
    ..... 4166, 4208
\l__stex_notations_cs .....
    ..... 4557, 4562, 4573, 4582
\__stex_notations_do_argname:nn .
    ..... 4270, 4273
\__stex_notations_do_argnames: ...
    ..... 4244, 4268
\__stex_notations_fun_precs: ...
    ..... 4171, 4220

```

```

\__stex_notations_make_arg:nnnn . . . . . 4374, 4378
\__stex_notations_make_arg_- html:nn . . . . . 4341, 4356
\__stex_notations_make_name: . . . . . 4502, 4526, 4534
\__stex_notations_map_args_i:w . . . . . 4078, 4082, 4085
\__stex_notations_map_args_i:w . . . . . 4091, 4095, 4098
\__stex_notations_map_cs: . . . . . 4327, 4329,
                                4665, 4667, 4675, 4690, 4692, 4703
\l__stex_notations_missing_str . . . . . 4172, 4290
\l__stex_notations_missing_tl . . . . . 4173, 4202, 4291
\l__stex_notations_name_str . . . . . 4528, 4529, 4530, 4535
\l__stex_notations_opprec_tl 4196,
                            4210, 4213, 4215, 4222, 4225, 4227,
                            4231, 4238, 4251, 4257, 4263, 4318
\__stex_notations_parse_notation_- args:nnnnw . . . . . 4593, 4596, 4606
\__stex_notations_parse_precs: . . . . . 4242, 4247
\l__stex_notations_pre_tl . . . . . 4571, 4586, 4623, 4636
\l__stex_notations_precs_seq . . . . . 4163, 4233,
                                4240, 4261, 4263, 4276, 4282, 4319
\__stex_notations_process_- notation:nnnnnn . . . . . 4540, 4546
\l__stex_notations_seq . . . . . 4249, 4250, 4252, 4253, 4260
\__stex_notations_simple:nnnnn . . . . . 4548, 4554
\l__stex_notations_str . . . . . 4250, 4251, 4252, 4254, 4260, 4261
\__stex_notations_styledefs: . . . . . 4134, 4141
\__stex_path_: . . . . . 682, 691
\l__stex_path_a_seq . . . . . 762, 768, 774
\l__stex_path_a_tl . . . . . 774, 776
\__stex_path_auth:n . . . . . 127, 804, 807, 808,
                         809, 812, 813, 858, 874, 879, 933, 935
\l__stex_path_auth_str . . . . . 826, 834, 839, 844
\l__stex_path_b_seq . . . . . 763, 769, 775, 781
\l__stex_path_b_tl . . . . . 775, 776
\__stex_path_canonicalize:N . . . . . 694, 705, 711
\__stex_path_colonslash . . . . . 820, 825, 830
\l__stex_path_do_hooks_pre_t1 . . . . . 1015, 1016, 1028, 1031
\__stex_path_dodots:n . . . . . 716, 720, 726
\l__stex_path_file . . . . . 893, 899, 905, 910
\__stex_path_from_repo_file:NNNNn . . . . . 886, 889, 892
\l__stex_path_maybein_str . . . . . 743, 744, 745
\l__stex_path_mod . . . . . 837, 840, 845
\__stex_path_module:n . . . . . 127,
                           808, 812, 815, 860, 874, 935, 937, 939
\l__stex_path_name . . . . . 842, 845
\__stex_path_name:n . . . . . 127,
                           809, 812, 816, 861, 874, 935, 944, 946
\l__stex_path_path . . . . . 831, 832, 835, 840, 845, 914, 915
\__stex_path_path:n . . . . . 127, 807, 812, 814, 859, 874, 935
\__stex_path_relativize:N . . . . . 902, 909
\l__stex_path_return_tl . . . . . 764, 770, 777, 780, 782
\l__stex_path_seq 714, 717, 722, 730,
                  731, 734, 787, 788, 789, 794, 795,
                  797, 799, 802, 910, 912, 916, 918, 920
\g__stex_path_stack . . . . . 977, 991, 1002, 1003, 1005, 1008
\l__stex_path_str . . . . . 684, 687, 689, 690, 691, 693, 696,
                           703, 704, 713, 715, 719, 731, 786,
                           787, 788, 793, 794, 795, 797, 798,
                           799, 968, 970, 972, 1003, 1008, 1009
\l__stex_path_tl . . . . . 912, 916, 918
\l__stex_path_uri . . . . . 895, 896, 897, 902, 920
\__stex_path_uri_set:NnN 821, 867, 871
\__stex_path_uri_set:Nnnnn . . . . . 834, 839, 844, 856, 864, 898, 904
\__stex_path_uri_use:w . . . . . 874, 880
\l__stex_path_win_drive . . . . . 683, 688, 695, 697
\__stex_path_win_take:w . . . . . 682, 691
\__stex_persist_env_str . . . . . 598, 599, 600, 604, 605, 606
\__stex_persist_load_file:n . . . . . 627, 659, 668
\__stex_persist_read_and_write: . . . . . 644, 666
\c__stex_persist_sms_iow . . . . . 610, 618, 640, 641, 656
\__stex_persist_write_only: . . . . . 639, 653, 660, 671
\__stex_proof_add_counter: 7121, 7302
\__stex_proof_begin_proof:nn . . . . . 7216, 7244, 7262

```

\l\_\_stex\_proof\_counter\_intarray . . . . .  
..... 7079, 7084,  
7087, 7096, 7099, 7108, 7116, 7117,  
7125, 7130, 7137, 7143, 7217, 7218  
\\_stex\_proof\_end\_list: . . . . .  
.. 7186, 7257, 7318, 7328, 7394, 7417  
\\_stex\_proof\_html: . 7190, 7212, 7368  
\\_stex\_proof\_html\_env:n . . . . .  
..... 7201, 7222, 7277  
\l\_\_stex\_proof\_in\_spfblock\_bool . . . . .  
... 7215, 7245, 7256, 7263, 7291,  
7294, 7313, 7316, 7318, 7323, 7326,  
7334, 7376, 7390, 7412, 7432, 7438  
\\_stex\_proof\_inblock\_restore: .. . . . .  
..... 7320, 7333, 7434  
\\_stex\_proof\_inc\_counter: . . . . .  
..... 7104, 7316, 7407, 7409  
\l\_\_stex\_proof\_inc\_counter\_bool 7078  
\\_stex\_proof\_insert\_number: ... . . . . .  
..... 7080, 7300, 7407, 7409  
\\_stex\_proof\_make\_step\_macro:Nnnnn  
..... 7374, 7407, 7408, 7409  
\\_stex\_proof\_number\_as\_string:N  
..... 7091, 7287, 7296, 7385  
\\_stex\_proof\_proof\_box\_tl 7061, 7165  
\\_stex\_proof\_remove\_counter: ... . . . . .  
..... 7133, 7327  
\\_stex\_proof\_start\_list:n . . . . .  
..... 7179, 7245, 7300, 7394, 7417  
\\_stex\_proof\_step\_html:nn 7356,  
7392, 7394, 7400, 7415, 7417, 7421  
\\_stex.refs.add\_doc\_ref:mn . . . . .  
..... 1513, 1535, 1546  
\l\_\_stex.refs.default\_archive\_-  
str . . . . . 1555, 1561, 1570  
\l\_\_stex.refs.default\_file\_str .. . . . .  
..... 1556, 1562, 1571  
\\_stex.refs.do\_autoref:n . . . . .  
..... 1631, 1643, 1654,  
1658, 1669, 1683, 1709, 1720, 1728  
\\_stex.refs.do\_internal\_link:nn  
..... 1820, 1832, 1838  
\\_stex.refs.do\_return:nnnn . . . . .  
..... 1737, 1751, 1759  
\\_stex.refs.do\_sref:nn .. 1639, 1768  
\\_stex.refs.do\_sref\_in:n . . . . .  
..... 1647, 1662, 1673, 1679, 1777  
\\_stex.refs.do\_url\_link:nn . . . . .  
..... 1826, 1840  
\l\_\_stex.refs.file . . . . .  
.. 1597, 1599, 1615, 1617, 1705, 1706

\l\_\_stex.refs\_file\_str . . . . .  
1689, 1695, 1700, 1705, 1706, 1707,  
1708, 1713, 1717, 1719, 1727, 1739  
\g\_\_stex.refs\_files\_seq . . . . .  
..... 1496, 1536, 1541, 1588, 1650  
\\_stex.refs\_find\_uri:n . . . . .  
..... 1575, 1767, 1772  
\\_stex.refs\_find\_uri\_in\_-  
file:nnn . . . . . 1584, 1589, 1621  
\\_stex.refs\_find\_uri\_in\_prop\_-  
file:N . . . . . 1595, 1604, 1609  
\l\_\_stex.refs\_id\_str . . . . .  
..... 1715, 1750, 1756, 1757  
\c\_\_stex.refs\_iow . . . . .  
..... 1487, 1488, 1489, 1515  
\\_stex.refs\_new\_symbol:n 1781, 1789  
\l\_\_stex.refs\_prop . . . . . 1603, 1604  
\\_stex.refs\_restore\_target:nnnn  
..... 1716, 1748  
\\_stex.refs\_return\_tl . . . . .  
..... 1714, 1718, 1724, 1738  
\\_stex.refs\_set\_keys\_b:n . . . . .  
..... 1559, 1646, 1661, 1672, 1773  
\l\_\_stex.refs\_str 1502, 1504, 1506,  
1511, 1513, 1518, 1791, 1793, 6893  
\\_stex.refs\_sym\_aux:nn . . . . .  
..... 1817, 1830, 1835, 1846  
\\_stex.refs\_unnamed\_counter\_-  
int . . . . . 1497, 1501, 1502  
\l\_\_stex.refs\_uri . . . . .  
..... 1511, 1512, 1616, 1618  
\l\_\_stex.refs\_uri\_str . . . . .  
... 1576, 1587, 1599, 1610, 1615,  
1618, 1625, 1640, 1650, 1651, 1652,  
1653, 1654, 1657, 1658, 1660, 1666,  
1668, 1669, 1671, 1681, 1682, 1683,  
1710, 1721, 1729, 1749, 1754, 1755  
\\_stex\_seqs\_add: . . . . . 4939, 4957  
\\_stex\_seqs\_args\_tl ... 5015, 5017  
\\_stex\_seqs\_check\_terms: 4938, 4985  
\l\_\_stex\_seqs\_clist . . . . .  
..... 5052, 5059, 5066, 5070  
\l\_\_stex\_seqs\_cs . . . . .  
..... 5013, 5017, 5073, 5077,  
5085, 5088, 5097, 5104, 5117, 5118  
\\_stex\_seqs\_do\_all:w ... 5111, 5121  
\\_stex\_seqs\_do\_first: ... 5034, 5095  
\\_stex\_seqs\_do\_first\_arg:n . . . . .  
..... 5093, 5100  
\\_stex\_seqs\_do\_first\_next: . . . . .  
..... 5102, 5107  
\\_stex\_seqs\_do\_one:w ... 5109, 5115  
\\_stex\_seqs\_do\_op:w ... 5033, 5037  
\\_stex\_seqs\_doop\_arg:n .. 5053, 5064

```

\__stex_seqs_doop_range:w  5046, 5050
\__stex_seqs_first_args_tl . . . . .
. . . . . 5099, 5117, 5118, 5122, 5123
\__stex_seqs_get_index_notation:n
. . . . . 5045, 5083, 5116
\__stex_seqs_html: . . . . . 4937, 4987
\__stex_seqs_macro: . . . . . 4940, 4971
\__stex_seqs_make_args: . . . 4950, 4984
\__stex_seqs_range_clist . . . .
. . . . . 4930, 4965, 4978
\g__stex_smsemode_allowed_escape_tl . . .
. . . . . 2093, 2100, 2229
\g__stex_smsemode_allowed_import_env_seq 2117, 2123, 2163, 2212, 2215
\g__stex_smsemode_allowed_import_tl . . .
. . . . . 2116, 2119, 2160, 2207
\g__stex_smsemode_allowed_tl . . .
. . . . . 2092, 2096, 2225
\g__stex_smsemode_allowedenvs_seq . . .
. . . . . 2094, 2104, 2234, 2237
\g__stex_smsemode_bool . . .
. . . . . 2111, 2112, 2114, 2130
\__stex_smsemode_check_begin:Nn . . .
. . . . . 2212, 2234, 2246
\__stex_smsemode_check_cs:NNn . . .
. . . . . 2185, 2193
\__stex_smsemode_check_end:Nn . . .
. . . . . 2215, 2237, 2255
\__stex_smsemode_do:w . . .
. . . . . 2179, 2191, 2193, 2195,
2217, 2227, 2239, 2252, 2259, 2262
\__stex_smsemode_do_aux:N . 2193, 2199
\__stex_smsemode_do_aux_curr:N . .
. . . . . 2159, 2170, 2201
\__stex_smsemode_do_aux_imports:N . .
. . . . . 2159, 2205
\__stex_smsemode_do_aux_normal:N . .
. . . . . 2170, 2223
\__stex_smsemode_importmodules_seq . . .
. . . . . 2148
\__stex_smsemode_in_smsemode:n . .
. . . . . 2128, 2158, 2168, 2169
\__stex_smsemode_sigmodules_seq 2149
\__stex_smsemode_smsemode_do: . .
. . . . . 2141, 2177
\__stex_smsemode_start_smsemode:n . .
. . . . . 2139, 2166, 2171
\__stex_statements_do_defref:nn . .
. . . . . 6928, 6957, 6963, 6971
\__stex_statements_force_id: . .
. . . . . 6787, 6890, 6898, 6921
\__stex_statements_html_keyvals:nn . .
. . . . . 6817, 6838, 6866
\__stex_statements_setup:nn . .
. . . . . 6779, 6899, 6904, 6918, 6922, 6925
\__stex_statements_setup_def: . .
. . . . . 6877, 6900, 6923
\__stex_statements_uri_str . .
. . . . . 6802, 6806, 6807, 6812, 6813, 6983,
6986, 6989, 6991, 7036, 7039, 7042
\__stex_structures_assigned_seq . .
. . . . . 6465, 6576, 6615, 6631
\__stex_structures_begin:nn . .
. . . . . 6062, 6079, 6156, 6211
\__stex_structures_check_def:nn . .
. . . . . 6230, 6243
\__stex_structures_clist . .
. . . . . 6439, 6460, 6466, 6468
\__stex_structures_comp_cs . .
. . . . . 6508, 6513
\__stex_structures_cs . .
. . . . . 6441, 6447, 6454
\__stex_structures_current_type: . .
. . . . . 6395, 6526, 6597
\__stex_structures_current_type_tl . .
. . . . . 6322, 6358, 6398, 6403
\__stex_structures_do_assign:nn . .
. . . . . 6342, 6346
\__stex_structures_do_assign_list:n . .
. . . . . 6318, 6339
\__stex_structures_do_decl:nnnnnnnn . .
. . . . . 6653, 6673
\__stex_structures_do_decl_nomacro:nnnnnnnn . .
. . . . . 6651, 6659
\__stex_structures_do_externals: . .
. . . . . 6066, 6104, 6171, 6233
\__stex_structures_end: . .
. . . . . 6286, 6290, 6306, 6311
\__stex_structures_exstruct_name_str . .
. . . . . 6206, 6211, 6255
\__stex_structures_extend_structure_nn . .
. . . . . 6183, 6206
\__stex_structures_extend_structure_i:Nnnnnnnn . .
. . . . . 6180, 6188
\__stex_structures_external_decl:nnnn . .
. . . . . 6107, 6111
\__stex_structures_extmod_str . .
. . . . . 6181, 6185
\__stex_structures_extname_count . .
. . . . . 6249, 6253, 6255
\__stex_structures_field_name_str . .
. . . . . 6563, 6567, 6568, 6599
\__stex_structures_fields_clist . .
. . . . . 6319,
6340, 6347, 6365, 6368, 6622, 6623
\__stex_structures_get_field_name:n . .
. . . . . 6562, 6574

```

```

\l__stex_structures_imports_seq .
    ..... 6144, 6149, 6157,
    6194, 6199, 6213, 6735, 6738, 6741
\l__stex_structures_invocation_-
    type:n ..... 6270, 6317
\l__stex_structures_invoke_-
    field:n ..... 6540, 6572
\l__stex_structures_invoke_maybe_-
    field:nn ..... 6529, 6533
\l__stex_structures_invoke_this:n
    ..... 6279, 6521
\l__stex_structures_invoke_top:n .
    ..... 6260,
    6263, 6287, 6291, 6294, 6297, 6299
\l__stex_structures_make_mod:n ...
    ..... 6364, 6376
\l__stex_structures_make_oml:n ...
    ..... 6368, 6382
\l__stex_structures_make_oml:nn ...
    ..... 6383, 6385
\l__stex_structures_make_prop: ...
    ..... 6407, 6534, 6612
\l__stex_structures_make_prop_-
    assign: ..... 6408, 6537, 6621
\l__stex_structures_make_prop_-
    assign:nn ..... 6624, 6629
\l__stex_structures_make_prop_-
    assign_replace:nnnn ... 6632, 6638
\l__stex_structures_make_type:n ..
    ..... 6327, 6332, 6334, 6350
\l__stex_structures_maybe_-
    notation:w ..... 6274, 6276, 6402
\l__stex_structures_merge:nw ...
    ..... 6268, 6284
\l__stex_structures_more_-
    nextsymbol_tl ... 6575, 6577, 6602
\l__stex_structures_name_str ...
    6054, 6071, 6075, 6081, 6082, 6087,
    6088, 6093, 6094, 6097, 6113, 6119
\l__stex_structures_new_extstruct_-
    name: ..... 6193, 6251
\l__stex_structures_present: ...
    ..... 6413, 6538
\l__stex_structures_present:nn ...
    ..... 6417, 6423, 6428, 6435, 6438
\l__stex_structures_present_-
    entry:nn ..... 6443, 6449, 6464
\l__stex_structures_present_i:w ..
    ..... 6409, 6415, 6421
\l__stex_structures_present_ii:nw
    ..... 6426, 6434
\l__stex_structures_prop .....
    6453, 6565, 6573, 6605, 6613, 6630,
    6633, 6639, 6660, 6662, 6674, 6676
\l__stex_structures_prop_do_-
    decls: ..... 6617, 6648
\l__stex_structures_prop_do_-
    notations: ..... 6618, 6697
\l__stex_structures_redo_tl ...
    .. 6592, 6616, 6642, 6689, 6700, 6715
\l__stex_structures_replace_-
    this_tl ..... 6105
\l__stex_structures_seq .....
    ..... 6614, 6661, 6675, 6699
\l__stex_structures_set_comp_tl ...
    ..... 6259, 6308, 6312, 6470, 6587
\l__stex_structures_set_custom_-
    comp:n ..... 6313, 6502
\l__stex_structures_set_customcomp:
    ..... 6286, 6311
\l__stex_structures_set_this:n ...
    ..... 6535, 6544
\l__stex_structures_set_thiscomp:
    ..... 6259, 6496
\l__stex_structures_set_thisnotation:
    ..... 6290, 6306
\l__stex_structures_shift_-
    argls:nn ..... 6120, 6125
\l__stex_structures_this_tl ...
    6271, 6471, 6472, 6475, 6490, 6547,
    6550, 6557, 6578, 6579, 6582, 6596
\l__stex_symdecl_add_decl: 3724, 3730
\l__stex_symdecl_args_tl . 3777, 3779
\l__stex_symdecl_cs .....
    .. 3775, 3779, 3969, 3971, 3973, 3999
\l__stex_symdecl_do_args: . 3794, 3853
\l__stex_symdecl_env_str .....
    ..... 3632, 3633, 3634
\l__stex_symdecl_get_from_one_-
    string:n ..... 4008, 4053
\l__stex_symdecl_get_symbol_from_-
    cs: ..... 3975, 3986
\l__stex_symdecl_get_symbol_from_-
    modules:nn ..... 4010, 4042
\l__stex_symdecl_get_symbol_from_-
    string:n ... 3976, 3978, 3981, 4003
\l__stex_symdecl_name ... 4006, 4012
\l__stex_symdecl_parse_arity: ...
    ..... 3793, 3799
\l__stex_symdecl_seq .....
    ..... 4005, 4006, 4007, 4011
\l__stex_symdecl_set_textsymdecl_-
    macro:nnn ..... 3926, 3944
\l__stex_symdecl_sym_from_str_-
    i:nnnn ..... 4016, 4047
\l__stex_symdecl_sym_i_finish:nnnnnnN
    ..... 4022, 4029

```

```

\__stex_symdecl_sym_i_gobble:nnnnnn ..... 93, 123
..... 4024, 4027
\__stex_vars_add: ..... 4734, 4762
\l__stex_vars_args_tl ... 4822, 4824
\l__stex_vars_bind_bool .....
..... 4717, 4720, 4722, 4806
\__stex_vars_check_var:nnnnnnnnN ..... 93
..... 4872, 4876
\l__stex_vars_cs ..... 4820, 4824
\__stex_vars_get_var:n .....
..... 4870, 4888, 4896
\__stex_vars_html: ..... 4736, 4789
\__stex_vars_macro: ..... 4735, 4776
\__stex_vars_set_vars:nnnnnnN .....
..... 4857, 4878, 4881
\sTeX/ComputerScience/Software ..... 14
stex_annotation_env (env.) ..... 130
\stexcommentfont .....
... 7176, 7246, 7313, 7335, 7398, 7419
\stexdoctitle ..... 1235,
1294, 1306, 2495, 7966, 8106, 8289
\STEXexport ..... 48, 75, 76, 110, 2544
\stexhtmlfalse ..... 318
\stexhtmltrue ..... 315
\STEXInternalAssocArgMarkerI ..... 116
\STEXInternalAssocArgMarkerII ..... 116
\STEXInternalNotation .. 4194, 4517, 4537
\STEXInternalSetSrefSymURL .....
..... 1492, 1793, 1796, 1799
\STEXInternalSrefRestoreTarget .....
..... 1490, 1516, 1716
\STEXInternalSymbolAfterInvocationTL .....
..... 116, 120
\STEXInternalTermMathArgiii ..... 116
\STEXInternalTermMathAssocArgiiii ..... 116
\STEXInternalTermMathOMAiii ..... 116
\STEXInternalTermMathOMBiii ..... 116
\STEXInternalTermMathOMSOrOMViii .. 116
\STEXinvisible . 70, 4291, 5155, 7226, 7281
\STEXRestoreNotsEnd .... 2396, 2420, 2427
\stextstyle ..... 122
\stextstyleassertion ..... 93
\stextstyleassign ..... 93
\stextstyleassignMorphism ..... 93
\stextstylecopymod ..... 93
\stextstylecopymodule ..... 93
\stextstyledefinition ..... 93
\stextstyleexample ..... 93
\stextstyleextstructure ..... 93
\stextstyleimportmodule ..... 93, 122
\stextstyleinterpretmod ..... 93
\stextstyleinterpretmodule ..... 93
\stextstylemathstructure ..... 93
\stextstyleMMTinclude ..... 93
\stextstylemodule ..... 93, 123
\stextstylenotation ..... 93
\stextstyleparagraph ..... 93
\stextstyleproof ..... 93
\stextstylerealization ..... 93
\stextstylerealize ..... 93
\stextstylerenamedecl ..... 93
\stextstylerequiremodule ..... 93
\stextstylepsfsketch ..... 93
\stextstylesubproof ..... 93
\stextstylesymdecl ..... 93
\stextstylesymdef ..... 93
\stextstyletextsymdecl ..... 93
\stextstyleusemodule ..... 93
\stextstylevardef ..... 93
\stextstylevarnotation ..... 93
\stextstylevarseq ..... 93
\stopssolutions ..... 100, 8409
str commands:
\c_backslash_str ..... 689
\c_colon_str ..... 820, 875, 1170, 2339
\c_dollar_str ..... 4283
\c_hash_str ..... 588, 2601, 4290
\c_percent_str ..... 47, 1037
\str_case:Nn ..... 1174
\str_case:nn ..... 4616, 5404
\str_case:mntF ..... 1405,
3803, 4357, 4379, 5432, 5460, 7745
\str_clear:N ..... 379, 380, 387, 405, 688,
895, 1055, 1130, 1576, 1891, 2053,
2155, 2337, 2471, 2779, 2974, 3062,
3089, 3104, 3389, 3520, 3521, 3660,
3661, 3662, 3663, 3673, 3674, 3696,
3876, 3915, 3916, 3966, 3967, 4102,
4103, 4104, 4105, 4860, 4887, 4895,
5492, 6071, 6135, 6752, 6753, 6755,
6802, 6980, 7033, 7092, 7341, 7572,
7874, 8145, 8146, 8240, 8241, 8342
\str_const:Nn ..... 7664
\str_count:n ..... 529, 534
\str_gset:Nn ..... 1493, 1811
\str_gset_eq:NN ..... 1050, 2845, 2846
\str_if_empty:NTF ... 82, 426, 432,
599, 605, 695, 715, 902, 1043, 1045,
1064, 1122, 1195, 1581, 1587, 1593,
1640, 1641, 1656, 1667, 1687, 1710,
1721, 1729, 1749, 1774, 1837, 2281,
2524, 2761, 2784, 2978, 2982, 3064,
3226, 3402, 3532, 3546, 3633, 3719,
3753, 3756, 3759, 3762, 3889, 3960,
4209, 4221, 4230, 4277, 4728, 4794,
4797, 4800, 4803, 4889, 4897, 4921,
4924, 4991, 4994, 4997, 5000, 5992,
6081, 6130, 6137, 6567, 6780, 6781,

```

6786, 6803, 6823, 6891, 6986, 7039,  
 7286, 7295, 7364, 7378, 7384, 7609,  
 7735, 7794, 7957, 8271, 8361, 8401,  
 8418, 8458, 8467, 8507, 8516, 8556  
 $\backslash$ str\_if\_empty:nTF . . . . . 465, 492,  
 676, 727, 822, 1500, 1851, 2474, 3375  
 $\backslash$ str\_if\_eq:NNTF . . . . . 165, 776  
 $\backslash$ str\_if\_eq:nnTF . . . . . . . . .  
 . . . . . 139, 155, 156, 157, 179,  
 295, 528, 533, 563, 579, 588, 600,  
 606, 690, 728, 729, 745, 1624, 1708,  
 1750, 1755, 1757, 1808, 2261, 2343,  
 2582, 2615, 2720, 2991, 2996, 3005,  
 3008, 3184, 3210, 3394, 3410, 3634,  
 4212, 4224, 4236, 4877, 4880, 5071,  
 7003, 7798, 7801, 7866, 8025, 8588  
 $\backslash$ str\_if\_eq\_p:nn 4018, 4019, 4058, 4059  
 $\backslash$ str\_if\_exist:NTF . . . . . 226, 1834  
 $\backslash$ str\_if\_in:NnTF . . . . . 4290, 5973, 6935  
 $\backslash$ str\_item:Nn . . . . . 690, 745, 3858  
 $\backslash$ str\_lowercase:n . . . . . 8588  
 $\backslash$ str\_map\_break: . . . . . 3804  
 $\backslash$ str\_map\_break:n . . . . . 3805, 3809, 3813,  
 3817, 3821, 3825, 3829, 3833, 3837  
 $\backslash$ str\_map\_inline:Nn . . . . . 3802  
 $\backslash$ str\_new:N . . . . . . . . .  
 . . . . . 128, 1555, 1556, 2019, 2266, 3692  
 $\backslash$ str\_put\_right:Nn . . . . . 7099  
 $\backslash$ str\_range:Nnn . . . . . 2013  
 $\backslash$ str\_range:nnn . . . . . 529, 534, 568  
 $\backslash$ str\_replace\_all:Nnn . . . . . 689  
 $\backslash$ str\_set:Nn . . . . . 41, 49, 130, 178, 221,  
 683, 684, 687, 703, 820, 1077, 1160,  
 1172, 1173, 1502, 1504, 1506, 1570,  
 1571, 1599, 1610, 1618, 1625, 1689,  
 1695, 1700, 1706, 1715, 1853, 1860,  
 1865, 1976, 1984, 2011, 2049, 2300,  
 2316, 2339, 2409, 2638, 2792, 2797,  
 2800, 2811, 2834, 2835, 3018, 3019,  
 3020, 3053, 3058, 3063, 3068, 3074,  
 3082, 3084, 3090, 3094, 3095, 3096,  
 3105, 3109, 3110, 3111, 3118, 3125,  
 3128, 3131, 3137, 3144, 3147, 3150,  
 3168, 3195, 3240, 3438, 3528, 3534,  
 3538, 3668, 3698, 3720, 3807, 3811,  
 3815, 3819, 3823, 3827, 3831, 3835,  
 3839, 3888, 3890, 3892, 3894, 3990,  
 3991, 4031, 4032, 4063, 4064, 4143,  
 4172, 4177, 4296, 4336, 4347, 4473,  
 4727, 4729, 4861, 4905, 4907, 4920,  
 4922, 4925, 5180, 5181, 5594, 5638,  
 5679, 5971, 5974, 6053, 6082, 6091,  
 6185, 6249, 6253, 6255, 6563, 6568,  
 6792, 6806, 6812, 6933, 6936, 6983,

7036, 7288, 7297, 7386, 7608, 7610,  
 8359, 8360, 8363, 8416, 8417, 8420,  
 8465, 8466, 8469, 8514, 8515, 8518  
 $\backslash$ str\_set\_eq>NN . . . . . 1061,  
 1561, 1562, 2511, 2782, 2812, 3917,  
 3918, 4114, 4142, 4433, 4443, 4462,  
 4476, 4477, 4751, 4946, 5935, 5948,  
 5961, 6075, 6139, 6782, 6791, 6841,  
 6893, 7272, 7353, 7571, 7575, 7599,  
 7601, 7614, 7628, 7630, 7631, 8293  
 $\backslash$ str\_uppercase:n . . . . . 7866  
 $\backslash$ l\_tmpa\_str . . . . . 152, 153, 154,  
 155, 156, 157, 158, 162, 178, 182,  
 183, 184, 186, 1891, 1892, 1893,  
 1898, 1906, 2797, 2800, 2805, 2812  
 $\backslash$ subparagraph . . . . . 26, 71  
subproof (env.) . . . . . 7272  
 $\backslash$ subproof . . . . . 7230, 7331  
 $\backslash$ subproofautorefname . . . . . 7272  
 $\backslash$ subsection . . . . . 25, 26, 71  
 $\backslash$ subsubsection . . . . . 26, 71  
 $\backslash$ svar . . . . . 84, 3746, 4902  
 $\backslash$ symbol . . . . . 78  
 $\backslash$ symdecl . . . . . 22,  
 23, 30, 31, 38, 39, 46, 76–78, 88,  
 93, 113, 118, 124, 131, 132, 2960,  
 3692, 7472, 7474, 7503, 7511, 7514  
 $\backslash$ symdef . . . . . 31, 32, 34, 39, 40, 77,  
 84, 113, 118, 124, 2962, 4453, 7456,  
 7459, 7465, 7467, 7469, 7471, 7481,  
 7482, 7483, 7484, 7490, 7498, 7509,  
 7518, 7519, 7522, 7525, 7527, 7529  
 $\backslash$ Symname . . . . . 78, 89, 5877  
 $\backslash$ symname 18, 19, 23, 61, 78, 79, 89, 91, 5877  
 $\backslash$ symref 18, 19, 23, 47, 77, 78, 82, 89, 91, 5877  
 $\backslash$ symrefemph . . . . . 91, 92, 5989  
 $\backslash$ symuse . . . . . 47, 78, 5126, 5205, 5682, 6359,  
 6366, 6377, 6459, 6524, 6594, 6595  
sys commands:  
 $\backslash$ sys\_get\_shell:nnN . . . . . 38  
 $\backslash$ sys\_if\_platform\_windows:TF . . . . .  
 . . . . . 44, 681, 740, 967, 1036

**T**

$\backslash$ target . . . . . 123, 126–128  
 $\backslash$ test . . . . . 105  
test . . . . . 99  
 $\backslash$ testemptypage . . . . . 104  
 $\backslash$ testemptypage . . . . . 8828  
testheading (env.) . . . . . 105  
 $\backslash$ testheading . . . . . 8785  
 $\backslash$ testnewpage . . . . . 104  
 $\backslash$ testnewpage . . . . . 8832  
 $\backslash$ testsmallspace . . . . . 104, 104, 104

\testspace	104
\testspace	8831
\TeX	22, 23
\tex	22, 23
TEX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>&lt;</sub> commands:	
\@	2068, 2071, 2075
\@addtoreset	8253
\@arabic	8117
\@author	8096
\@auxout	1786, 7651, 8335
\@bonuspointsfalse	8816
\@bonuspointstrue	8821
\@currentHref	7288, 7297, 7386
\@currentcounter	1519
\@currentlabel	1520, 7287, 7288, 7296, 7297, 7385, 7386, 7956
\@currentlabelname	1522, 1523
\@currenvir	7840, 7841
\@dblarg	8089, 8098
\@gobble	354
\@ifnextchar	353
\@ifstar	8068
\@mainmatterfalse	1449, 1459
\@mainmattertrue	1470, 1481
\@notprerr	133, 1415
\@onlypreamble	133, 1415
\@rusttexfalse	291
\@title	1264, 1265, 8105
\activate@excursion	8129, 8134
\bb@loaded	8221, 8680
\beamer@shortauthor	8092, 8094, 8109
\beamer@shorttitle	8101, 8103, 8112
\c@chapter	7821
\c@framenumber	8117
\c@part	7833
\compemph@uri	92, 120, 4333, 5981, 5989
\correction@table	8764
\defemph@uri	92, 5989
\define@key	2021, 2035, 2048
\Gin@height	8862, 8865, 8870, 8875
\Gin@ewidth	8059, 8060, 8861, 8871, 8875
\Gin@exclamation	8861, 8862, 8870
\Gin@mrepos	.. 2022, 2025, 2029, 2050, 2054, 2059
\hwexam@bonuspts	8818
\hwexam@checktime	8812
\hwexam@duration	8788, 8791, 8797, 8802
\hwexam@kw@forgrading	8780
\hwexam@kw@grade	8781
\hwexam@kw@probs	8781
\hwexam@kw@pts	8782
\hwexam@kw@reached	8783
\hwexam@kw@sum	8781
\hwexam@kw@testemptypage	8829
\hwexam@min	8789, 8796, 8801, 8812
\hwexam@minutes@kw	8789
\hwexam@reqpts	8798, 8803, 8815, 8819
\hwexam@tools	8799, 8804
\hwexam@totalmin	8811, 8812
\hwexam@totalpts	8810, 8819
\if@bonuspoin	8814
\if@rustex	299
\itemize@inner	7906, 7912, 7921
\itemize@label	7910, 7913, 7916
\itemize@level	7904, 7909, 7912, 7921
\itemize@outer	7905, 7909
\lst@mhrepos	2036, 2039, 2043
\ltx@ifpackageloaded	134, 138, 2020, 2034, 2047, 7166, 8220, 8610, 8679
\m@switch	5848
\mdf@patchamsthm	7930
\metakeys@show@keys	7907
\notesslides@slidelabel	7933, 7948
\ns@author	8089, 8090
\ns@title	8098, 8099
\pgf@temp	2065, 2066, 2067
\pgfkeys@spdef	2065
\pgfutil@empty	2067
\pgfutil@InputIfFileExists	2074
\prematurestop@endsfragment	.. 7839, 7842, 7848
\problem@kw@*	8217
\problem@kw@correct	8618
\problem@kw@minutes	8316, 8569
\problem@kw@points	8313, 8564
\problem@kw@wrong	8620
\problem@restore	8336, 8340, 8758
\stex@backend	.. 129, 289
\symrefemph@uri	.. 91, 92, 5893, 5903, 5914, 5989
\var@emph@uri	.. 92, 120, 5934, 5947, 5960, 5989
\texname	23
\text	19
\textbf	18, 6031, 7726, 8327, 8729, 8734
\textcolor	8648
\textsymdecl	22, 23, 77, 118, 2961, 3874
\textwarning	97
\textwidth	8057, 8079, 8777
\the	244, 245, 247, 249, 251, 2068, 2069, 2070
\theassignment	8722, 8729
\thechapter	1325, 1366, 1394, 1429, 7752, 7757, 7761, 7765, 7769, 7773
\theframenumber	7956
\theparagraph	7769, 7773
\thepart	1321, 1363, 1390, 1423, 7747
\theplainsproblem	8257, 8258

```

\thesection ..... 7757, 7761, 7765, 7769, 7773, 8258
\thesproblem . 8254, 8258, 8327, 8336, 8722
\thes subparagraph ..... 7773
\thes subsection ... 7761, 7765, 7769, 7773
\thes subsection .. 7765, 7769, 7773
\this ..... 56–58, 85, 86, 5177,
       6050, 6475, 6552, 6553, 6557, 6582
\thisarchive ..... 3266, 3307
\thisargs ..... 3708, 4460
\thiscopyname ..... 3458, 3478, 3502, 3572, 3592, 3613
\thisdeclname .... 3705, 3933, 4143, 4457
\thisdecluri ..... 3704, 3932, 4144, 4147, 4456, 4464
\thisdefiniens ..... 3707, 4459
\thisfor ..... 6842
\thismodulename ..... 122, 2512, 3043, 3268, 3309, 8294
\thismoduleuri ..... 122, 2511, 3042, 3267, 3308, 3459,
       3479, 3503, 3573, 3593, 3614, 8293
\thisname ..... 6841
\thisnotation 4145, 4444, 4463, 4752, 4947
\thisnotationvariant ..... 4142, 4443, 4462, 4751, 4946
\ThisStyle ..... 5848
\thisstyle ..... 122, 452, 455,
       456, 457, 503, 506, 507, 508, 509,
       517, 520, 521, 3044, 3269, 3310,
       3709, 3934, 4442, 4461, 4750, 4945
\thistitle ..... 92, 122, 2493,
       2494, 2495, 6840, 8295, 8328, 8329
\thistype ..... 3706, 4458
\thisvarname ..... 4441, 4445, 4749, 4753, 4944, 4948
\throwaway ..... 124
\tikzinput ... 107, 2059, 8854, 8859, 8885
tikzinput commands:
  \c_tikzinput_image_bool 28, 8843, 8850
\tiny ..... 7934, 7948
\ttitle ..... 105
\ttitle ..... 8098
tl commands:
  \tl_clear:N ..... 393,
       452, 503, 764, 1337, 1714, 2150,
       2269, 2475, 2739, 3044, 3269, 3310,
       3522, 3675, 3676, 3677, 3709, 3854,
       3877, 3878, 3914, 3934, 4164, 4173,
       4269, 4412, 4442, 4461, 4500, 4695,
       4750, 4910, 4933, 4945, 5175, 5185,
       5282, 5295, 5493, 5880, 5881, 5882,
       5979, 6070, 6547, 6575, 6616, 6756,
       6757, 6758, 7059, 7060, 7062, 7063,
       7064, 7065, 7342, 7343, 8144, 8595,
       8597, 8598, 8697, 8698, 8699, 8769,
       8770, 8771, 8796, 8797, 8798, 8799
  \tl_const:Nn ..... 5820, 5821
  \tl_count:N ..... 5437, 5443
  \tl_count:n ..... 5568, 5579
  \tl_gclear:N ..... 2295
  \tl_gput_left:Nn ..... 365, 367, 368
  \tl_gput_right:Nn ..... 245, 2096,
       2100, 2119, 2272, 2536, 3277, 8130
  \tl_gset:Nn ..... 247,
       360, 361, 1016, 1236, 1243, 2120,
       2124, 4180, 5138, 5142, 8761, 8762
  \tl_gset_eq:NN ..... 39, 2771
  \tl_head:N .. 879, 933, 3973, 6497, 6503
  \tl_head:n ..... 547, 588, 591, 4646, 5569, 5580
  \tl_if_empty:NTF .. 517, 780, 1018,
       1254, 1264, 1307, 1741, 2358, 2494,
       2504, 3327, 3549, 3736, 3767, 3770,
       3773, 3902, 4077, 4090, 4158, 4167,
       4175, 4345, 4499, 4812, 4815, 4818,
       4826, 4848, 4931, 5005, 5008, 5011,
       5019, 5249, 5261, 5352, 5372, 5457,
       5596, 5709, 5722, 5764, 5793, 6052,
       7172, 7192, 7195, 7583, 8060, 8151,
       8283, 8288, 8328, 8618, 8620, 8622,
       8717, 8730, 8735, 8738, 8788, 8815
  \tl_if_empty:nTF ..... 362, 546,
       562, 578, 587, 744, 753, 875, 876,
       938, 945, 1092, 1560, 1718, 1740,
       1895, 1917, 1948, 2002, 2192, 2558,
       2573, 2625, 2685, 2710, 2780, 2795,
       2894, 2953, 3054, 3088, 3103, 3117,
       3136, 3167, 3194, 3383, 3971, 4084,
       4097, 4547, 4597, 4904, 5400, 5421,
       5497, 5520, 5923, 6120, 6244, 6293,
       6296, 6341, 6351, 6440, 6442, 6536,
       6545, 6640, 6650, 6714, 6859, 6981,
       7034, 7050, 7546, 8091, 8100, 8651
  \tl_if_empty_p:N ..... 770
  \tl_if_eq:NNTF 1134, 6138, 6497, 6503
  \tl_if_eq:NnTF ..... 8177
  \tl_if_eq:nnTF .. 2427, 4671, 5065, 5580, 5625, 5663
  \tl_if_exist:NTF ..... 203,
       244, 259, 298, 456, 508, 520, 1280,
       1289, 1522, 1682, 2528, 6205, 7778
  \tl_if_in:NnTF ..... 2207, 2225, 2229
  \tl_item:nn ..... 5571
  \tl_map_inline:Nn ..... 2160
  \tl_map_inline:nn ..... 210, 214
  \tl_new:N ..... 958, 1028, 1234, 1557, 2092, 2093,

```

2116, 2127, 2445, 4716, 5131, 5135  
 \tl\_put\_left:Nn ..... 4586, 4587, 5326, 8263, 8710  
 \tl\_put\_right:Nn ..... 1210, 2346, 2721, 2725, 3856,  
 3857, 4275, 4281, 4291, 4507, 4512,  
 4515, 4618, 4626, 4631, 4639, 4702,  
 5163, 5187, 5197, 5307, 6546, 6642,  
 6689, 6700, 6715, 8773, 8774, 8775  
 \tl\_range:nnn ..... 564, 580  
 \tl\_set:Nn ..... 204, 241, 344,  
 447, 466, 468, 487, 488, 493, 494,  
 496, 497, 743, 777, 857, 951, 1015,  
 1272, 1313, 1447, 1457, 1572, 1738,  
 2022, 2025, 2512, 2621, 2623, 2626,  
 2841, 3022, 3023, 3024, 3025, 3026,  
 3266, 3307, 3458, 3478, 3502, 3529,  
 3535, 3539, 3572, 3592, 3613, 3704,  
 3777, 3932, 3993, 3994, 3995, 3996,  
 3997, 4034, 4035, 4036, 4037, 4038,  
 4066, 4067, 4068, 4069, 4070, 4144,  
 4159, 4168, 4193, 4210, 4213, 4222,  
 4225, 4231, 4238, 4257, 4456, 4504,  
 4517, 4535, 4556, 4571, 4572, 4584,  
 4623, 4636, 4655, 4668, 4694, 4777,  
 4822, 4863, 4864, 4865, 4866, 4867,  
 4972, 5015, 5099, 5132, 5137, 5139,  
 5146, 5176, 5182, 5183, 5184, 5296,  
 5317, 5437, 5443, 5456, 5462, 5465,  
 5468, 5511, 5597, 5629, 5631, 5681,  
 5824, 5825, 5869, 5870, 6105, 6117,  
 6186, 6259, 6271, 6307, 6308, 6312,  
 6315, 6322, 6358, 6397, 6471, 6477,  
 6487, 6550, 6552, 6577, 6578, 6584,  
 6593, 6641, 6643, 6685, 6690, 6702,  
 6705, 6709, 6712, 6717, 6720, 6724,  
 6727, 7165, 7171, 7747, 7748, 7752,  
 7753, 7757, 7758, 7761, 7762, 7765,  
 7766, 7769, 7770, 7773, 7774, 7789,  
 8238, 8239, 8326, 8350, 8609, 8616,  
 8626, 8749, 8755, 8756, 8812, 8818  
 \tl\_set\_eq:NN ..... 343, 364,  
 366, 371, 1563, 2493, 3042, 3043,  
 3267, 3268, 3308, 3309, 3459, 3479,  
 3503, 3573, 3593, 3614, 3705, 3706,  
 3707, 3708, 3933, 4147, 4215, 4227,  
 4251, 4399, 4407, 4441, 4457, 4458,  
 4459, 4460, 4503, 4740, 4749, 4840,  
 4845, 4849, 4929, 4944, 5376, 5516,  
 5557, 5636, 6403, 6475, 6557, 6582,  
 6840, 7061, 8294, 8295, 8810, 8811  
 \tl\_set\_rescan:Nnn ..... 132, 183  
 \tl\_tail:n ..... 547, 548, 593, 2192, 5542, 6367  
 \tl\_to\_str:n ..... 59,  
 62, 90, 93, 103, 115, 211, 215, 216,  
 240, 343, 565, 581, 677, 699, 706,  
 805, 823, 825, 828, 847, 853, 904,  
 974, 1066, 1170, 1242, 1257, 1493,  
 1780, 1804, 1831, 1832, 1838, 1887,  
 1935, 2082, 2104, 2120, 2123, 2124,  
 2161, 2164, 2206, 2208, 2224, 2226,  
 2230, 2248, 2257, 2401, 2403, 2404,  
 2407, 2408, 2438, 2439, 2497, 2565,  
 2566, 2567, 2589, 2822, 2976, 3051,  
 3326, 3335, 3523, 4254, 4555, 4569,  
 4648, 4693, 5482, 6115, 6148, 6198,  
 6265, 6285, 6289, 6448, 6576, 6631,  
 6661, 6675, 6699, 6737, 6773, 7648  
 \tl\_trim\_spaces:N ..... 42  
 \l\_tmpa\_tl ..... 38, 39, 41, 4285,  
 4695, 4702, 4709, 5126, 5424, 5430,  
 5435, 5437, 5441, 5443, 5446, 5455,  
 5457, 5462, 5465, 5468, 5683, 6355,  
 8626, 8638, 8639, 8769, 8773, 8781  
 \l\_tmpb\_tl .....  
 5424, 5430, 5432, 5446, 5456, 5460,  
 8616, 8629, 8631, 8770, 8774, 8782  
 tmpc commands:  
 \l\_tmpc\_t1 ..... 8771, 8775, 8783  
 \to ..... 7491, 7494, 7504, 7505  
 \today ..... 8119  
 \TODO ..... 4984, 4985, 6342, 8175  
 todo internal commands:  
 \l\_\_todo\_mmt\_module\_str 7571, 7576,  
 7589, 7592, 7595, 7599, 7614, 7630  
 \\_\_todo\_newlabel:n ..... 7647, 7654  
 \l\_\_todo\_old\_metagroup\_cd 7615, 7627  
 \\_\_todo\_old\_newlabel: ... 7648, 7653  
 \l\_\_todo\_stex\_module\_str .....  
 ..... 7575, 7576, 7601, 7628  
 token commands:  
 \c\_math\_subscript\_token .....  
 ..... 48, 76, 4381, 4382, 4385,  
 4386, 4390, 6487, 7496, 7507, 7509  
 \topsep ..... 7181  
 \type ..... 105  
  
**U**  
 \undefined ..... 123, 34  
 \univ ..... 62  
 \unless ..... 7840  
 \uppercase ..... 5917  
 \uri ..... 127, 128  
 use commands:  
 \use:N ..... 242, 375,  
 453, 457, 459, 504, 509, 511, 518,  
 521, 523, 1308, 1310, 1717, 1740,

1835, 1840, 2161, 2164, 2639, 3904, 5041, 5251, 5277, 6405, 7779, 7785	\varempf . . . . . 92, 5989
\use:n . . . . . 1823, 2012, 2020, 2034, 2047, 3774, 4819, 5012, 5207, 5297, 5329, 6457, 6564	\Varname . . . . . 5877 \varname . . . . . 5877 \varnotation . . . . . 84, 4430 \varref . . . . . 5877
\use:nn . . . . . 48, 191, 194, 200, 540, 1310, 2436, 2460, 2791, 2836, 2992, 2997, 3731, 3898, 4148, 4299, 4338, 4446, 4465, 4754, 4949, 5096, 5118, 5123, 5339, 5542, 5589, 5674, 5849, 6086, 6118, 6523, 6590, 6632, 6794, 7617, 8155, 8352, 8751	\varseq . . . . . 42, 84, 4918 \vbox . . . . . 124, 7167, 7928, 8017, 8032, 8382, 8439, 8488, 8537, 8611 vbox commands: \ vbox_set:Nn . . . . . 2129
\use_i:nn . . . . . 6564, 8774	\vdash . . . . . 7519
\use_ii:nn . . . . . 1827, 2913, 2928, 6604	\vfill . . . . . 7783, 8829
\use_none:nnnnnn . . . . . 3011	\vM . . . . . 52 \vMs . . . . . 54
\usebox . . . . . 8011	\vn . . . . . 42
\usemodule . . . . . 14, 17, 21, 22, 25, 36, 48, 82, 83, 85, 201, 416, 423, 3030	\vop . . . . . 44, 46 \vrule . . . . . 7167, 8079, 8611
\usepackage . . . . . 2012, 7854	\vskip . . . . . 7167, 8078, 8080, 8611
\useSGvar . . . . . 98, 8168	\vspace . . . . . 8831
\usestructure . . . . . 85, 6733	
\usetHEME . . . . . 7854	<b>W</b>
\usetikzlibrary . . . . . 69, 2081, 8853	\wff . . . . . 19
	\withbrackets . . . . . 81, 5867, 5875
	<b>X</b>
	\xspace . 1280, 1289, 3948, 3949, 3955, 3956
	<b>Y</b>
	\yield . . . . . 7235, 7426, 7429