

The $\text{\textit{S}\text{\textit{T}\text{\textit{E}\text{\textit{X}}}}}_3$ Package Collection*

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

2023-12-08

Contents

1	Introduction & Setup	7
1.1	What is $\text{\textit{S}\text{\textit{T}\text{\textit{E}\text{\textit{X}}}}}_3$?	7
1.2	The $\text{\textit{S}\text{\textit{T}\text{\textit{E}\text{\textit{X}}}}}_3$ package	8
1.3	What is MMT?	8
1.4	Math archives and the MathHub Directory	8
1.5	Setting Up the $\text{\textit{S}\text{\textit{T}\text{\textit{E}\text{\textit{X}}}}}_3$ IDE	9
I	Tutorial	11
2	The Basics	12
2.1	Text symbols	15
2.1.1	Using Modules & Search in the IDE	15
2.2	Symbol References	18
2.3	Modules and Simple Symbol Declarations	21
2.4	Documenting Symbols	24
2.5	Sectioning and Reusing Document Fragments	26
2.6	Building and Exporting HTML	28
3	Mathematical Concepts	31
3.1	Simple Symbol Declarations	31
3.1.1	Semantic Macros and Notations	31
3.1.2	Types and Variables	34
3.1.3	Flexary Macros and Argument Modes	39
3.1.4	Precedences	41
3.1.5	Implicit Arguments	41
3.1.6	Finishing Equality	43
3.1.7	Variable Sequences	43
3.2	Statements	44
3.2.1	Definitions	44
	Semantic Macros in Text Mode	46

*Version 3.4.0 (last revised 2023-12-08)

Definientia	47
Using Symbols Without Semantic Macros and Exporting Code in Modules	48
3.2.2 Assertions	49
3.2.3 Proofs	52
3.3 Mathematical Structures	52
3.3.1 Declaring and Using Structures	52
Instantiating Structures	54
3.3.2 Extending Structures and Axioms	55
Conservative Extensions	56
3.3.3 Nesting Structures and \this	57
3.4 Complex Inheritance and Theory Morphisms	59
3.4.1 Glueing Structures Together	61
3.4.2 Realizations	63
4 Extensions for Education	65
4.1 Slides and Course Notes	65
4.2 Problems and Exercises	65
4.2.1 Background	65
4.2.2 The Package, Options, and Configuration	66
4.2.3 (Open) Problems	67
4.2.4 Structured Problems	68
4.2.5 Single/Multiple Choice Blocks	69
4.2.6 Filling-In Concrete Solutions	71
4.2.7 Answer Classes	72
4.2.8 Including Problems	72
4.2.9 Testing and Spacing	72
4.3 Homework Assignments and Exams	73
4.3.1 Introduction	73
4.3.2 Package Options	73
4.3.3 Assignments	73
4.3.4 Including Assignments	74
4.3.5 Typesetting Exams	74
II User Manual	76
5 Basics	77
5.1 Package and Class Options	77
5.2 Math Archives and the MathHub Directory	78
5.2.1 The Structure of Math Archives	79
5.2.2 MANIFEST.MF-Files	79
5.3 The lib-Directory	80
5.4 Basic Macros	81
6 Document Features	82
6.1 Document Fragments	82
6.2 Using and Referencing Document Fragments	83
6.3 Cross-Document References	83

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

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

13.3	Math Archives	170
13.4	Documents	174
13.4.1	Title	174
13.4.2	Sectioning	175
13.4.3	References	179
13.4.4	Inputs	187
13.5	SMS Mode	192
13.6	Modules	196
13.6.1	The <code>smodule</code> -environment	196
13.6.2	Structural Features	207
13.7	Inheritance	212
13.7.1	<code>\importmodule</code> / <code>\usemodule</code>	212
13.7.2	Theory Morphisms	218
13.8	Symbols	224
13.8.1	Declarations	224
13.8.2	Notations	233
a/B-mode argument handling	244	
13.8.3	Variables	246
13.8.4	Sequences	250
13.8.5	Expressions	254
Invoking Semantic Macros	255	
Argument Handling and Annotating	261	
Term HTML Annotations	266	
Automated Bracketing	268	
Symname and Variants	269	
Highlighting	272	
13.9	Mathematical Structures	273
13.10	Statements	286
13.11	Proofs	292
13.12	Metatheory	300
13.13	MMT Interfaces	302
14	Additional Packages	306
14.1	Implementation: The <code>notesslides</code> Package	306
14.1.1	Class and Package Options	306
14.1.2	Notes and Slides	310
14.1.3	Environment and Macro Patches	313
14.1.4	Styling Across Notes/Slides	315
14.1.5	Beamer Compatibility	315
14.1.6	TODO Excursions	316
14.2	Implementation: The <code>problem</code> Package	318
14.2.1	Package Options	318
14.2.2	Problems and Solutions	319
14.3	Implementation: The <code>hwexam</code> Package	334
14.3.1	Package Options	334
14.3.2	Assignments	335
14.3.3	Leftovers	338
14.4	Tikzinput Implementation	338



`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

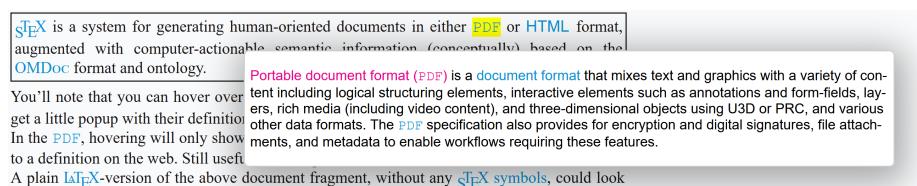


Figure 3: Definition on Hover

```

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

2.1.1 Using Modules & Search in the IDE

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

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

You can click the preview button to see the generated **HTML** for the document – the resulting window that pops up also has an **OMDoc** tab you can select, which (among other things) shows you the **semantic macros** provided by the respective **module**: In this case, it tells us that there is a **text symbol** named “**s_TeX**” with **semantic macro** `\stex` in the **module** `mod/systems/tex?sTeX` that is in the `\sTeX/ComputerScience/Software` archive. It produces the presentation “**s_TeX**” 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 **s_TeX symbol** in a document – which we have open in the **IDE** – we simply click on the **use** button, and the **IDE** will automatically insert the line `\usemodule[sTeX/ComputerScience/Software]{mod/systems/tex?sTeX}`, making all **symbols** in that **module** available to use – in particular, we can now use the `\stex` **semantic macro** instead of the plain, non-semantic `\sTeX macro` – that is, of course, after we include the `stex` **package** first.

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

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

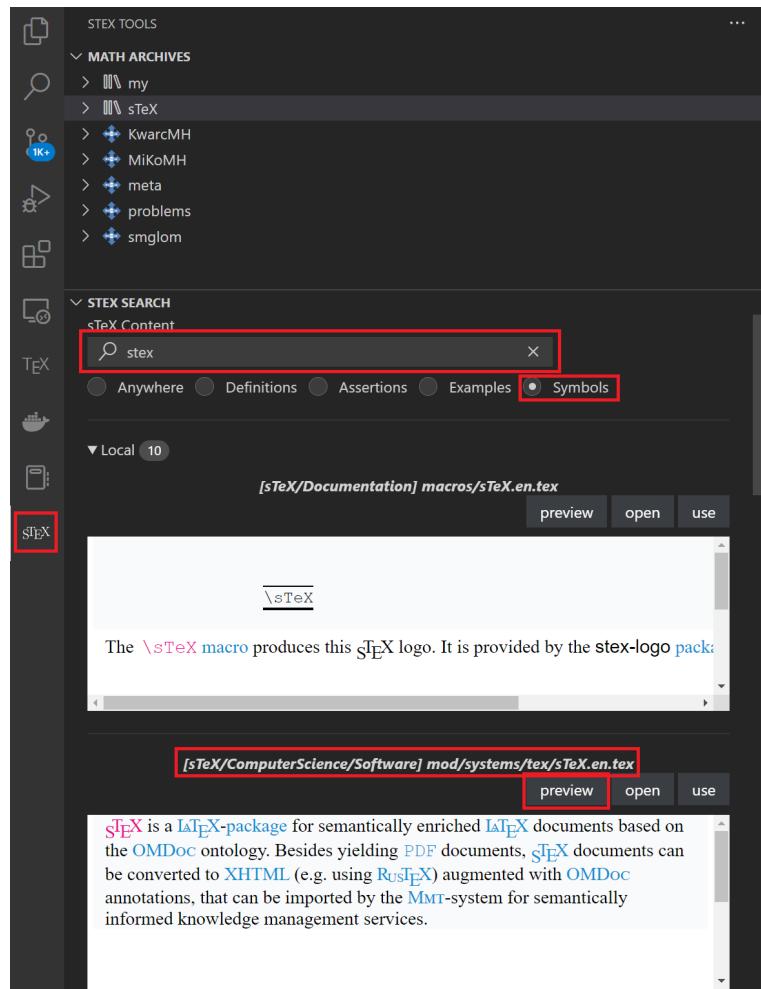


Figure 5: Search in the STeX IDE



Figure 6: OMDoc Preview

Example 2

Input:

```
File [sTeX/Documentation]tutorial/intro/intro1stex.en.tex
1 \documentclass{article}
2 \usepackage{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.

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^AT_EX**, 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^AT_EX**”. But if we look at the documentation, we will note that *more* is highlighted:

At its core is the **sTeX** package for **L^AT_EX**, 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^AT_EX** 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^AT_EX**, **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](#)”.

sTeX

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^AT_EX](#), 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^AT_EX](#) [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 “`\TEX`” 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).

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

2.5 Sectioning and Reusing Document Fragments

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

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

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

In there, we want sections as follows:

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

We could of course do the following:

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

...but this approach has two drawbacks:

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

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

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

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

Instead of using `\section` and `\subsection`, **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 `\ifinptref` conditional: by wrapping the header in an `\ifinptref \else... \fi`, it will only be processed if the file is *not* being loaded using `\inputref`. `\ifinptref` 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



Figure 14: The Build PDF/XHTML/OMDoc Button

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

This will do the following:

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

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



Figure 15: Building Archives in the [IDE](#)

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

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

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



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

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

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

Chapter 3

Mathematical Concepts

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

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

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.

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

3.1.2 Types and Variables

You might have noticed – after you save the file – that the expressions `\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

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

\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**

<http://mathhub.info/sTeX/MathTutorial/mod/Equality1/Equality?en?term 1?definition>

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

Figure 18: Type Checking Proof Tree

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

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



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_EX-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 N are the numbers 0,1,2,....
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` definitions¹:

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.

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

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

Example 7

Input:

```
\begin{sparagraph}[style=symdoc]
\vardef{va}{name=a,type=\Nat}{a}\vardef{vb}{name=b,type=\Nat}{b}
\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.

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 <code>function space (\funspace{a_1, ..., a_n}{b})</code>		
Type	$(A : \text{SET}, B : \text{SET}) \rightarrow \text{SET}$	
Notations	id	notation
	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](#).

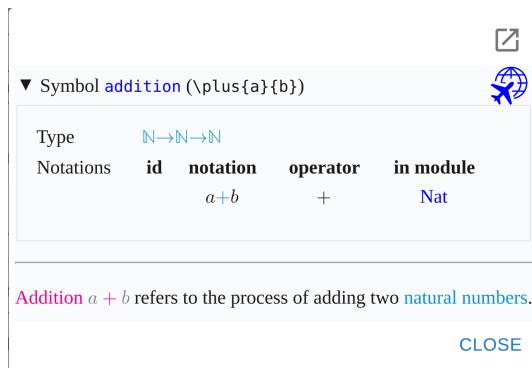


Figure 20: On-Click Popup in the `HTML`

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** 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`².

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

▼ Document Elements

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

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

Figure 21: Inferred Type

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:

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

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

Example 8

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

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

- in `\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{paragraph}[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{paragraph}
```

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 $a \cdot b \cdot c$.

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

3.1.4 Precedences

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

We do *not*. Instead, we can assign *precedences* to *notations* to have *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 false³, and “*equal*” takes two argu-

³Assuming 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	$a \equiv b$	\equiv	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 N (by the $\underbrace{\dots}_{\text{N}}$), 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_ℓ^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 $m_1^1 = \dots = m_\ell^k$.

TODO⁴

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.

⁴TODO: 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})\{ \vA, \vA \} \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}) \vA \} \$) }.
23 \end{sdefinition}
24 %

```

Output:

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

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

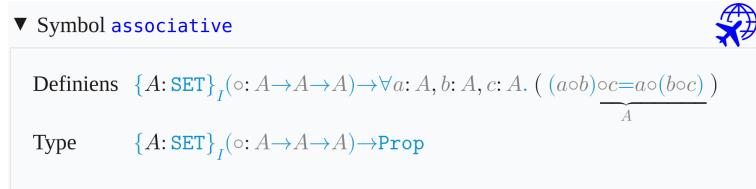


Figure 24: Type Inferred from Definiens

Using Symbols Without Semantic Macros and Exporting Code in Modules

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

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

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

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

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

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

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

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

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

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

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



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

Exercise

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

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

TODO⁵

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:

⁵TODO: intent?

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

which yields

Addition is associative

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

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

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

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

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

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

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

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

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

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

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

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

$$\triangleright \text{Assertion addition is associative} \vdash \text{apply } \left(\text{apply } \left(\underbrace{\text{associativeN}}_{\mathbb{N}} \right) + \right)$$

Figure 25: Assertion Statement in **OMDoc**

Not exactly pretty – the **OMDoc** tab uses **notations** to render content, and we did not provide any for **associative**.

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

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 .
sTeX
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
       $\varphi(\text{plus}\{m, 1\})$  also holds,
  \end{enumerate}
  then  $\varphi(n)$  holds for all  $n \in \mathbb{N}$ .
\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 φ , 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”.

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

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

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

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

Hence, if we mark the variable φ 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 \definename{magma} is a \sr{mathstruct}{structure} $\magma$,
15    where $\univ$ is a \sn{collection} and $\op$ is
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
<code>\$\vM\$</code>	M
<code>\$\vM{}\$</code>	$\langle U_M, o_M \rangle$
<code>\$\vM{univ}\$</code>	U_M
<code>\$\vM{op}!\$</code>	o_M
<code>\$\vM{op}{a,b,c}\$</code>	$a o_M b o_M c$

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{op}} 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:

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

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

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** $\langle U, o \rangle$, where o is **associative**.

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

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



Figure 27: Axioms in OMDoc

Conservative Extensions

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

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

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

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

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

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

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

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

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

Consider the following conservative extension:

Example 13

Input:

```
File [sTeX/MathTutorial]algebra/MagmaSquare.en.tex
7 \begin{extstructure}{magma}
8   \begin{sdefinition}[macro=sq,args=1]
9     \notation{sq}[op=\cdot^2]{\#1^\text{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, that generates a `semantic macro` as well). Whenever we import the containing `module`, we now have an additional field `sq` in (any extension of) `magma` – e.g., as `semigroup` extends `magma`, the following is now valid:

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

...producing a^2 .

3.3.3 Nesting Structures and `\this`

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

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

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

So a `homomorphism` is a `function` with certain properties. And `structures` can be used to “bundle” the `function` itself with both the `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` distinct from the name of its `semantic macro`. We then add two fields that `return` `magmas`. So far, so unexciting:

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

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

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

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

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

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

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

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

Example 14

Input:

```
File [sTeX/MathTutorial]algebra/Homomorphism.en.tex
9 \begin{mathstructure}{magmahom}[magma homomorphism,this=F]
10   \symdef{dom}[name=domain,return={\magma[comp={##1}_1]}]{M_1}
11   \symdef{cod}[name=codomain,return={\magma[comp={##1}_2]}]{M_2}
12   \symdef{f}[op=\this,args=1,
13     type=\funspace{\dom{univ}}{\cod{univ}}
14   ]{\this \dobrackets{#1}}
15 
16 \begin{sdefinition}[for={magmahom,dom,cod,f}]
17   \vardef{va}[name=a,type=\dom{univ}]{a}
18   \vardef{vb}[name=b,type=\dom{univ}]{b}
19   Let $\dom!=\dom{}$ and $\cod!=\cod{}$ \sns{magma}.
20   A \definename{magmahom} is a function
21   $f!\{\dom{}\}\{\cod{}\}$ 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 \vee absorbs \wedge and vice versa; i.e. $a \vee (a \wedge b) = a$ and $a \wedge (a \vee b) = a$.

The operations \wedge and \vee are called **meet** and **join**, respectively.

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

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

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

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

In our case, we want two copies of `semilattice`, which we will call `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 $\text{\texttt{STEX}}$. 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 $\text{\texttt{STEX}}$ ecosystem, we will cover them here. In fact, educational documents have been one of the initial document categories $\text{\texttt{STEX}}$ has been developed for. The idea is that if we can mark up the meaning and didactic role of learning objects, we can base learning support services on that and embed them into the documents.

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

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

4.1 Slides and Course Notes

TODO⁶

Contents

4.2 Problems and Exercises

4.2.1 Background

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

⁶TODO: `notesslides.sty`

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

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

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

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

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

4.2.2 The Package, Options, and Configuration

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

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

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

The `problem` package generates content and labels according to the language specified in the `lang` key of the main document¹, these can be localized by in the files `1df/problem-<lang>.1df`⁷.

¹EdNOTE: MK: document the attribute somewhere

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

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

4.2.3 (Open) Problems

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

Example 16

Input:

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

Output:

Exercise (Fitting Elefants)

How many Elefants can you fit into a Volkswagen beetle?

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

The `sproblem` environment takes an optional key/value argument with the keys `id` as an identifier that can be reference later, `pts` for the points to be gained from this exercise in homework or quiz situations, `min` for the estimated minutes needed to solve the problem, and finally `title` for an informative title of the problem.

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

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

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

4.2.4 Structured Problems

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

Example 17

Input:

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

Output:

Exercise (Fitting Elefants Modularly)

Consider the problem of fitting elephants into a Volkswagen beetle.

1. Estimate the number of elephants on the front seats.

Lösung: Two on each side one.

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

Lösung: Three little elephants.

3. What is the total number?

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

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

4.2.5 Single/Multiple Choice Blocks

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

`\mcc` `\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 correct answers, `F` (the default value) for false ones,
- `Ttext` the verdict for correct answers, `Ftext` for false ones, and
- `feedback` for a short feedback text given to the student.

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

Example 18

Input:

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

Output:

Exercise (Functions)

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

- def^a
- function^b
- fun^c
- public static void^d

^aKorrekt

^bFalsch

that is for C and C++

^cFalsch

that is for Standard ML

^dNooooooooo

that is for Java

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

Example 19

Input:

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

Output:

Exercise (Functions)

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

- def
- function
- fun
- public static void

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

Example 20

Input:

```
1 \startproblems
2 \begin{sproblem}[title=Functions,name=functions1]
3 What is the keyword to introduce a function definition in python?
4
5 \begin{mcb}[style=inline]
6   \mcc[T]{def} \mcc[F,feedback=that is for C and C++]{function}
7   \mcc[F,feedback=that is for Standard ML]{fun}
8   \mcc[F,Ftext=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^a
- function^b
- fun^c
- public static void^d

^aKorrekt

^bFalsch

that is for C and C++

^cFalsch

that is for Standard ML

^dNoooooooooooo

that is for Java

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

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

4.2.6 Filling-In Concrete Solutions

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

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

Example 21

Input:

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

Output:

Exercise (Fitting Elefants)

How many Elefants can you fit into a Volkswagen beetle?

and the actual solution in solutions mode:

Example 22

Input:

²EDNOTE: MK: are there any differences between mcc and scc we need to discuss here?

```

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

```

Output:

Exercise (Fitting Elefants)

How many Elefants can you fit into a Volkswagen beetle?

4

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

Obviously, the required argument of `\fillinsol` can be used for auto-grading. For concrete data like numbers, this is immediate, for more complex data like strings “soft comparisons” might be in order.⁸

4.2.7 Answer Classes

³.

4.2.8 Including Problems

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

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

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

4.2.9 Testing and Spacing

The `problem` package is often used by the `hwexam` package, which is used to create homework assignments and exams. Both of these have a “test mode” (invoked by the

⁸For 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.

³EDNOTE: MK: describe them

package option `test`), where certain information –master solutions or feedback – is not shown in the presentation.

`\testspace` `\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: `\testsmallspace`, `\testmedspace`, `\testlargepage` 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` The `solution` environment takes an `TODO`⁹
`solution`

4.3 Homework Assignments and Exams

4.3.1 Introduction

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the `problem` package. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

4.3.2 Package Options

`solutions` The `hwexam` package and class take the options `solutions`, `notes`, `hints`, `gnotes`, `pts`, `notes` `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation hints for a description of the intended behavior).
`gnotes`
`pts`
`min`

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

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

4.3.3 Assignments

`assignment` (*env.*) This package supplies the `assignment` environment that groups problems into assignment `number` sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents `title` — the ordinal of the `assignment` environment), `title` (for the assignment title; this is `type` referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. “quiz”, `given` or “homework”), `given` (for the date the assignment was given), and `due` (for the date `due` the assignment is due).

⁹TODO: check what is still undescribed `problem.sty` and make examples for it.

4.3.4 Including Assignments

\inputassignment

The `\inputassignment` macro can be used to input an assignment from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one `assignment` environment in the included file). The keys `number`, `title`, `type`, `given`, and `due` are just as for the `assignment` environment and (if given) overwrite the ones specified in the `assignment` environment in the included file.

4.3.5 Typesetting Exams

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

```
reqpts1 \title{320101 General Computer Science (Fall 2010)}
2 \begin{testheading}[duration=one hour,min=60,reqpts=27]
3   Good luck to all students!
4 \end{testheading}
```

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2023-12-08

You have one hour (sharp) for the test;

Write the solutions to the sheet.

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

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

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

	To be used for grading, do not write here								
prob.	4.1	1.1	1.2	2.1	2.2	2.3	2.4	2.5	2.6
total	0	0	0	10	0	0	0	0	0
reached									

good luck

10

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

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_TE_X** will load the `babel` package with the provided languages. Supported languages (currently) are:

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

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

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

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

This offers two advantages:

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

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

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

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

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

5.2 Math Archives and the MathHub Directory

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

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

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

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

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

5.2.1 The Structure of Math Archives

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

Each such `archive` needs two subdirectories:

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

An additional `lib`-directory is optional, and is discussed in [section 5.3](#).

5.2.2 MANIFEST.MF-Files

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

```
id: sTeX/Documentation
ns: http://mathhub.info/sTeX/Documentation
narration-base: http://mathhub.info/sTeX/Documentation
format: stex
title: The sTeX Documentation
teaser: The full Documentation for the sTeX system
url-base: https://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^ITeX 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`

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

`\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`
`\cmhgraphics`

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`
`\clstinputmhlisting`

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*.¹¹

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` `TODO12`.

---

`\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` `\symref{<symbol>}{<text>}`

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

---

<sup>12</sup>TODO: experimental

This is as good a place as any other to explain how **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 23

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

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

1. `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** `\dobracket{<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** `TODO13`

#### 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`.

M → Applications of `semantic macros` with arguments are translated to `MMT/OMDoc` as `OMA-terms` with head `<OMS name="⟨symbol⟩"/>`, or `<OMBIND name="⟨symbol⟩"/>`,  
 M → 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{\<name>}[\<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`.

M → `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{\<name>}[\<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  $\langle name \rangle$  (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]):

### 7.7.1 Semantic Macros for Structures

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

#### Example 24

```
\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{\dots}{fieldname}`).
  - `\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^{\text{Foo}}]{S}{}{}` produces  $\langle a^{\text{Foo}}, b^{\text{Foo}}, c^{\text{Foo}} \rangle$ , and `\struct[comp=\#1^{\text{\this}}]{S}{fieldb}{x}{y}` yields  $x b_S^{\text{y}}$ .
- `\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, the `for=` attribute specifies the concept this is an example for. For counterexamples use `style=counterexample`

**sparagraph** (*env.*)

The **sparagraph environment** represents all other kinds of (logical) paragraphs, such as remarks, comments, transitions between topics, recaps, reminders, etc.

All of these take the same arguments:

- `for=(csl)`: a comma-separated list of *symbols*.
- The same optional arguments as `\symdecl`, with `macro=` replacing the name of the *semantic macro*. All of them are only relevant, if either `name=` or `macro=` are provided.

As with `\symdecl`, if no `name` is given, but `macro` is, then the same name is used for both the *semantic macro* and the *symbol* itself.

If `name` is given but `macro` isn't, no *semantic macro* is generated. Subsequently, the newly generated *symbol* is added the `for`-list.

- `style`: see [chapter 9](#).
- `title`: a title to use in various styles (see [chapter 9](#)).
- `id`: a label to use for `\sref`.

\inlineass The macros `\inlineass`, `\inlinedef` and `\inlineex` behave like the `sassertion`, `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 `\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>14</sup>

---

<sup>14</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 (marked with `\comp` or `\maincomp`) in `notations`.

---

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

---

`\varemph` Like `\compemph` and `\compemph@uri`, but governs the highlighting of components (marked 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           TODO15
```

```
\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.
```

---

```
TODO16
```

```
\stexstyleusemodule
\stexstyleimportmodule
\stexstylerequiremodule
\stexstyleassign
\stexstylerenamedecl
\stexstyleassignMorphism
\stexstylecopymod
\stexstyleinterpretmod
\stexstylerealize
\stexstylesymdecl
\stexstyleextsymdecl
\stexstylenotation
\stexstylevarnotation
\stexstylesymdef
\stexstylevardef
\stexstylevarseq
\stexstylepsfesketch
\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.



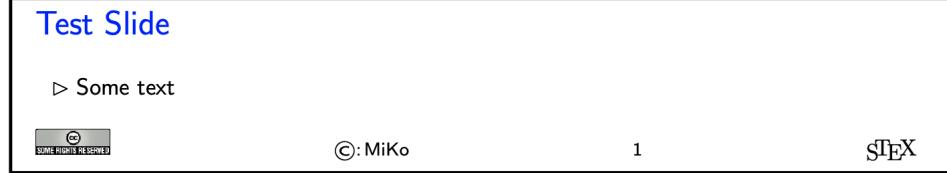
The **beamer** class recommends not to use the **allowframebreaks** option on frames (even though it is very convenient). This holds even more in the **notesslides** case: At least in conjunction with **\newpage**, frame numbering behaves funny (we have tried to fix this, but who knows).

`\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{<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` `\printexcursion` `\excursionref` 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. 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 25

Input:

```

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

```

Output:

### Exercise (Fitting Elefants)

How many Elefants can you fit into a Volkswagen beetle?

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

**solution (env.)** The **solution** environment can be used to specify a solution to a problem. If the package option **solutions** is set or **\solutionstrue** is set in the text, then the solution will be presented in the output. The **solution** environment takes an optional KeyVal argument with the keys **id** for an identifier that can be 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 26

Input:

```
1 \startSolutions
2 \begin{sproblem}[title=Functions,name=functions1]
3 What is the keyword to introduce a function definition in python?
4 \begin{mcb}
5   \mcc[T]{def}
6   \mcc[F,feedback=that is for C and C++]{function}
7   \mcc[F,feedback=that is for Standard ML]{fun}
8   \mcc[F,Ftext=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<sup>a</sup>
- function<sup>b</sup>
- fun<sup>c</sup>
- public static void<sup>d</sup>

<sup>a</sup>Korrekt

<sup>b</sup>Falsch

*that is for C and C++*

<sup>c</sup>Falsch

*that is for Standard ML*

<sup>d</sup>Noooooooooo

*that is for Java*

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

### Example 27

Input:

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

Output:

### Exercise (Functions)

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

- def
- function
- fun
- public static void

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

### Filling-In Concrete Solutions

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

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

**Example 28**

Input:

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

**Exercise (Fitting Elefants)**  
and the actual solution in solutions mode:

**Example 29** Many Elefants can you fit into a Volkswagen beetle?

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>17</sup>

#### 10.2.4 Including Problems

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

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

<sup>17</sup>For the moment we only assume a single concrete value as correct. In the future we will almost certainly want to extend the functionality to multiple answer classes that allow different feedback like in MCQ. This still needs a bit of design. Also we want to make the formatting of the answer in solutions/test mode configurable.

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

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

### 10.2.5 Testing and Spacing

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

```
\testspace      \testspace takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: \testsmallspace, \testsmallspace, \testsmallspace \testsmallspace give small (1cm), medium (2cm), and big (3cm) vertical space.  
\testsmallspace \testnewpage makes a new page in test mode, and \testemptypage generates an empty page with the cautionary message that this page was intentionally left empty.  
\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).

`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 **type** referenced in the title of the assignment sheet), **type** (for the assignment type; e.g. “quiz”, **given** or “homework”), **given** (for the date the assignment was given), and **due** (for the date **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 of minutes, and **reqpts** the points that are required for a perfect grade.

```
reqpts1 \title{320101 General Computer Science (Fall 2010)}
2 \begin{testheading}[duration=one hour,min=60,reqpts=27]
3 Good luck to all students!
4 \end{testheading}
```

Will result in

Name:

Matriculation Number:

## 320101 General Computer Science (Fall 2010)

2023-12-08

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

Write the solutions to the sheet.

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

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

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

|         | To be used for grading, do not write here |     |     |     |     |     |     |  |     |     |
|---------|-------------------------------------------|-----|-----|-----|-----|-----|-----|--|-----|-----|
| prob.   | 4.1                                       | 1.1 | 1.2 | 2.1 | 2.2 | 2.3 | 2.4 |  | 2.5 | 2.6 |
| total   | 0                                         | 0   | 0   | 10  | 0   | 0   | 0   |  | 0   | 0   |
| reached |                                           |     |     |     |     |     |     |  |     |     |

good luck

18

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

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

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

```
1 \documentclass{standalone}
2 \usepackage{tikz}
```

<sup>18</sup>MK: The first three “problems” come from the stex examples above, how do we get rid of this?

```

3 \usetikzpackage{...}
4 \begin{document}
5   \begin{tikzpicture}
6     ...
7   \end{tikzpicture}
8 \end{document}

```

The `standalone` class is a minimal L<sup>A</sup>T<sub>E</sub>X class that when loaded in a document that uses the `standalone` package: the preamble and the `document` environment are disregarded during loading, so they do not pose any problems. In effect, an `\input` of the file above only sees the `tikzpicture` environment, but the file itself is standalone in the sense that we can run L<sup>A</sup>T<sub>E</sub>X over it separately, e.g. for generating an image file from it.

---

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

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

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

---

**`\mhtikzinput`** `\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

## sT<sub>E</sub>X 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 e-type argument (inside an @\_cmd:nomodule=TeX,replace=false,\expanded), x-type argument (in plain TeX terms, inside an @\_cmd:nomodule=TeX,replace=false,\edef), as well as within an f-type argument. ☆ indicates restricted expandable functions, which can be used within an x-type argument or an e-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:nnnn \stex_set_notation_macro:nnnn
\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 30

Assume we have a symbol and notation:

```
1 \symdecl{someSymbol}[args=iai]
2 \notation{someSymbol}[prec=10;20x30x40,variant=foo]
3 {First: #1; Second: #2; Third: #3; End}
4 {(#1 -- ##1 split ##2 -- #3)}
```

Since the symbol corresponds to an OMA, the whole notation is wrapped in `\STEXInternalTermMathOMAiii`, taking as arguments the variant identifier (foo), the operator precedence (10) and the body of the notation.

The second argument in the notation, being associative, is wrapped in a `\STEXInternalTermMathAssocArgiiii`, taking as arguments the argument number (2), the precedence (30), the T<sub>E</sub>X parameter token (#2) the notation body ((#1 -- ##1 split ##2 -- #3)), and finally the argument mode (a). Additionally, the markers ##1 and ##2 are replaced by `\STEXInternalAssocArgMarkerI` and `\STEXInternalAssocArgMarkerII`, respectively.

Subsequently, the non-sequence parameter tokens are wrapped in `\STEXInternalTermMathArgiii` with arguments mj (where m is the mode 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

---

```
\_stex_invoke_symbol:nnnnnnN      \l_stex_current_symbol_str  
\l_stex_current_arity_str  
\l_stex_current_args_tl  
\l_stex_current_return_tl  
\l_stex_current_type_tl
```

---

```
\_stex_invoke_symbol:nnnnnnN  
{\module}{\name}{\arity}{\args}{\definiens}  
{\type}  
{\return_code}  
{\invocation_macro}
```

is how a semantic macro is/should be defined. `\_stex_invoke_symbol:nnnnnnN` first checks whether semantic macros are currently allowed, and throws an error if not. Otherwise, it sets the `\comp`-controlled highlighting to `\compemph@uri`, initializes `\STEXInternalSymbolAfterInvocationTL`, defines the following macros for all of its arguments, and subsequently calls the `\invocation_macro`:

- `\l_stex_current_symbol_str ={\<module>}{\<name>}`
- `\l_stex_current_arity_str ={\<arity>}`
- `\l_stex_current_args_tl ={\<args>}`
- `\l_stex_current_type_tl ={\<type>}`
- `\l_stex_current_return_tl ={\<return code>}`

The simplest example for an `\invocation_macro` is `\stex_invoke_symbol`:

---

```
\_stex_invoke_variable:nnnnnnN
```

analogous to `\_stex_invoke_symbol:nnnnnnN`, 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 ={\<return>}`.
    - \* If  $\langle return \rangle$  is empty, call the notation macro,
    - \* otherwise, call  $\langle return \rangle \{ \stex_invoke_symbol ! \}$ .
- 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** {\i{id}}{\i{CSL}} first executes *setup code* (e.g. to empty the macros holding the values) and then sets the keys in set *id* with the values provided in *CSL*.

---

**\\_stex\_do\_id:** should be called whenever a macro or environment has a label id, i.e. calls **\stex\_keys\_set:nn**{*id*}{{...}}, after the title has been typeset. Sets a **\label** by calling **\stex\_ref\_new\_doc\_target:n**{*id*}.

### Example 31

If we define a set of keys with:

```
1 \stex_keys_define:nnnn{archive file}{  
2   \str_clear:N \l_stex_key_archive_str  
3   \str_clear:N \l_stex_key_file_str  
4 }{  
5   archive .str_set_x:N = \l_stex_key_archive_str ,  
6   file .str_set_x:N = \l_stex_key_file_str  
7 }{id}
```

then calling **\stex\_keys\_set:nn**{*archive file*}{{*id=foo,file=bar*}} sets **\l\_stex\_key\_file\_str={bar}**, assures that **\l\_stex\_key\_archive\_str** is empty, and executes the code associated with *id*, i.e. it sets **\l\_stex\_key\_id\_str={foo}**.

## 11.9 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 32

$\text{\importmodule}$  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 33

$\text{\smodule}$  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`).

`\ifstexhtml` is set accordingly.

---

```
\stex_annotationnn
```

```
\stex_annotationnn {⟨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\_innn.

`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 34

```
1 \stex_pseudogroup:nn{  
2   something changing \foo  
3   something changing \num  
4 }{  
5   \stex_pseudogroup_restore:N\foo  
6   \int_set:Nn \num {\int_use:N \num}  
7 }
```

restores the values of macro `\foo` and register `\num` after calling the first block.

Commands like `\symdecl` and `\importmodule` that generate (semantic) macros should be local *to the current module*, e.g. `smodule`. For that purpose, we open a new “metagroup” with some identifier (e.g. `\l_stex_current_module_str`) and then execute the relevant code *in the metagroup with that identifier*:

---

```
\stex_metagroup_new:n  
\stex_metagroup_new: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}{2023/10/13}{3.4.0}{sTeX document class}
6  \IfFileExists{stex-expl-compat.sty}{
7      \usepackage{stex-expl-compat}
8  }{}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14
15 \LoadClass{article}
16 </cls>

    Setup code for the package

17 <*package>
18 \RequirePackage{expl3, l3keys2e, ltxcmds}
19 \ProvidesExplPackage{stex}{2023/10/13}{3.4.0}{sTeX package}
20 \IfFileExists{stex-expl-compat.sty}{
21     \usepackage{stex-expl-compat}
22 }{}
23 \RequirePackage{stex-logo} % externalized for backwards-compatibility reasons
24 \RequirePackage{standalone}
25
26 \message{^^J*~This~is~sTeX~version~3.4.0~*^^J}

    Package options:

27 \keys_define:nn { stex / package } {
28     debug      .str_set_x:N = \c_stex_debug_clist ,
29     lang       .clist_set:N = \c_stex_languages_clist ,
30     mathhub    .tl_set_x:N = \mathhub ,
```

```

31   usesms     .bool_set:N = \c_stex_persist_mode_bool ,
32   writesms   .bool_set:N = \c_stex_persist_write_mode_bool ,
33   checkterms .bool_set:N = \c_stex_check_terms_bool ,
34   image      .bool_set:N = \c_tikzinput_image_bool,
35   nofrontmatter .bool_set:N = \c_stex_no_frontmatter_bool,
36   unknown     .code:n    = {}
37 }
38 \exp_args:NNo \clist_set:Nn \c_stex_debug_clist \c_stex_debug_clist
39 \ProcessKeysOptions { stex / package }

Error messages:
40 \input{stex-en.ldf}

```

## 13.2 Utilities

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

### 13.2.1 Calling kpsewhich and Environment Variables

42 <@=stex\_envs>

\stex\_kpsewhich:Nn

```

43 \cs_new_protected:Nn \stex_kpsewhich:Nn {\group_begin:
44   \catcode`\|=12
45   \sys_get_shell:nnN { kpsewhich ~ #2 } { } \l_tmpa_tl
46   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
47   \group_end:
48   \exp_args:NNo\str_set:Nn #1 {\l_tmpa_tl}
49   \tl_trim_spaces:N #1
50 }

```

(End of definition for \stex\_kpsewhich:Nn. This function is documented on page 142.)

\stex\_get\_env:Nn

```

51 \sys_if_platform_windows:T {
52   \cs_new_protected:Nn \stex_get_env:Nn {\group_begin:
53     \escapechar=-1\catcode`\\=12
54     \exp_args:NN \stex_kpsewhich:Nn #1 {-expand-var~\c_percent_str#2\c_percent_str}
55     \exp_args:NNx\use:nn\group_end:{\str_set:Nn \exp_not:N #1 { #1 }
56     }
57   }
58 }
59 }{
60   \cs_new_protected:Nn \stex_get_env:Nn {
61     \stex_kpsewhich:Nn #1 {-var-value~#2}
62   }
63 }

```

(End of definition for \stex\_get\_env:Nn. This function is documented on page 142.)

### 13.2.2 Logging

```
64  <@=stex_debug>
65  \cs_new_protected:Nn \stex_debug:nn {
66      \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist { \tl_to_str:n{all} }{
67          \__stex_debug_:nn{#1}{#2}
68      }{
69          \exp_args:NNo \clist_if_in:NnT \c_stex_debug_clist { \tl_to_str:n{#1} }{
70              \__stex_debug_:nn{#1}{#2}
71          }
72      }
73  }
74
75 \cs_new_protected:Nn \__stex_debug_:nn {
76     \msg_set:nnn{stex}{debug / #1}{
77         \\Debug~#1:~#2\\
78     }
79     \msg_none:nn{stex}{debug / #1}
80 }
```

(End of definition for `\stex_debug:nn`. This function is documented on page 120.)

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 }{ warning }
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 }{ warning }
99         \stex_debug:nn{#1}{Logging~#1}
100    }
101 }
```

### 13.2.3 File Paths

```
102 <@=stex_path>
```

```
\c_stex_filepath_sep_str
```

```
103 \sys_if_platform_windows:TF{
104     \str_set_eq:NN \c_stex_filepath_sep_str \c_backslash_str
105 }{
106     \str_const:Nn \c_stex_filepath_sep_str {}
```

107 }

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

### 13.2.4 Languages

108 <@=stex\_lang>

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

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

```
109 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
110   en = english ,
111   de = ngerman ,
112   ar = arabic ,
113   bg = bulgarian ,
114   ru = russian ,
115   fi = finnish ,
116   ro = romanian ,
117   tr = turkish ,
118   fr = french
119 } }
120 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
121   english = en ,
122   ngerman = de ,
123   arabic = ar ,
124   bulgarian = bg ,
125   russian = ru ,
126   finnish = fi ,
127   romanian = ro ,
128   turkish = tr ,
129   french = fr
130 } }
131 % todo: chinese simplified (zhs)
132 %         chinese traditional (zht)
```

(End of definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 139.)

`\l_stex_current_language_str`

134 \str\_new:N \l\_stex\_current\_language\_str

(End of definition for `\l_stex_current_language_str`. This variable is documented on page 139.)

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

`\stex_set_language:n`

`\stex_set_language:x`

`\stex_set_language:o`

```
135 \cs_new_protected:Nn \stex_set_language:n {
136   \str_set:Nn \l_stex_current_language_str { #1 }
137   \prop_if_in:NnTF \c_stex_languages_prop {#1} {
138     \tl_set_rescan:Nnx \l__stex_lang_lang_str {}{\prop_item:Nn \c_stex_languages_prop {#1}}
139     \cs_if_eq:NNTF \onlypreamble \notprerr {
140       \ltx@ifpackageloaded{babel} {
141         \exp_args:No \selectlanguage \l__stex_lang_lang_str
142       } {}
143     } {
144       \ltx@ifpackageloaded{babel} {} {
```

```

145      \str_if_eq:nnTF {#1} {tr} {
146          \RequirePackage[turkish,shorthands=:!]{babel}
147      }{
148          \RequirePackage[\l__stex_lang_lang_str]{babel}
149      }
150  }
151 }
152 }{
153     \msg_error:nnx{stex}{error/unknownlanguage}{#1}
154 }
155 }
156 \cs_generate_variant:Nn \stex_set_language:n {x,o}

```

(End of definition for `\stex_set_language:n`. This function is documented on page 140.)

### `\stex_language_from_file:`

```

157 \cs_new_protected:Nn \stex_language_from_file: {
158     \seq_get_right:NN \g_stex_current_file \l_tmpa_str
159     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
160     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str \% = ".tex/.dtx/.sty"
161     \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
162         \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
163             \exp_args:No \str_if_eq:nnF \l_tmpa_str {ltx} {
164                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
165             }
166         }
167     }
168     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str \% <filename>
169     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
170         \seq_pop_right:NN \l_tmpa_seq \l__stex_lang_str
171         \str_if_eq:NNF \l__stex_lang_str \l_stex_current_language_str {
172             \exp_args:NNo \prop_if_in:NnT \c_stex_languages_prop \l__stex_lang_str {
173                 \stex_set_language:o \l__stex_lang_str
174             }
175         }
176         \stex_debug:nn{lang} {Language~\l_stex_current_language_str~
177             inferred~from~file~name}
178     }
179 }

```

(End of definition for `\stex_language_from_file:`. This function is documented on page 140.)

Loading babel:

```

180 \clist_if_empty:NF \c_stex_languages_clist {
181     \bool_set_false:N \l__stex_lang_turkish_bool
182     \seq_clear:N \l_tmpa_seq
183     \clist_map_inline:Nn \c_stex_languages_clist {
184         \str_set:Nx \l_tmpa_str {#1}
185         \str_if_eq:nnT {#1}{tr} {
186             \bool_set_true:N \l__stex_lang_turkish_bool
187         }
188         \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
189             \tl_set_rescan:Nno \l_tmpa_str {} \l_tmpa_str
190             \seq_put_right:No \l_tmpa_seq \l_tmpa_str
191     }

```

```

192      \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
193    }
194  }
195 \stex_debug:nn{lang} {Languages:~\seq_use:Nn \l_tmpa_seq {,~} }
196 \bool_if:NTF \l_stex_lang_turkish_bool {
197   \exp_args:NNe \use:nn \RequirePackage
198   {[main=\seq_use:Nn \l_tmpa_seq, ,shorthands=:!]}{babel}
199 }{
200   \exp_args:NNe \use:nn \RequirePackage
201   {[main=\seq_use:Nn \l_tmpa_seq, ]}{babel}
202 }
203 }
```

### 13.2.5 Group-like Behaviours

204 `<@@=stex_groups>`

`\stex_pseudogroup:nn`

`\stex_pseudogroup_restore:N`

```

205 \cs_new_protected:Npn \stex_pseudogroup:nn {
206   \exp_args:Nne \use:nn
207 }
208 \cs_new:Nn \stex_pseudogroup_restore:N {
209   \tl_if_exist:NTF #1 {
210     \tl_set:Nn \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
211   }{
212     \cs_undefine:N \exp_not:N #1
213   }
214 }
```

(End of definition for `\stex_pseudogroup:nn` and `\stex_pseudogroup_restore:N`. These functions are documented on page 142.)

`\stex_pseudogroup_with:nn`

```

215 \cs_new_protected:Nn \stex_pseudogroup_with:nn {
216   \tl_map_inline:nn{#1} {
217     \cs_set_eq:cN{\_stex_groups_\tl_to_str:n{##1}}##1
218   }
219 #2
220 \tl_map_inline:nn{#1} {
221   \cs_set_eq:Nc##1{\_stex_groups_\tl_to_str:n{##1}}
222   \cs_undefine:c\#1{\_stex_groups_\tl_to_str:n{##1}}
223 }
224 }
```

(End of definition for `\stex_pseudogroup_with:nn`. This function is documented on page 142.)

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

`\stex_metagroup_new:o` `\seq_new:N \l_stex_groups_ids_seq`

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

```

226 \cs_new_protected:Nn \stex_metagroup_new:n {
227   \str_set:cx{\l_stex_groups_#1_int} {\int_use:N\currentgrouplevel}
228   \seq_put_right:Nn \l_stex_groups_ids_seq {#1}
229 }
230 \cs_generate_variant:Nn \stex_metagroup_new:n {o}
```

(End of definition for `\stex_metagroup_new:n`. This function is documented on page 143.)

```
\stex_metagroup_do_in:nn
\stex_metagroup_do_in:nx
231 \prg_new_conditional:Nnn \__stex_groups_exists:n {TF} {
232     \str_if_exist:cTF{l__stex_groups_#1_int}
233         \prg_return_true: \prg_return_false:
234 }
235
236 \cs_new_protected:Nn \stex_metagroup_do_in:nn {
237     \__stex_groups_exists:nTF{#1} {
238         \__stex_groups_do_in:nn{#1}{#2}
239     }{
240         \msg_error:nnn{stex}{error/metagroup/missing}{#1}
241     }
242 }
243 \cs_generate_variant:Nn \stex_metagroup_do_in:nn {nx}
244
245 \cs_new_protected:Nn \__stex_groups_do_in:nn {
246     \exp_args:Nnx\stex_debug:nn{metagroup}{adding-to-\detokenize{#1}:^J\tl_to_str:n{#2}}
247     \tl_set:Nn\__stex_groups_tmp{#2}
248     \exp_args:Nx \int_compare:nNnF {\use:c{l__stex_groups_#1_int}}
249         = \currentgrouplevel {
250             \tl_if_exist:cTF{g__stex_groups_#1_\the\currentgrouplevel _content} {
251                 \exp_args:Nno \tl_gput_right:cn{g__stex_groups_#1_\the\currentgrouplevel _content}
252             }{
253                 \exp_args:Nno \tl_gset:cn{g__stex_groups_#1_\the\currentgrouplevel _content}
254             }\__stex_groups_tmp
255             \bool_if_exist:cF {l__stex_groups_\the\currentgrouplevel _bool} {
256                 \group_insert_after:N \__stex_groups_do:
257                 \bool_set_true:c {l__stex_groups_\the\currentgrouplevel _bool}
258             }
259         }
260         \__stex_groups_tmp
261     }
262
263 \cs_new_protected:Nn \__stex_groups_do: {
264     \seq_map_inline:Nn \l__stex_groups_ids_seq {
265         \tl_if_exist:cT{g__stex_groups_#1_\int_eval:n{\currentgrouplevel+1}_content} {
266             \exp_args:NNno \exp_args:Nno \__stex_groups_do_in:nn{##1} {
267                 \csname g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content\endcsname
268             }
269             \cs_undefine:c{g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content}
270         }
271         \bool_if_exist:cF {l__stex_groups_\int_eval:n\currentgrouplevel _bool} {
272             \group_insert_after:N \__stex_groups_do:
273             \bool_set_true:c {l__stex_groups_\int_eval:n\currentgrouplevel _bool}
274         }
275     }
276 }
```

(End of definition for `\stex_metagroup_do_in:nn`. This function is documented on page 143.)

### 13.2.6 HTML Annotations

```

277 <@=stex_annotation>
\stex_if_do_html:TF Whether to (locally) produce HTML output
278 \bool_new:N \_stex_html_do_output_bool
279 \bool_set_true:N \_stex_html_do_output_bool
280
281 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
282     \bool_if:nTF \_stex_html_do_output_bool
283         \prg_return_true: \prg_return_false:
284 }

```

(End of definition for `\stex_if_do_html:TF`. This function is documented on page 140.)

`\stex_suppress_html:n` Temporarily disable HTML output

```

285 \cs_new_protected:Nn \stex_suppress_html:n {
286     \stex_pseudogroup:n{
287         \bool_set_false:N \_stex_html_do_output_bool
288         #1
289     }{
290         \stex_if_do_html:T {
291             \bool_set_true:N \_stex_html_do_output_bool
292         }
293     }
294 }

```

(End of definition for `\stex_suppress_html:n`. This function is documented on page 140.)

We determine the backend:

```

\stex_if_html_backend_p:
\stex_if_html_backend:TF
    \ifstexhtml
        \stex@backend
    \fi
299
300 \stex_get_env:Nn\__stex_annotation_env_str{STEX_FORCE_PDF}
301 \exp_args:No \str_if_eq:nnTF \__stex_annotation_env_str {true} {
302     \def\stex@backend{pdflatex}
303 }{
304     \tl_if_exist:NF\stex@backend{
305         \if@rustex
306             \def\stex@backend{rustex}
307         \else
308             \cs_if_exist:NTF\HCodef{
309                 \def\stex@backend{tex4ht}
310             }{
311                 \def\stex@backend{pdflatex}
312             }
313         \fi
314     }
315 }
316 \input{stex-backend-\stex@backend.cfg}
317 \newif\ifstexhtml
318 \stex_if_html_backend:TF {

```

```

321   \stexhtmltrue
322   \bool_set_true:N \_stex_html_do_output_bool
323 }{
324   \stexhtmlfalse
325   \bool_set_false:N \_stex_html_do_output_bool
326 }

```

(End of definition for `\stex_if_html_backend:TF`, `\ifstexhtml`, and `\stex@backend`. These functions are documented on page 140.)

```

\_stex_annotation_force_break:n
327 \stex_if_html_backend:TF {
328   \cs_new_protected:Nn \_stex_annotation_force_break:n {
329     \stex_annotation_invisible:n{~}
330     #1
331     \stex_annotation_invisible:n{~}
332   }
333 }{
334   \cs_new_protected:Nn \_stex_annotation_force_break:n { #1 }
335 }

```

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

```

\mmlintent
\mmlarg
336 \stex_if_html_backend:TF {
337   \cs_new_protected:Npn \mmlintent #1 #2 {
338     \stex_annotation_nn:mml:intent={#1}{#2}
339   }
340   \cs_new_protected:Npn \mmlarg #1 #2 {
341     \stex_annotation_nn:mml:arg={#1}{#2}
342   }
343 }{
344   \cs_new_protected:Npn \mmlintent #1 #2 { #2 }
345   \cs_new_protected:Npn \mmlarg #1 #2 { #2 }
346 }

```

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

### 13.2.7 Auxiliary Methods

```
347 <@@=stex_aux>
```

```

\stex_deactivate_macro:Nn
\stex_reactivate_macro:N
348 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
349   \tl_set_eq:cN{\tl_to_str:n{#1}~~orig}#1
350   \cs_set_protected:Npn#1{
351     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
352   }
353 }
354 \cs_new_protected:Nn \stex_reactivate_macro:N {
355   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1}~~orig\endcsname
356 }

```

(End of definition for `\stex_deactivate_macro:Nn` and `\stex_reactivate_macro:N`. These functions are documented on page 142.)

```

\stex_ignore_spaces_and_pars:
357 \protected\def\stex_ignore_spaces_and_pars:{%
358   \begingroup\catcode13=10\relax
359   \@ifnextchar\par{%
360     \endgroup\expandafter\stex_ignore_spaces_and_pars:\@gobble
361   }{%
362     \endgroup
363   }
364 }

```

(End of definition for `\stex_ignore_spaces_and_pars`. This function is documented on page 142.)

`\stex_keys_define:nnnn`

```

365 \cs_new_nopar:Nn \stex_keys_define:nnnn {
366   \tl_gset:cn {__stex_aux_keys_#1_pre_tl}{#2}
367   \tl_gset:cn {__stex_aux_keys_#1_def_tl}{#3}
368   \tl_if_empty:nF{#4}{%
369     \clist_map_inline:nn{#4}{%
370       \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_pre_tl}
371       \tl_gput_left:co{__stex_aux_keys_#1_pre_tl} \l__stex_aux_tl
372       \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_def_tl}
373       \tl_gput_left:cn{__stex_aux_keys_#1_def_tl} ,
374       \tl_gput_left:co{__stex_aux_keys_#1_def_tl} \l__stex_aux_tl
375     }
376   }
377   \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_#1_def_tl}
378   \exp_args:Nno \keys_define:nn {stex / #1} {\l__stex_aux_tl}
379 }

```

(End of definition for `\stex_keys_define:nnnn`. This function is documented on page 132.)

`\stex_keys_set:nn`

```

380 \cs_new_nopar:Nn \stex_keys_set:nn {
381   \use:c{\__stex_aux_keys_#1_pre_tl}
382   \keys_set:nn {stex / #1} { #2 }
383 }

```

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

Some ubiquitous key sets:

```

384 \stex_keys_define:nnnn{archive file}{%
385   \str_clear:N \l_stex_key_archive_str
386   \str_clear:N \l_stex_key_file_str
387 }{%
388   archive .str_set_x:N = \l_stex_key_archive_str ,
389   file .str_set_x:N = \l_stex_key_file_str
390 }{}%
391 \stex_keys_define:nnnn{id}{%
392   \str_clear:N \l_stex_key_id_str
393 }{%
394   id .str_set_x:N = \l_stex_key_id_str
395 }{}%
396 \stex_keys_define:nnnn{title}{%

```

```

399   \tl_clear:N \l_stex_key_title_tl
400 }{
401   title .tl_set:N = \l_stex_key_title_tl
402 }{}
403
404 \stex_keys_define:nnnn{style}{%
405   \clist_clear:N \l_stex_key_style_clist
406 }{
407   style .clist_set:N = \l_stex_key_style_clist
408 }{}
409
410 \stex_keys_define:nnnn{deprecate}{%
411   \str_clear:N \l_stex_key_deprecate_str
412 }{
413   deprecate .str_set_x:N = \l_stex_key_deprecate_str
414 }{}
415
416 \stex_keys_define:nnnn{uses}{%
417   uses .code:n = {
418     \clist_map_inline:nn{#1}{%
419       \stex_if_starts_with:nTF{##1}[%
420         \_stex_aux_split_at_bracket:w ##1 \_stex_end:
421       }{
422         \usemodule{##1}
423       }
424     }
425   }%
426 }{}
427
428 \cs_new_protected:Npn \_stex_aux_split_at_bracket:w [ #1 ] #2 \_stex_end: {%
429   \usemodule[#1]{#2}
430 }

```

**\\_stex\_do\_deprecation:n**

```

431 \cs_new:Nn \_stex_do_deprecation:n {
432   \str_if_empty:NF \l_stex_key_deprecate_str {
433     \msg_warning:nnxx{stex}{warning/deprecated}{#1}{\l_stex_key_deprecate_str}
434   }
435 }

```

(End of definition for `\_stex_do_deprecation:n`. This function is documented on page [120](#).)

**\\_stex\_do\_id:**

```

436 \cs_new_protected:Nn \_stex_do_id: {
437   \stex_if_smsmode:F {
438     \str_if_empty:NF \l_stex_key_id_str {
439       \exp_args:No \stex_ref_new_doc_target:n \l_stex_key_id_str
440     }
441   }
442 }

```

(End of definition for `\_stex_do_id:`. This function is documented on page [132](#).)

```

\stex_new_stytable_env:nmmmmnnn
\stex_new_stytable_cmd:nnnn
\stex_style_apply:
443 \cs_new_protected:Nn \stex_new_stytable_cmd:nnnn {
444   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[2] [] {
445     \__stex_aux_patch:nnn{#1}{##1}{##2}
446   }
447   \exp_after:wN \NewDocumentCommand\cs:w #1\cs_end:{#2} {
448     \cs_set:Npn \stex_style_apply: {
449       \__stex_aux_apply_patch:n{#1}
450     }
451     #3
452   }
453   \tl_set:cn {__stex_aux_style_#1:} { #4 }
454 }
455
456 \cs_new_protected:Nn \__stex_aux_apply_patch:n {
457   \clist_if_empty:NTF \l_stex_key_style_clist {
458     \tl_clear:N \thisstyle
459     \use:c{__stex_aux_style_#1:}
460   }{
461     \clist_get:NN \l_stex_key_style_clist \thisstyle
462     \tl_if_exist:cTF{__stex_aux_style_#1_\thisstyle :} {
463       \use:c{__stex_aux_style_#1_\thisstyle :}
464     }{
465       \use:c{__stex_aux_style_#1:}
466     }
467   }
468 }
469
470 \cs_new_protected:Nn \__stex_aux_patch:nnn {
471   \str_if_empty:nTF {#2} {
472     \tl_set:cn{__stex_aux_style_#1:}{#3}
473   }{
474     \tl_set:cn{__stex_aux_style_#1_#2:}{#3}
475   }
476 }
477
478 \cs_new_protected:Nn \stex_new_stytable_env:nnnnnnnn {
479   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[3] [] {
480     \__stex_aux_patch:nnnn{#1}{##1}{##2}{##3}
481   }
482   \NewDocumentEnvironment{#7#1}{#2} {
483     \cs_set:Npn \stex_style_apply: {
484       \__stex_aux_apply_patch_begin:n{#1}
485     }
486     #3
487   }{
488     \cs_set:Npn \stex_style_apply: {
489       \__stex_aux_apply_patch_end:n{#1}
490     }
491     #4
492   }
493   \tl_set:cn {__stex_aux_style_#1_start:} { #5 }
494   \tl_set:cn {__stex_aux_style_#1_end:} { #6 }
495 }

```

```

496 \cs_new_protected:Nn \__stex_aux_patch:n {
497   \str_if_empty:nTF {#2} {
498     \tl_set:cn{\__stex_aux_style_#1_start:}{#3}
499     \tl_set:cn{\__stex_aux_style_#1_end:}{#4}
500   }{
501     \tl_set:cn{\__stex_aux_style_#1_#2_start:}{#3}
502     \tl_set:cn{\__stex_aux_style_#1_#2_end:}{#4}
503   }
504 }
505 }
506
507 \cs_new_protected:Nn \__stex_aux_apply_patch_begin:n {
508   \clist_if_empty:NTF \l_stex_key_style_clist {
509     \tl_clear:N \thisstyle
510     \use:c{\__stex_aux_style_#1_start:}
511   }{
512     \clist_get:NN \l_stex_key_style_clist \thisstyle
513     \stex_debug:nn{styling}{dominant-style:~\thisstyle}
514     \tl_if_exist:cTF{\__stex_aux_style_#1_\thisstyle_start:}{
515       \use:c{\__stex_aux_style_#1_\thisstyle_start:}
516     }{
517       \use:c{\__stex_aux_style_#1_start:}
518     }
519   }
520 }
521
522 \cs_new_protected:Nn \__stex_aux_apply_patch_end:n {
523   \tl_if_empty:NTF \thisstyle {
524     \use:c{\__stex_aux_style_#1_end:}
525   }{
526     \tl_if_exist:cTF{\__stex_aux_style_#1_\thisstyle_end:}{
527       \use:c{\__stex_aux_style_#1_\thisstyle_end:}
528     }{
529       \use:c{\__stex_aux_style_#1_end:}
530     }
531   }
532 }

```

(End of definition for `\stex_new_stytable_env:nnnnnnn`, `\stex_new_stytable_cmd:nnnn`, and `\stex_style_apply:..`. These functions are documented on page 133.)

`\stex_str_if_ends_with_p:nn`

`\stex_str_if_ends_with:nnTF`

```

533 \prg_new_conditional:Nnn \stex_str_if_ends_with:nn {p,T,F,TF} {
534   \exp_args:Ne \str_if_eq:nnTF {
535     \str_range:nnn{#1}{-}\str_count:n{#2}{-1}
536   }{#2}\prg_return_true: \prg_return_false:
537 }

```

(End of definition for `\stex_str_if_ends_with:nnTF`. This function is documented on page 136.)

`\stex_str_if_starts_with_p:nn`

`\stex_str_if_starts_with:nnTF`

```

538 \prg_new_conditional:Nnn \stex_str_if_starts_with:nn {p,T,F,TF} {
539   \exp_args:Ne \str_if_eq:nnTF {
540     \str_range:nnn{#1}{1}{\str_count:n{#2}}
541   }{#2}\prg_return_true: \prg_return_false:

```

542 }

(End of definition for `\stex_str_if_starts_with:nTF`. This function is documented on page 136.)

### `\stex_macro_body:N`

```
543 \cs_new:Npn \__stex_aux_start:#1\__stex_aux_end: {\exp_not:n{#1}}
544 \cs_new_protected:Nn \__stex_aux_end: {}
545 \cs_new:Nn \stex_macro_body:N {
546     \exp_args:Nne\use:nn{\exp_after:wN \__stex_aux_start: #1} {
547         \__stex_aux_args:e {\cs_parameter_spec:N #1}\__stex_aux_end:
548     }
549 }
550
551 \cs_new:Nn \__stex_aux_args:n {
552     \tl_if_empty:nF{#1} {
553         {##}\exp_args:Ne \tl_head:n {\tl_tail:n {#1}}
554         \__stex_aux_args:e {\exp_args:Ne\tl_tail:n{\tl_tail:n{#1}}}
555     }
556 }
557 \cs_generate_variant:Nn \__stex_aux_args:n {e}
```

(End of definition for `\stex_macro_body:N`. This function is documented on page 141.)

### `\stex_macro_definition:N`

```
558 \cs_new:Nn \stex_macro_definition:N {
559     \__stex_aux_prefix:e {\cs_prefix_spec:N #1}
560     \def\exp_not:N #1
561     \__stex_aux_params:e {\cs_parameter_spec:N #1}
562     {
563         \stex_macro_body:N #1
564     }
565 }
566
567 \cs_new:Nn \__stex_aux_prefix:n {
568     \tl_if_empty:nF{#1} {
569         \str_if_eq:eeTF {
570             \tl_range:nnn{#1}{1}{10}~
571         }{\tl_to_str:n{\protected}}{
572             \protected
573             \__stex_aux_prefix_long:e {
574                 \str_range:nnn{#1}{11}{-1}
575             }
576         }{
577             \__stex_aux_prefix_long:n {#1}
578         }
579     }
580 }
581 \cs_generate_variant:Nn \__stex_aux_prefix:n {e}
582
583 \cs_new:Nn \__stex_aux_prefix_long:n {
584     \tl_if_empty:nF{#1} {
585         \str_if_eq:eeT {
586             \tl_range:nnn{#1}{1}{10}~
587         }{\tl_to_str:n{\long}}{\long}
588     }
589 }
```

```

589 }
590 \cs_generate_variant:Nn \__stex_aux_prefix_long:n {e}
591
592 \cs_new:Nn \__stex_aux_params:n {
593     \tl_if_empty:nF{#1} {
594         \exp_args:NNe \str_if_eq:VnTF \c_hash_str {\tl_head:n{#1}}{
595             #####
596         }{
597             \tl_head:n{#1}
598         }
599         \__stex_aux_params:e {\tl_tail:n{#1}}
600     }
601 }
602 \cs_generate_variant:Nn \__stex_aux_params:n {e}

```

(End of definition for `\stex_macro_definition:N`. This function is documented on page [141](#).)

### 13.2.8 Persistence

```
603 <@=stex_persist>
```

We check the environment variables:

```

604 \stex_get_env:Nn\__stex_persist_env_str{STEX_USESMS}
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_mode_bool
608     }
609 }
610 \stex_get_env:Nn\__stex_persist_env_str{STEX_WRITESMS}
611 \str_if_empty:NF\__stex_persist_env_str{
612     \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false} {
613         \bool_set_true:N \c_stex_persist_write_mode_bool
614     }
615 }

```

```

\stex_persist:n
\stex_persist:e
616 \iow_new:N \c__stex_persist_sms_iow
617
618 \bool_if:NTF \c_stex_persist_write_mode_bool {
619     \stex_if_html_backend:TF{
620         \cs_new:Npn \stex_persist:n #1 {}
621         \cs_new:Npn \stex_persist:e #1 {}
622     }{
623         \cs_new_protected:Nn \stex_persist:n {
624             \iow_now:Nn \c__stex_persist_sms_iow {#1}
625         }
626         \cs_generate_variant:Nn \stex_persist:n {e}
627     }
628 }{
629     \cs_new:Npn \stex_persist:n #1 {}
630     \cs_new:Npn \stex_persist:e #1 {}
631 }

```

(End of definition for `\stex_persist:n`. This function is documented on page [141](#).)

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

```

632 \cs_new_protected:Nn \__stex_persist_load_file:n {
633   \file_if_exist:nT{#1} {
634     \group_begin:
635     \cs_set:Npn \stex_persist:n ##1 {}
636     \cs_set:Npn \stex_persist:e ##1 {}
637     \stex_debug:n{persist}{restoring~from~sms~file}
638     \catcode`\ =10\relax
639     \cs:w @ @ input \cs_end:#1\relax
640     \group_end:
641   }
642 }
643 }
644
645 \cs_new_protected:Nn \__stex_persist_write_only: {
646   \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
647   \AtEndDocument{ \iow_close:N \c__stex_persist_sms_iow }
648 }
649
650 \cs_new_protected:Nn \__stex_persist_read_and_write: {
651   \file_if_exist:nTF{\jobname.sms} {
652     \ior_open:Nn \g_tmpa_ior {\jobname.sms}
653     \iow_open:Nn \g_tmpa_iow {\jobname sms2}
654     \ior_str_map_inline:Nn \g_tmpa_ior {
655       \iow_now:Nn \g_tmpa_iow {##1}
656     }
657     \iow_close:N \g_tmpa_iow
658     \ior_close:N \g_tmpa_ior
659     \__stex_persist_write_only:
660     \ior_open:Nn \g_tmpa_ior {\jobname sms2}
661     \ior_str_map_inline:Nn \g_tmpa_ior {
662       \iow_now:Nn \c__stex_persist_sms_iow {##1}
663     }
664     \ior_close:N \g_tmpa_ior
665     \__stex_persist_load_file:n{\jobname sms2}
666   }\__stex_persist_write_only:
667 }
668
669 \cs_new_protected:Nn \_stex_persist_read_now: {
670   \bool_if:NTF \c_stex_persist_mode_bool {
671     \bool_if:NTF \c_stex_persist_write_mode_bool {
672       \__stex_persist_read_and_write:
673       {
674         \__stex_persist_load_file:n{\jobname sms}
675       }
676     }{
677       \bool_if:NT \c_stex_persist_write_mode_bool \__stex_persist_write_only:
678     }
679 }

```

### 13.2.9 Files, Paths and URIs

680 <@=stex\_path>

```
\stex_file_set:Nn
\stex_file_set:No
\stex_file_set:Nx
```

```

681 \cs_new_protected:Nn \stex_file_set:Nn {
682   \str_if_empty:nTF {#2} { \seq_clear:N #1 }{
683     \exp_args:NNno \seq_set_split:Nnn #1 / { \tl_to_str:n{#2} }
684   }
685 }
686 \cs_generate_variant:Nn \stex_file_set:Nn {No, Nx}

```

(End of definition for `\stex_file_set:Nn`. This function is documented on page 137.)

```

\stex_file_resolve:Nn
\stex_file_resolve:Nn
\stex_file_resolve:Nx
687 \sys_if_platform_windows:TF{
688   \cs_new_protected:Npn \__stex_path_win_take:w #1#2#3 \__stex_path_ : {
689     \uppercase{ \str_set:Nn \l_stex_path_str{#1}}
690     \str_set:Nx \l_stex_path_drive {\l_stex_path_str #2}
691     \str_set:Nn \l_stex_path_str{#3}
692   }
693   \cs_new_protected:Nn \stex_file_resolve:Nn {
694     \str_set:Nn \l_stex_path_str {#2}
695     \str_clear:N \l_stex_path_win_drive
696     \exp_args:NNo \str_replace_all:Nnn \l_stex_path_str \c_backslash_str /
697     \exp_args:Nx \str_if_eq:nnT {\str_item:Nn \l_stex_path_str 2} : {
698       \exp_after:wN \__stex_path_win_take:w \l_stex_path_str \__stex_path_ :
699     }
700     \stex_file_set:No #1 \l_stex_path_str
701     \__stex_path_canonicalize:N #1
702     \str_if_empty:NF \l_stex_path_win_drive {
703       \seq_pop_left:NN #1 \l_stex_path_str
704       \seq_put_left:No #1 \l_stex_path_win_drive
705     }
706     %\stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
707   }
708 }{
709   \cs_new_protected:Nn \stex_file_resolve:Nn {
710     \str_set:Nn \l_stex_path_str {#2}
711     \stex_file_set:No #1 \l_stex_path_str
712     \__stex_path_canonicalize:N #1
713     %\stex_debug:nn{files}{Set~\tl_to_str:n{#1}~to~\stex_file_use:N #1}
714   }
715 }
716 \cs_generate_variant:Nn \stex_file_resolve:Nn {No, Nx}
717
718 \cs_new_protected:Nn \__stex_path_canonicalize:N {
719   \seq_if_empty:NF #1 {
720     \seq_pop:NN #1 \l_stex_path_str
721     \seq_clear:N \l_stex_path_seq
722     \str_if_empty:NTF \l_stex_path_str {
723       \seq_map_function:NN #1 \__stex_path_dodots:n
724       \seq_put_left:Nn \l_stex_path_seq {}
725     }{
726       \seq_push:No #1 \l_stex_path_str
727       \seq_map_function:NN #1 \__stex_path_dodots:n
728     }
729     \seq_set_eq:NN #1 \l_stex_path_seq
730   }

```

```

731 }
732 \cs_new_protected:Nn \__stex_path_dodots:n {
733   \str_if_empty:nF{#1} {
734     \str_if_eq:nnF {#1} {.} {
735       \str_if_eq:nnTF {#1} {..} {
736         \seq_if_empty:NF \l__stex_path_seq {
737           \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
738         }
739       }{
740         \seq_put_right:Nn \l__stex_path_seq {#1}
741       }
742     }
743   }
744 }
745 }
```

(End of definition for `\stex_file_resolve:Nn`. This function is documented on page 137.)

```

\stex_if_file_absolute_p:N
\stex_if_file_absolute:NTF
746 \sys_if_platform_windows:TF {
747   \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
748     \seq_if_empty:NTF #1 \prg_return_false: {
749       \tl_set:Nx \l__stex_path_maybein_str {\seq_item:Nn #1 1}
750       \exp_args:No \tl_if_empty:nTF \l__stex_path_maybein_str \prg_return_true: {
751         \exp_args:Nx \str_if_eq:nnTF {\str_item:Nn \l__stex_path_maybein_str 2} :
752           \prg_return_true: \prg_return_false:
753         }
754       }
755     }
756   }
757 }{
758   \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
759     \seq_if_empty:NTF #1 \prg_return_false: {
760       \exp_args:Nx \tl_if_empty:nTF {\seq_item:Nn #1 1}
761         \prg_return_true: \prg_return_false:
762       }
763     }
764 }
```

(End of definition for `\stex_if_file_absolute:NTF`. This function is documented on page 137.)

```

\stex_file_use:N
765 \cs_new:Nn \stex_file_use:N {
766   \seq_use:Nn #1 /
767 }
```

(End of definition for `\stex_file_use:N`. This function is documented on page 137.)

```

stex_if_file_starts_with:NNTF
768 \prg_new_protected_conditional:Nnn \stex_if_file_starts_with:NN {T,F,TF} {
769   \seq_set_eq:NN \l__stex_path_a_seq #1
770   \seq_set_eq:NN \l__stex_path_b_seq #2
771   \tl_clear:N \l__stex_path_return_tl
772   \bool_while_do:nn{
773     \bool_not_p:n{
```

```

774     \bool_lazy_any_p:n{
775         {\seq_if_empty_p:N \l__stex_path_a_seq}
776         {\seq_if_empty_p:N \l__stex_path_b_seq}
777         {\bool_not_p:n{\tl_if_empty_p:N \l__stex_path_return_tl}}
778     }
779 }
780 }{
781     \seq_pop_left:NN \l__stex_path_a_seq \l__stex_path_a_tl
782     \seq_pop_left:NN \l__stex_path_b_seq \l__stex_path_b_tl
783     \str_if_eq:NNF \l__stex_path_a_tl \l__stex_path_b_tl {
784         \tl_set:Nn \l__stex_path_return_tl {\prg_return_false:}
785     }
786 }
787 \tl_if_empty:NTF \l__stex_path_return_tl {
788     \seq_if_empty:NTF \l__stex_path_b_seq \prg_return_true: \prg_return_false:
789 } \l__stex_path_return_tl
790 }

```

(End of definition for `\stex_if_file_starts_with:NNTF`. This function is documented on page 137.)

```

\stex_file_split_off_ext:NN
\stex_file_split_off_lang:NN
791 \cs_new_protected:Nn \stex_file_split_off_ext:NN {
792     \seq_set_eq:NN #1 #2
793     \seq_pop_right:NN #1 \l__stex_path_str
794     \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
795     \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
796     \seq_put_right:Nx #1 {\seq_use:Nn \l__stex_path_seq .}
797 }
798 \cs_new_protected:Nn \stex_file_split_off_lang:NN {
799     \seq_set_eq:NN #1 #2
800     \seq_pop_right:NN #1 \l__stex_path_str
801     \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
802     \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
803
804     \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
805     \exp_args:NNo \prop_if_in:NnF \c_stex_languages_prop \l__stex_path_str {
806         \seq_put_right:No \l__stex_path_seq \l__stex_path_str
807     }
808
809     \seq_put_right:Nx #1 {\seq_use:Nn \l__stex_path_seq .}
810 }

```

(End of definition for `\stex_file_split_off_ext:NN` and `\stex_file_split_off_lang:NN`. These functions are documented on page 137.)

URIs:

```

\stex_map_uri:Nnnnn
811 \cs_set_protected:Nn \__stex_path_auth:n {
812     \msg_error:nnx{stex}{error/misused-uri}{\tl_to_str:n{#1}}
813 }
814 \cs_set_eq:NN \__stex_path_path:n \__stex_path_auth:n
815 \cs_set_eq:NN \__stex_path_module:n \__stex_path_auth:n
816 \cs_set_eq:NN \__stex_path_name:n \__stex_path_auth:n
817
818 \cs_set_protected:Nn \stex_map_uri:Nnnnn{

```

```

819 \stex_pseudogroup_with:nn{\__stex_path_auth:n\__stex_path_path:n\__stex_path_module:n\__st
820   \cs_set:Npn \__stex_path_auth:n ##1 {#2}
821   \cs_set:Npn \__stex_path_path:n ##1 {#3}
822   \cs_set:Npn \__stex_path_module:n ##1 {#4}
823   \cs_set:Npn \__stex_path_name:n ##1 {#5}
824   #1
825 }
826 }

```

(End of definition for `\stex_map_uri:Nnnnn`. This function is documented on page 138.)

```

\stex_uri_set:Nn
\stex_uri_set:Nn
\stex_uri_set:Nx
827 \str_set:Nx\__stex_path_colonslash{\cColonStr}
828 \cs_new_protected:Nn \__stex_path_uri_set:NnN {
829   \str_if_empty:nTF {#2} {
830     \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{empty}
831   }
832   \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn #1 \__stex_path_colonslash { \tl_to_str
833   \seq_pop_left:NN #1 \l__stex_path_auth_str
834   \seq_if_empty:NTF #1 {
835     \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{missing-authority}
836   }
837   \exp_args:NNnx \seq_set_split:Nnn #1 ? {\exp_args:NNo \seq_use:Nn #1 \__stex_path_colon
838   \seq_pop_left:NN #1 \l__stex_path_path
839   #3 \l__stex_path_path \l__stex_path_path
840   \seq_if_empty:NTF #1 {
841     \exp_args:NNo \__stex_path_uri_set:Nnxnn #1 \l__stex_path_auth_str
842     {\stex_file_use:N \l__stex_path_path} {} {}
843   }
844   \seq_pop_left:NN #1 \l__stex_path_mod
845   \seq_if_empty:NTF #1 {
846     \exp_args:NNo \__stex_path_uri_set:Nnxon #1 \l__stex_path_auth_str
847     {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod {}
848   }
849   \seq_pop_left:NN #1 \l__stex_path_name
850   \seq_if_empty:NTF #1 {
851     \exp_args:NNo \__stex_path_uri_set:Nnxon #2 \l__stex_path_auth_str
852     {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod \l__stex_path_name
853   }
854   \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{too-many~?s}
855   }
856   }
857 }
858 }
859 }
860 \stex_debug:nn{uris}{Set~\tl_to_str:n{#1}~to~\stex_uri_use:N #1}
861 }
862
863 \cs_new_protected:Nn \__stex_path_uri_set:Nnnnn{
864   \tl_set:Nn #1 {
865     \__stex_path_auth:n{ #2 }
866     \__stex_path_path:n{ #3 }
867     \__stex_path_module:n{ #4 }
868     \__stex_path_name:n{ #5 }

```

```

869     }
870   }
871 \cs_generate_variant:Nn\__stex_path_uri_set:Nnnnn {Nnxnn,Nnxon,Nnxoo}
872
873 \cs_new_protected:Nn \stex_uri_set:Nn {
874   \__stex_path_uri_set:NnN #1 {#2} \stex_file_set:No
875 }
876 \cs_generate_variant:Nn \stex_uri_set:Nn {No, Nx}

```

(End of definition for `\stex_uri_set:Nn`. This function is documented on page 138.)

### `\stex_uri_resolve:Nn`

```

\stex_uri_resolve:Nn
\stex_uri_resolve:No
\stex_uri_resolve:Nx
877 \cs_new_protected:Nn \stex_uri_resolve:Nn {
878   \__stex_path_uri_set:NnN #1 {#2} \stex_file_resolve:No
879 }
880 \cs_generate_variant:Nn \stex_uri_resolve:Nn {No, Nx}

```

(End of definition for `\stex_uri_resolve:Nn`. This function is documented on page 138.)

### `\stex_uri_use:N`

```

881 \cs_new:Npn \__stex_path_uri_use:w \__stex_path_auth:n #1 \__stex_path_path:n #2 \__stex_pat
882   #1\c_colon_str/ #2 \tl_if_empty:nF { #3 }{ ? #3
883     \tl_if_empty:nF { #4 }{ ? #4 } }
884 }
885 \cs_new:Nn \stex_uri_use:N {
886   \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N #1 } \__stex_path_auth:n {
887     \exp_after:wn \__stex_path_uri_use:w #1
888   }{
889     \msg_error:nnnn{stex}{error/invalid-uri}{#1}{Not~a~URI}
890   }
891 }

```

(End of definition for `\stex_uri_use:N`. This function is documented on page 138.)

### `\stex_uri_from_repo_file>NNNn`

```

\stex_uri_from_repo_file_nolang:NNNn
892 \cs_new_protected:Npn \stex_uri_from_repo_file_nolang:NNNn {
893   \__stex_path_from_repo_file:NNNNn \stex_file_split_off_lang:NN
894 }
895 \cs_new_protected:Npn \stex_uri_from_repo_file:NNNn {
896   \__stex_path_from_repo_file:NNNNn \stex_file_split_off_ext:NN
897 }
898
899 \cs_new_protected:Nn \__stex_path_from_repo_file:NNNNn {
900   #1 \l_stex_path_file #4
901   \prop_if_exist:NTF #3 {
902     \str_clear:N \l_stex_path_uri
903     \prop_get:NnNF #3 {#5} \l_stex_path_uri {
904       \prop_get:NnNF #3 {ns} \l_stex_path_uri {
905         \__stex_path_uri_set:Nnxnn #2 {file}
906         {\stex_file_use:N \l_stex_path_file} {} {}
907       }
908     }
909     \str_if_empty:NF \l_stex_path_uri {\__stex_path_relativize:N #2}
910   }{
911     \exp_args:NNx \__stex_path_uri_set:Nnxnn #2 {\tl_to_str:n{file}}

```

```

912     {\stex_file_use:N \l__stex_path_file} {} {}
913   }
914 }
915 \cs_new_protected:Nn \__stex_path_relativize:N {
916   \seq_set_eq:NN \l__stex_path_seq \l__stex_path_file
917   \seq_map_inline:Nn \c_stex_mathhub_file { % mathhub path
918     \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
919   }
920   \stex_file_set:Nx \l__stex_path_path {\prop_item:Nn \l_stex_current_archive_prop {id} }
921   \seq_map_inline:Nn \l__stex_path_path { % id
922     \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
923   }
924   \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl % source
925   \stex_uri_set:Nx #1 { \l__stex_path_uri / \stex_file_use:N \l__stex_path_seq }
926
927 }
```

(End of definition for `\stex_uri_from_repo_file:NNNn` and `\stex_uri_from_repo_file_nolang:NNNn`.  
These functions are documented on page [138](#).)

```

\stex_uri_from_current_file:Nn
\stex_uri_from_current_file_nolang:Nn
929 \cs_new_protected:Nn \stex_uri_from_current_file:Nn {
930   \stex_debug:nn{docuri}{Here:~\stex_file_use:N \g_stex_current_file}
931   \stex_uri_from_repo_file:NNNn #1 \l_stex_current_archive_prop
932   \g_stex_current_file {#2}
933   \stex_debug:nn{docuri}{resolved:~\stex_uri_use:N #1}
934 }
935 \cs_new_protected:Nn \stex_uri_from_current_file_nolang:Nn {
936   \stex_uri_from_repo_file_nolang:NNNn #1 \l_stex_current_archive_prop
937   \g_stex_current_file {#2}
938 }
```

(End of definition for `\stex_uri_from_current_file:Nn` and `\stex_uri_from_current_file_nolang:Nn`.  
These functions are documented on page [139](#).)

```

\stex_uri_add_module:NNn
\stex_uri_add_module:NNo
939 \cs_new_protected:Nn \stex_uri_add_module:NNn {
940   \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N #2 } \__stex_path_auth:n {
941     \stex_pseudogroup_with:nn
942     {\__stex_path_auth:n\__stex_path_path:n\__stex_path_module:n\__stex_path_name:n}
943     {
944       \cs_set:Npn \__stex_path_module:n ##1 {
945         \tl_if_empty:nTF{##1} {
946           \exp_not:N \__stex_path_module:n {#3}
947         }{
948           \msg_error:nnn{stex}{error/invalid-dpath}{#2}
949         }
950       }
951       \cs_set:Npn \__stex_path_name:n ##1 {
952         \tl_if_empty:nTF{##1} {
953           \exp_not:N \__stex_path_name:n {}
954         }{
955           \msg_error:nnn{stex}{error/invalid-dpath}{#2}
956         }
957     }
958   }
```

```

957         }
958         \tl_set:Nx #1 {#2}
959     }
960     }{
961         \msg_error:nnnn{stex}{error/invalid-uri}{#2}{Not~a~URI}
962     }
963 }
964 \cs_generate_variant:Nn \stex_uri_add_module:NNn {NNo}

```

(End of definition for `\stex_uri_add_module:NNn`. This function is documented on page 139.)

### `\l_stex_current_doc_uri`

```
965 \tl_new:N \l_stex_current_doc_uri
```

(End of definition for `\l_stex_current_doc_uri`. This variable is documented on page 139.)

### `\stex_set_document_uri:`

```

966 \cs_new_protected:Nn \stex_set_document_uri: {
967     \stex_uri_from_current_file:Nn \l_stex_current_doc_uri {narr}
968     \%stex_debug:nn{sref}{Document~URI:~\stex_uri_use:N \l_stex_current_doc_uri}
969 }

```

(End of definition for `\stex_set_document_uri:`. This function is documented on page 139.)

### `\stex_set_current_namespace:`

```

970 \cs_new_protected:Nn \stex_set_current_namespace: {
971     \stex_uri_from_current_file_nolang:Nn \l_stex_current_ns_uri {source-base}
972     \%stex_debug:nn{modules}{Namespace~URI:~\stex_uri_use:N \l_stex_current_ns_uri}
973 }

```

(End of definition for `\stex_set_current_namespace:`. This function is documented on page 139.)

We determine the PWD

```

\c_stex_pwd_file
\c_stex_main_file
974 \sys_if_platform_windows:TF{
975     \stex_get_env:Nn\l_stex_path_str{CD}
976 }{
977     \stex_get_env:Nn\l_stex_path_str{PWD}
978 }
979 \stex_file_resolve:No \c_stex_pwd_file \l_stex_path_str
980 \seq_set_eq:NN \c_stex_main_file \c_stex_pwd_file
981 \seq_put_right:Nx \c_stex_main_file {\jobname\tl_to_str:n{.tex}}
982
983 \stex_debug:nn {files} {PWD:-\stex_file_use:N \c_stex_pwd_file}

```

(End of definition for `\c_stex_pwd_file` and `\c_stex_main_file`. These variables are documented on page 138.)

### 13.2.10 File Hooks

keeps track of file changes:

```
984 \seq_gclear_new:N\g__stex_path_stack
985 \seq_gclear_new:N\g_stex_current_file
```

`\stex_filestack_push:n`

```
986 \cs_new_protected:Nn \stex_filestack_push:n {
987     \stex_if_ends_with:nnTF {#1}{.tex} {
988         \stex_file_resolve:No \g_stex_current_file {#1}
989     }{
990         \stex_file_resolve:No \g_stex_current_file {#1.tex}
991     }
992     \stex_if_file_absolute:NF \g_stex_current_file {
993         \stex_file_resolve:Nx \g_stex_current_file {
994             \stex_file_use:N \c_stex_pwd_file / \stex_file_use:N \g_stex_current_file
995         }
996     }
997     \seq_gset_eq:NN \g_stex_current_file \g_stex_current_file
998     \exp_args:NNx \seq_gpush:Nn \g__stex_path_stack {\stex_file_use:N \g_stex_current_file}
999     \_stex_every_file:
1000 }
1001
1002 \cs_new_protected:Nn \_stex_every_file: {
1003     \stex_set_document_uri:
1004     \stex_language_from_file:
1005     \stex_set_current_namespace:
1006 }
1007 \%AtBeginDocument{\_stex_every_file:}
```

(End of definition for `\stex_filestack_push:n`. This function is documented on page 137.)

`\stex_filestack_pop:`

```
1008 \cs_new_protected:Nn \stex_filestack_pop: {
1009     \seq_if_empty:NF \g__stex_path_stack {
1010         \seq_gpop>NN \g__stex_path_stack \l__stex_path_str
1011     }
1012     \seq_if_empty:NTF \g__stex_path_stack {
1013         \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
1014     }{
1015         \seq_get>NN \g__stex_path_stack \l__stex_path_str
1016         \exp_args:NNo \stex_file_set:Nn \g_stex_current_file \l__stex_path_str
1017         \seq_gset_eq:NN \g_stex_current_file \g_stex_current_file
1018     }
1019     \_stex_every_file:
1020 }
```

(End of definition for `\stex_filestack_pop:`. This function is documented on page 137.)

Hooks for the current file:

```
1021 \cs_new_protected:Nn \stex_input_with_hooks:n {
1022     \tl_set:Nn \l__stex_path_do_hooks_pre_tl {
1023         \tl_gset:Nn \l__stex_path_do_hooks_pre_tl {}
1024         \stex_debug:nn{HERE}{Hook~for~#1^Jmeaning\CurrentFilePath^J\CurrentFile}
1025         \tl_if_empty:NTF\CurrentFilePath{
```

```

1026     \exp_args:No \stex_filestack_push:n {\CurrentFile}
1027     }{
1028         \exp_args:Ne \stex_filestack_push:n { \CurrentFilePath / \CurrentFile }
1029     }
1030 }
1031 \input{#1}
1032 \stex_debug:nn{HERE}{Hook-end-for~#1}
1033 \stex_filestack_pop:
1034 }
1035 \tl_new:N \l__stex_path_do_hooks_pre_tl {}
1036
1037 \AddToHook{file/before}{
1038     \l__stex_path_do_hooks_pre_tl
1039 }
1040 \%AddToHook{file/after}{ \stex_filestack_pop: }

```

### 13.3 Math Archives

```

1041 <@=stex_mathhub>
\mathhub
\c_stex_home_file
\c_stex_mathhub_file
1042
1043 \sys_if_platform_windows:TF{
1044     \stex_get_env:Nn \l__stex_mathhub_str {homedrive\c_percent_str\c_percent_str homepath}
1045 }{
1046     \stex_get_env:Nn \l__stex_mathhub_str {HOME}
1047 }
1048 \stex_file_resolve:No \c_stex_home_file \l__stex_mathhub_str
1049
1050 \str_if_empty:NTF\mathhub{
1051     \stex_get_env:Nn \l__stex_mathhub_str {MATHHUB}
1052     \str_if_empty:NTF \l__stex_mathhub_str {
1053         \ior_open:NnTF \g_tmpa_iorf{\stex_file_use:N \c_stex_home_file/.stex/mathhub.path}{
1054             \group_begin:
1055                 \escapechar=-1\catcode`\\"=12
1056                 \ior_str_get:NN \g_tmpa_ior \l__stex_mathhub_str
1057                 \str_gset_eq:NN \l__stex_mathhub_str \l__stex_mathhub_str
1058             \group_end:
1059             \ior_close:N \g_tmpa_ior
1060             \stex_debug:nn{mathhub}{MathHub-directory-determined-from-home-directory}
1061         }{
1062             \str_clear:N \l__stex_mathhub_str
1063         }
1064     }{
1065         \stex_debug:nn{mathhub}{MathHub-directory-determined-from-environment-variable}
1066     }
1067 }{
1068     \str_set_eq:NN \l__stex_mathhub_str \mathhub
1069 }
1070
1071 \str_if_empty:NTF \l__stex_mathhub_str {
1072     \msg_warning:nn{stex}{warning/nomathhub}

```

```

1073   \exp_args:NNe \stex_file_set:Nn \c_stex_mathhub_file {\stex_file_use:N \c_stex_home_file \
1074 }{
1075   \stex_file_resolve:No \c_stex_mathhub_file \l_stex_mathhub_str
1076   \stex_if_file_absolute:NF \c_stex_mathhub_file {
1077     \exp_args:NNe \stex_file_resolve:Nn \c_stex_mathhub_file {
1078       \stex_file_use:N \c_stex_main_file / .. / \l_stex_mathhub_str
1079     }
1080   }
1081 }
1082
1083 \exp_args:NNe \str_set:Nn \mathhub {\stex_file_use:N \c_stex_mathhub_file}
1084 \stex_debug:nn{mathhub}{MATHHUB:~\mathhub}

```

(End of definition for `\mathhub`, `\c_stex_home_file`, and `\c_stex_mathhub_file`. These variables are documented on page ??.)

### `\l_stex_current_archive`

#### `\stex_set_current_archive:n`

```

1085 \cs_new_protected:Nn \stex_set_current_archive:n {
1086   \stex_require_archive:n { #1 }
1087   \stex_debug:nn{mathhub}{switching-to-archive-#1}
1088   \prop_set_eq:Nc \l_stex_current_archive_prop {
1089     c_stex_mathhub_#1_manifest_prop
1090   }
1091 }

```

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

#### `\stex_in_archive:nn`

```

1092 \cs_new_protected:Nn \stex_in_archive:nn {
1093   \cs_if_exist:NF \l_stex_mathhub_cs {
1094     \cs_set:Npn \l_stex_mathhub_cs ##1 {}
1095   }
1096   \stex_pseudogroup:nn{
1097     \cs_set:Npn \l_stex_mathhub_cs ##1 {#2}
1098     \tl_if_empty:nTF{#1} {
1099       \prop_if_exist:NTF \l_stex_current_archive_prop {
1100         \exp_args:Ne \l_stex_mathhub_cs {\prop_item:Nn \l_stex_current_archive_prop { id }}
1101       }{
1102         \l_stex_mathhub_cs {}
1103       }
1104     }{
1105       \stex_set_current_archive:n{#1}
1106       \l_stex_mathhub_cs {#1}
1107     }
1108   }{
1109     \stex_pseudogroup_restore:N \l_stex_current_archive_prop
1110     \cs_set:Npn \exp_not:N \l_stex_mathhub_cs ##1 {
1111       \exp_args:No \exp_not:n {\l_stex_mathhub_cs {##1}}
1112     }
1113   }
1114 }

```

(End of definition for `\stex_in_archive:nn`. This function is documented on page 134.)

```

\stex_require_archive:n
\stex_require_archive:o
1115 \cs_new_protected:Nn \stex_require_archive:n {
1116   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
1117     \seq_if_empty:NTF \c_stex_mathhub_file {
1118       \msg_fatal:nn{stex}{warning/nomathhub}
1119     }{
1120       \stex_debug:nn{mathhub}{Opening~archive:~#1}
1121       \__stex_mathhub_do_manifest:n { #1 }
1122     }
1123   }
1124 }
1125 \cs_generate_variant:Nn \stex_require_archive:n {o}

```

(End of definition for `\stex_require_archive:n`. This function is documented on page 134.)

Code for finding and parsing manifest files:

```

1126 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
1127   \exp_args:Ne \__stex_mathhub_find_manifest:n {\stex_file_use:N \c_stex_mathhub_file / #1}
1128   \str_if_empty:NT \l__stex_mathhub_manifest_str {
1129     \msg_fatal:nne{stex}{error/noarchive}
1130     {#1}\stex_file_use:N \c_stex_mathhub_file
1131   }
1132   \__stex_mathhub_parse_manifest:n {#1}
1133 }
1134
1135 \cs_new_protected:Nn \__stex_mathhub_find_manifest:n {
1136   \str_clear:N \l__stex_mathhub_manifest_str
1137   \seq_set_split:Nnn \l__stex_mathhub_seq / {#1}
1138   \bool_set_true:N \l__stex_mathhub_bool
1139   \bool_while_do:Nn \l__stex_mathhub_bool {
1140     \tl_if_eq:NNTF \l__stex_mathhub_seq \c_stex_mathhub_file {
1141       \bool_set_false:N \l__stex_mathhub_bool
1142     }{
1143       \__stex_mathhub_check_manifest:
1144       \bool_if:NT \l__stex_mathhub_bool {
1145         \seq_pop_right:NN \l__stex_mathhub_seq \l__stex_mathhub_tl
1146       }
1147     }
1148   }
1149 }
1150 \cs_generate_variant:Nn \__stex_mathhub_find_manifest:n {x}
1151
1152 \cs_new_protected:Nn \__stex_mathhub_check_manifest: {
1153   \__stex_mathhub_check_manifest:n {MANIFEST.MF}
1154   \bool_if:NT \l__stex_mathhub_bool {
1155     \__stex_mathhub_check_manifest:n {META-INF/MANIFEST.MF}
1156     \bool_if:NT \l__stex_mathhub_bool {
1157       \__stex_mathhub_check_manifest:n {meta-inf/MANIFEST.MF}
1158     }
1159   }
1160 }
1161
1162 \cs_new_protected:Nn \__stex_mathhub_check_manifest:n {
1163   \stex_debug:nn{mathhub}{Checking~\stex_file_use:N \l__stex_mathhub_seq / #1}
1164   \file_if_exist:nT {\stex_file_use:N \l__stex_mathhub_seq / #1} {

```

```

1165     \bool_set_false:N \l__stex_mathhub_bool
1166     \str_set:Nx \l__stex_mathhub_manifest_str {\stex_file_use:N \l__stex_mathhub_seq / #1}
1167   }
1168 }
1169
1170 \ior_new:N \c__stex_mathhub_manifest_ior
1171 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
1172   \ior_open:Nn \c__stex_mathhub_manifest_ior \l__stex_mathhub_manifest_str
1173   \prop_clear:N \l__stex_mathhub_prop
1174   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
1175     \exp_args:NNNo \exp_args:NNN
1176       \seq_set_split:Nnn \l__stex_mathhub_seq \c_colon_str {\tl_to_str:n{##1}}
1177     \seq_pop_left:NNT \l__stex_mathhub_seq \l__stex_mathhub_key {
1178       \exp_args:NNo \str_set:Nn \l__stex_mathhub_key \l__stex_mathhub_key
1179       \str_set:Nx \l__stex_mathhub_val {\seq_use:Nn \l__stex_mathhub_seq :}
1180       \str_case:Nn \l__stex_mathhub_key {
1181         {id}           {\prop_put:Nno \l__stex_mathhub_prop { id }      \l__stex_mathhub_}
1182         {narration-base} {\prop_put:Nno \l__stex_mathhub_prop { narr }    \l__stex_mathhub_}
1183         {furl-base}    {\prop_put:Nno \l__stex_mathhub_prop { docurl }  \l__stex_mathhub_}
1184         {source-base}   {\prop_put:Nno \l__stex_mathhub_prop { ns }     \l__stex_mathhub_}
1185         {ns}            {\prop_put:Nno \l__stex_mathhub_prop { ns }     \l__stex_mathhub_}
1186       }
1187     }
1188   }
1189 \ior_close:N \c__stex_mathhub_manifest_ior
1190 \prop_gset_eq:cN { c_stex_mathhub_#1_manifest_prop } \l__stex_mathhub_prop
1191 \stex_debug:nn{mathhub}{Result:~\prop_to_keyval:N \l__stex_mathhub_prop}
1192 \stex_persist:e {
1193   \prop_gset_from_keyval:cn {c_stex_mathhub_#1_manifest_prop} {
1194     \prop_to_keyval:N \l__stex_mathhub_prop
1195   }
1196 }
1197 }

```

Current MathHub archive

```

\c_stex_main_archive_prop
\l_stex_current_archive_prop
1198 \cs_new_protected:Nn \stex_main_archive: {
1199   \stex_if_file_starts_with:NNTF \c_stex_pwd_file \c_stex_mathhub_file {
1200     \__stex_mathhub_find_manifest:x { \stex_file_use:N \c_stex_pwd_file }
1201     \str_if_empty:NT \l__stex_mathhub_manifest_str {
1202       \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~archive}
1203     }{
1204       \__stex_mathhub_parse_manifest:n { main }
1205       \prop_set_eq:NN \c_stex_main_archive_prop \c_stex_mathhub_main_manifest_prop
1206       \cs_undefine:N \c_stex_mathhub_main_manifest_prop
1207       \prop_get:NnN \c_stex_main_archive_prop {id}
1208         \l__stex_mathhub_str
1209       \prop_set_eq:cN { c_stex_mathhub_\l__stex_mathhub_str _manifest_prop }
1210         \c_stex_main_archive_prop
1211       \exp_args:No \stex_set_current_archive:n { \l__stex_mathhub_str }
1212       \stex_debug:nn{mathhub}{Current~archive:~}
1213         \prop_item:Nn \l_stex_current_archive_prop {id}
1214     }
1215     \bool_if:NT \c_stex_persist_write_mode_bool {

```

```

1216   \tl_put_right:Nx \_stex_persist_read_now: {
1217     \stex_persist:n {
1218       \prop_gset_from_keyval:c {c_stex_mathhub_\l_stex_mathhub_str _manifest_prop}{}
1219       \prop_to_keyval:N \c_stex_main_archive_prop
1220     }
1221     \prop_gset_eq:Nc \exp_not:N \l_stex_current_archive_prop {
1222       c_stex_mathhub_\l_stex_mathhub_str _manifest_prop
1223     }
1224     \prop_gset_eq:Nc \exp_not:N \c_stex_main_archive_prop {
1225       c_stex_mathhub_\l_stex_mathhub_str _manifest_prop
1226     }
1227   }
1228 }
1229 }
1230 }
1231 }{
1232   \stex_debug:nn{mathhub}{Not~currently~in~the~MathHub~directory}
1233 }
1234 }

1235 \%bool_if:NF \c_stex_persist_mode_bool {
1236   \_stex_main_archive:
1237 }
1238 }

(End of definition for \c_stex_main_archive_prop and \l_stex_current_archive_prop. These variables are documented on page 134.)
```

## 13.4 Documents

### 13.4.1 Title

Stores the title, if it exists:

```

1239 <@=stex_doc>
1240 \tl_new:N \g__stex_doc_title_tl
```

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

```

1241 \cs_new_protected:Npn \stexdoctitle #1 {
1242   \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1243   \global\def\stexdoctitle##1{}
1244 }
```

At begin document, we switch to:

```

1245 \cs_new_protected:Nn \__stex_doc_set_title:n {
1246   \stex_if_smsmode:F{
1247     \global\def\stexdoctitle##1{}
1248     \stex_debug:nn{title}{Setting~title~to:\tl_to_str:n##1}
1249     \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1250     \__stex_doc_title_html:
1251   }
1252 }
```

Hooks, changes and HTML:

```

1253 \cs_new_protected:Nn \__stex_doc_title_html: {
1254   \stex_if_do_html:T{\stex_if_html_backend:T{
```

```

1255     \stex_annotation_invisible:nn{shml:doctitle={}}{ \hbox{\g__stex_doc_title_tl} }
1256   }
1257 }
1258
1259 \AtBeginDocument {
1260   \tl_if_empty:NTF \g__stex_doc_title_tl {
1261     \cs_set_eq:NN \stexdoctitle \__stex_doc_set_title:n
1262   }{
1263     \stex_debug:nn{title}{Setting~title~to:\exp_args:No\tl_to_str:n\g__stex_doc_title_tl}
1264     \global\def\stexdoctitle{\#1}
1265     \__stex_doc_title_html:
1266   }
1267
1268   \cs_set_eq:NN \__stex_doc_maketitle: \maketitle
1269   \global\protected\def\maketitle{
1270     \tl_if_empty:NF \@title {
1271       \exp_args:No \stexdoctitle \@title
1272     }
1273     \__stex_doc_maketitle:
1274   }
1275 }
```

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

### 13.4.2 Sectioning

```

1276 \int_new:N \l_stex_docheader_sect
1277
1278 \tl_set:Nn \stex_current_section_level {document}
1279
1280 \cs_set_protected:Npn \currentsectionlevel {
1281   \stex_if_do_html:TF{
1282     \stex_annotation:nn{shml:currentsectionlevel={},shml:capitalize=false}{}}
1283   }{
1284     \stex_current_section_level
1285   }
1286 \tl_if_exist:NT\xspace\xspace
1287 }
1288
1289 \cs_set_protected:Npn \Currentsectionlevel {
1290   \stex_if_do_html:TF{
1291     \stex_annotation:nn{shml:currentsectionlevel={},shml:capitalize=true}{}}
1292   }{
1293     \exp_args:No \__stex_capitalize:n \stex_current_section_level
1294   }
1295 \tl_if_exist:NT\xspace\xspace
1296 }
1297
1298 \stex_if_html_backend:TF {
1299   \cs_new_protected:Nn \_sfragment_do_level:nn {
1300     \stexdoctitle{\#2}
1301     \par
1302     \begin{stex_annotation_env}{shml:section={\int_use:N \l_stex_docheader_sect}}
1303       \noindent\stex_annotation:nn{shml:sectiontitle={}}{
```

```

1304     \_stex_annotation_force_break:n{#2}
1305   }\par
1306 }
1307 \cs_new_protected:Nn \_sfragment_end: {
1308   \end{stex_annotation_env}
1309 }
1310 }{
1311   \cs_new_protected:Nn \_sfragment_do_level:nn {
1312     \stexdotitle{#2}
1313     \tl_if_empty:NTF \l_stex_key_short_tl {
1314       \use:c{#1}
1315     }{
1316       \exp_args:Nnx \use:nn{\use:c{#1}}{[\exp_args:No \exp_not:n \l_stex_key_short_tl]}
1317     }{#2}
1318     \int_incr:N \l_stex_docheader_sect
1319     \tl_set:Nn \stex_current_section_level{#1}
1320   }
1321   \cs_new_protected:Nn \_sfragment_end: {}
1322 }
1323
1324
1325 \cs_new_protected:Npn \__stex_doc_do_section:n {
1326   \int_case:nnF \l_stex_docheader_sect {
1327     {0}{\cs_if_exist:NTF \thepart {\_sfragment_do_level:nn{part}}{
1328       \int_incr:N \l_stex_docheader_sect
1329       \__stex_doc_do_section:n
1330     }}
1331     {1}{\cs_if_exist:NTF \thechapter {\_sfragment_do_level:nn{chapter}}{
1332       \int_incr:N \l_stex_docheader_sect
1333       \__stex_doc_do_section:n
1334     }}
1335     {2}{\_sfragment_do_level:nn{section}}
1336     {3}{\_sfragment_do_level:nn{subsection}}
1337     {4}{\_sfragment_do_level:nn{subsubsection}}
1338     {5}{\_sfragment_do_level:nn{paragraph}}
1339     {\_sfragment_do_level:nn{subparagraph}}
1340   }
1341
1342 \stex_keys_define:nnnn{ sfragment }{
1343   \tl_clear:N \l_stex_key_short_tl
1344 }{
1345   short .tl_set:N = \l_stex_key_short_tl
1346 }{id}
1347
1348 \NewDocumentEnvironment{sfragment}{ O{} m} {
1349   \stex_keys_set:nn{sfragment}{#1}
1350   \__stex_doc_do_section:n{#2}
1351   \stex_do_id:
1352 }{
1353   \_sfragment_end:
1354 }
1355
1356 \%int_incr:N \l_stex_docheader_sect
1357 \NewDocumentEnvironment{blindfragment}{}{

```

```

1358     \__stex_doc_skip_section:
1359 }{
1360   \stex_if_html_backend:T{
1361     \stex_annotate_invisible:n{~}
1362     \end{stex_annotate_env}
1363   }
1364 }
1365
1366
1367 \cs_new_protected:Nn \__stex_doc_skip_section_i: {
1368   \int_case:nn \l_stex_docheader_sect {
1369     {0}{\cs_if_exist:NF \thepart {
1370       \int_incr:N \l_stex_docheader_sect \__stex_doc_skip_section_i:
1371     }}
1372     {1}{\cs_if_exist:NF \thechapter {
1373       \int_incr:N \l_stex_docheader_sect \__stex_doc_skip_section_i:
1374     }}
1375   }
1376   \int_incr:N \l_stex_docheader_sect
1377 }
1378
1379 \stex_if_html_backend:TF {
1380   \cs_new_protected:Nn \__stex_doc_skip_section: {
1381     \__stex_doc_skip_section_i:
1382     \begin{stex_annotate_env}{shtml:skipsection={\int_use:N \l_stex_docheader_sect}}
1383     \stex_annotate_invisible:n{~}
1384   }
1385 }
1386   \cs_set_eq:NN \__stex_doc_skip_section: \__stex_doc_skip_section_i:
1387 }
1388
1389
1390 \cs_new_protected:Nn \__stex_doc_skip_fragment:n {
1391   \stepcounter{#1}
1392 }
1393
1394 \cs_new_protected:Npn \skipfragment {
1395   \int_case:nnF \l_stex_docheader_sect {
1396     {0}{\cs_if_exist:NTF \thepart {\__stex_doc_skip_fragment:n{part}}{
1397       \int_incr:N \l_stex_docheader_sect
1398       \skipfragment
1399     }}
1400     {1}{\cs_if_exist:NTF \thechapter {\__stex_doc_skip_fragment:n{chapter}}{
1401       \int_incr:N \l_stex_docheader_sect
1402       \skipfragment
1403     }}
1404     {2}{\__stex_doc_skip_fragment:n{section}}
1405     {3}{\__stex_doc_skip_fragment:n{subsection}}
1406     {4}{\__stex_doc_skip_fragment:n{subsubsection}}
1407     {5}{\__stex_doc_skip_fragment:n{paragraph}}
1408   }{\__stex_doc_skip_fragment:n{subparagraph}}
1409 }

```

\setsectionlevel

```

1410 \cs_new_protected:Npn \setsectionlevel #1 {
1411   \str_case:nnF{#1}{
1412     {part}{\int_set:Nn \l_stex_docheader_sect 0}
1413     {chapter}{\int_set:Nn \l_stex_docheader_sect 1}
1414     {section}{\int_set:Nn \l_stex_docheader_sect 2}
1415     {subsection}{\int_set:Nn \l_stex_docheader_sect 3}
1416     {subsubsection}{\int_set:Nn \l_stex_docheader_sect 4}
1417     {paragraph}{\int_set:Nn \l_stex_docheader_sect 5}
1418   }{
1419     \int_set:Nn \l_stex_docheader_sect 6
1420   }
1421   \cs_if_eq:NNTF \onlypreamble \notprerr {
1422     \stex_annotation_invisible:nn{shml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
1423   }{}
1424 }
1425
1426 \stex_if_html_backend:T{
1427   \cs_new_protected:Nn \__stex_doc_check_topsect: {
1428     \int_case:nnF \l_stex_docheader_sect {
1429       {0}{\cs_if_exist:NTF \thechapter {
1430         \stex_annotation_invisible:nn{shml:sectionlevel=0}{}}
1431       }{
1432         \int_incr:N \l_stex_docheader_sect
1433         \__stex_doc_check_topsect:
1434       }
1435       {1}{\cs_if_exist:NTF \thechapter {
1436         \stex_annotation_invisible:nn{shml:sectionlevel=1}{}}
1437       }{
1438         \int_incr:N \l_stex_docheader_sect
1439         \__stex_doc_check_topsect:
1440       }
1441     }{
1442       \stex_annotation_invisible:nn{shml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
1443     }
1444   }
1445   \AtBeginDocument{\__stex_doc_check_topsect:}
1446 }
1447
1448 \AtBeginDocument{
1449   \bool_if:NF \c_stex_no_frontmatter_bool {
1450     \cs_if_exist:NTF \frontmatter {
1451       \let\__stex_doc_orig_frontmatter \frontmatter
1452       \let\frontmatter \relax
1453     }{
1454       \tl_set:Nn \__stex_doc_orig_frontmatter {
1455         \clearpage
1456         \%@mainmatterfalse
1457         \pagenumbering{roman}
1458       }
1459     }
1460     \cs_if_exist:NTF \backmatter {
1461       \let\__stex_doc_orig_backmatter \backmatter
1462       \let\backmatter \relax
1463     }

```

```

1464     \tl_set:Nn\__stex_doc_orig_backmatter{
1465         \clearpage
1466         \%@\mainmatterfalse
1467         \pagenumbering{roman}
1468     }
1469 }
1470 \newenvironment{frontmatter} {
1471     \__stex_doc_orig_frontmatter
1472 }{
1473     \cs_if_exist:NTF\mainmatter{
1474         \mainmatter
1475     }{
1476         \clearpage
1477         \%@\mainmattertrue
1478         \pagenumbering{arabic}
1479     }
1480 }
1481 \newenvironment{backmatter} {
1482     \__stex_doc_orig_backmatter
1483 }{
1484     \cs_if_exist:NTF\mainmatter{
1485         \mainmatter
1486     }{
1487         \clearpage
1488         \%@\mainmattertrue
1489         \pagenumbering{arabic}
1490     }
1491 }
1492 }
1493 }

```

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

### 13.4.3 References

```
1494 <@=stex_refs>
```

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

```

1495 \iow_new:N \c__stex_refs_iow
1496 \AtBeginDocument{\iow_open:Nn \c__stex_refs_iow {\jobname.sref}}
1497 \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:

```

1498 \cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}
1499
1500 \cs_new_protected:Npn \STEXInternalSetSrefSymURL #1 #2 {
1501     \str_gset:cn{g_stex_sref_sym_\tl_to_str:n{#1}_target}{#2}
1502 }
1503

```

```

\stex_ref_new_doc_target:n
    \sreflabel
1504 \seq_new:N \g__stex_refs_files_seq
1505 \int_new:N \l__stex_refs_unnamed_counter_int

```

```

1506 \cs_new_protected:Nn \_stex_ref_new_id:n {
1507   \str_if_empty:nTF {#1} {
1508     \int_gincr:N \l__stex_refs_unnamed_counter_int
1509     \str_set:Nx \l__stex_refs_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1510   }{
1511     \str_set:Nn \l__stex_refs_str {#1}
1512   }
1513 }
1514 \str_set:Nx \l_stex_ref_url_str {\stex_uri_use:N \l_stex_current_doc_uri ? \l__stex_refs_s
1515 }
1516
1517 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1518   \stex_ref_new_id:n{#1}
1519   \%stex_uri_add_module:NNo \l__stex_refs_uri \l_stex_current_doc_uri \l__stex_refs_str
1520   \%stex_debug:nn{sref}{New-document-target:~\stex_uri_use:N \l__stex_refs_uri}
1521   \__stex_refs_add_doc_ref:xo {\stex_uri_use:N \l_stex_current_doc_uri} \l__stex_refs_str
1522   \stex_if_smemode:F {
1523     \iow_now:Nx \c__stex_refs_iow {
1524       \STEXInternalSrefRestoreTarget
1525       {\stex_uri_use:N \l_stex_current_doc_uri}
1526       {\l__stex_refs_str}
1527       {\l@currentcounter}
1528       {\l@currentlabel}
1529       {
1530         \tl_if_exist:NT\l@currentlabelname{
1531           \exp_args:No\exp_not:n\l@currentlabelname
1532         }
1533       }
1534     }
1535     \exp_args:Nx \label {sref@\l_stex_ref_url_str}
1536     \stex_if_do_html:T {
1537       \pdfdest name "sref@\l_stex_ref_url_str" xyz\relax
1538     }
1539   }
1540 }
1541 \NewDocumentCommand \sreflabel {m} {\stex_ref_new_doc_target:n {#1}}
1542
1543 \cs_new_protected:Nn \__stex_refs_add_doc_ref:nn {
1544   \seq_if_in:NnTF \g__stex_refs_files_seq {#1} {
1545     \seq_if_in:cnF {g__stex_refs_#1_seq}{#2} {
1546       \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1547     }
1548   }{
1549     \seq_gput_right:Nn \g__stex_refs_files_seq {#1}
1550     \seq_new:c{g__stex_refs_#1_seq}
1551     \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1552   }
1553 }
1554 \cs_generate_variant:Nn \__stex_refs_add_doc_ref:nn {xo,xx}

```

(End of definition for `\stex_ref_new_doc_target:n` and `\sreflabel`. These functions are documented on page 121.)

**\sref** Optional arguments:  
**\extref**

```

1555 \stex_keys_define:nnnn{sref / 1}{}{
1556   % TODO get rid of this
1557   fallback .code:n = {},
1558   pre     .code:n = {},
1559   post    .code:n = {}
1560 }{archive file}
1561 \stex_keys_define:nnnn{sref / 2}{}{}{archive file, title}
1562
1563 \str_new:N \l__stex_refs_default_archive_str
1564 \str_new:N \l__stex_refs_default_file_str
1565 \tl_new:N \l__stex_refs_default_title_tl
1566
1567 \cs_set_protected:Nn \__stex_refs_set_keys_b:n {
1568   \tl_if_empty:nTF{#1}{%
1569     \str_set_eq:NN \l_stex_key_archive_str \l__stex_refs_default_archive_str
1570     \str_set_eq:NN \l_stex_key_file_str \l__stex_refs_default_file_str
1571     \tl_set_eq:NN \l_stex_key_title_tl \l__stex_refs_default_title_tl
1572   }{%
1573     \stex_keys_set:nn{ sref / 2 }{ #1 }
1574   }
1575 }
1576
1577 \newcommand\srefsetin[3][]{%
1578   \str_set:Nx \l__stex_refs_default_archive_str {#1}
1579   \str_set:Nx \l__stex_refs_default_file_str {#2}
1580   \tl_set:Nn \l__stex_refs_default_title_tl {#3}
1581 }
1582
Auxiliary methods:
1583 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1584   \str_clear:N \l__stex_refs_uri_str
1585   \stex_debug:nn{sref}{%
1586     File:~\l_stex_key_file_str^~J
1587     Repo:\l_stex_key_archive_str
1588   }
1589   \str_if_empty:NTF \l_stex_key_file_str {
1590     \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1591     \seq_if_exist:cT{\g__stex_refs_\stex_uri_use:N \l_stex_current_doc_uri _seq}{%
1592       \exp_args:Nnx \__stex_refs_find_uri_in_file:nnn{#1}%
1593         {\l_stex_uri_use:N \l_stex_current_doc_uri}\seq_map_break:
1594     }
1595     \str_if_empty:NT \l__stex_refs_uri_str {
1596       \seq_map_inline:Nn \g__stex_refs_files_seq {
1597         \__stex_refs_find_uri_in_file:nnn{#1}{##1}{\seq_map_break:n{\seq_map_break:}}
1598       }
1599     }%
1600   }%
1601   \str_if_empty:NTF \l_stex_key_archive_str {
1602     \prop_if_exist:NTF \l_stex_current_archive_prop {
1603       \__stex_refs_find_uri_in_prop_file:N \l_stex_current_archive_prop
1604     }%
1605     \stex_file_resolve:Nx \l__stex_refs_file
1606       { \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str }

```

```

1607     \str_set:Nx \l__stex_refs_uri_str { file:/ \stex_file_use:N \l__stex_refs_file }
1608     }
1609     }{
1610         \stex_require_archive:o \l_stex_key_archive_str
1611         \prop_set_eq:Nc \l__stex_refs_prop { c_stex_mathhub_\l_stex_key_archive_str _manifest_
1612             \__stex_refs_find_uri_in_prop_file:N \l__stex_refs_prop
1613         }
1614     }
1615 }
1616
1617 \cs_new_protected:Nn \__stex_refs_find_uri_in_prop_file:N {
1618     \str_set:Nx \l__stex_refs_uri_str {
1619         \stex_file_use:N \c_stex_mathhub_file /
1620         \prop_item:Nn #1 {id} /
1621             source / \l_stex_key_file_str .sref
1622     }
1623     \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_uri_str
1624     \stex_uri_from_repo_file:NNNn \l__stex_refs_uri #1
1625         \l__stex_refs_file {narr}
1626     \str_set:Nx \l__stex_refs_uri_str {\stex_uri_use:N \l__stex_refs_uri}
1627 }
1628
1629 \cs_new_protected:Nn \__stex_refs_find_uri_in_file:nnn {
1630     \stex_debug:nn{sref}{Checking~file~#2}
1631     \seq_map_inline:cn{g__stex_refs_#2_seq}{%
1632         \str_if_eq:nnT{#1}{##1}{%
1633             \str_set:Nx \l__stex_refs_uri_str {\stex_uri_use:N \l_stex_current_doc_uri}
1634             \stex_debug:nn{sref}{Found.}
1635             #3
1636         }
1637     }
1638 }

```

Doing the actual referencing:

```

1639 \cs_new_protected:Nn \__stex_refs_do_autoref:n {
1640     \cs_if_exist:cTF{autoref}{%
1641         \exp_args:Nx\autoref{sref@#1}
1642     }{
1643         \exp_args:Nx\ref{sref@#1}
1644     }
1645 }
1646
1647 \cs_new_protected:Nn \__stex_refs_do_sref:nn {
1648     \str_if_empty:NTF \l__stex_refs_uri_str {
1649         \str_if_empty:NTF \l_stex_key_file_str {
1650             \stex_debug:nn{sref}{autoref~on~\stex_uri_use:N \l_stex_current_doc_uri?#1}
1651             \exp_args:Ne \__stex_refs_do_autoref:n{\stex_uri_use:N \l_stex_current_doc_uri ? #1}
1652         }{
1653             \stex_debug:nn{sref}{srefin~on~#1}
1654             \__stex_refs_set_keys_b:n{ #2 }
1655             \__stex_refs_do_sref_in:n{#1}
1656         }
1657     }{
1658         \exp_args:NNo \seq_if_in:NnTF \g__stex_refs_files_seq \l__stex_refs_uri_str {

```

```

1659 \stex_debug:nn{sref}{Using~ref~file~\l_stex_refs_uri_str}
1660 \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref_\l_stex_refs_uri_str_seq}{\detokenize{\#1}{}}
1661   \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref~on~\l_stex_refs_uri_str?}
1662   \l_stex_refs_do_autoref:n{\l_stex_refs_uri_str?\#1}
1663 }{
1664   \str_if_empty:NTF \l_stex_key_file_str {
1665     \stex_debug:nn{sref}{in~empty;~autoref~on~\l_stex_refs_uri_str?\#1}
1666     \l_stex_refs_do_autoref:n{\l_stex_refs_uri_str?\#1}
1667   }{
1668     \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l_stex_refs_uri_str?\#1}
1669     \l_stex_refs_set_keys_b:n{ \#2 }
1670     \l_stex_refs_do_sref_in:n{\#1}
1671   }
1672 }
1673 }{
1674   \stex_debug:nn{sref}{No~ref~file~found~for~\l_stex_refs_uri_str}
1675   \str_if_empty:NTF \l_stex_key_file_str {
1676     \stex_debug:nn{sref}{in~empty;~autoref~on~\l_stex_refs_uri_str?\#1}
1677     \l_stex_refs_do_autoref:n{\l_stex_refs_uri_str?\#1}
1678   }{
1679     \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l_stex_refs_uri_str?\#1}
1680     \l_stex_refs_set_keys_b:n{ \#2 }
1681     \l_stex_refs_do_sref_in:n{\#1}
1682   }
1683 }
1684 }
1685 }
1686 \cs_new_protected:Nn \l_stex_refs_do_sref_in:n {
1687   \stex_debug:nn{sref}{In: \l_stex_key_file_str^JRepo:\l_stex_key_archive_str}
1688   \stex_debug:nn{sref}{URI: \l_stex_refs_uri_str?\#1}
1689   \tl_if_exist:cTF{r@sref@\l_stex_refs_uri_str?\#1}){
1690     \l_stex_refs_do_autoref:n{\l_stex_refs_uri_str?\#1}
1691   }{
1692     \%msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1693     \group_begin:\catcode13=9\relax\catcode10=9\relax
1694       \str_if_empty:NTF \l_stex_key_archive_str {
1695         \prop_if_exist:NTF \l_stex_current_archive_prop {
1696           \str_set:Nx \l_stex_refs_file_str {
1697             \stex_file_use:N \c_stex_mathhub_file /
1698             \prop_item:Nn \l_stex_current_archive_prop { id }
1699             / source / \l_stex_key_file_str .sref
1700           }
1701         }{
1702           \str_set:Nx \l_stex_refs_file_str {
1703             \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str . sref
1704           }
1705         }
1706       }{
1707         \str_set:Nx \l_stex_refs_file_str {
1708           \stex_file_use:N \c_stex_mathhub_file / \l_stex_key_archive_str
1709             / source / \l_stex_key_file_str .sref
1710           }
1711       }
1712     }

```

```

1713     \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_file_str
1714     \str_set:Nx \l__stex_refs_file_str {\stex_file_use:N \l__stex_refs_file }
1715     \stex_debug:nn{sref}{File: \l__stex_refs_file_str }
1716     \exp_args:NNNx \exp_args:No \str_if_eq:nnTF \l__stex_refs_file_str {\stex_file_use:N\l_
1717         \l__stex_refs_do_autoref:n{
1718             \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1719         }
1720     }{
1721         \exp_args:No \IfFileExists \l__stex_refs_file_str {
1722             \tl_clear:N \l__stex_refs_return_tl
1723             \str_set:Nn \l__stex_refs_id_str {#1}
1724             \let\STEXInternalSrefRestoreTarget\l__stex_refs_restore_target:nnnn
1725             \use:c{@ input}{\l__stex_refs_file_str}
1726             \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1727                 \exp_args:Nnno \msg_warning:nnnn{sstex}{warning/smslabelmissing}\l__stex_refs_file_
1728                 \l__stex_refs_do_autoref:n{
1729                     \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1730                 }
1731             }{
1732                 \l__stex_refs_return_tl
1733             }
1734         }{
1735             \exp_args:Nnno \msg_warning:nnnn{sstex}{warning/smsmissing}\l__stex_refs_file_str
1736             \l__stex_refs_do_autoref:n{
1737                 \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1738             }
1739         }
1740     }
1741     \group_end:
1742 }
1743 }
1744
1745 \cs_new_protected:Nn \l__stex_refs_do_return:nnnn {
1746     \tl_set:Nn \l__stex_refs_return_tl {
1747         \stex_annotation:nn{shtml:sref={#4},shtml:srefin={\l__stex_refs_file_str}}{
1748             \use:c{#3autorefname}~#1\tl_if_empty:nF{#2}{~(#2)}
1749             \tl_if_empty:NF\l_stex_key_title_tl{
1750                 {}~in~\l_stex_key_title_tl
1751             }
1752         }
1753     }
1754 }
1755
1756 \cs_new_protected:Nn \l__stex_refs_restore_target:nnnn {
1757     \str_if_empty:NTF \l__stex_refs_uri_str {
1758         \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1759             \l__stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
1760         }
1761     }{
1762         \stex_debug:nn{sref}{\l__stex_refs_uri_str{}~ == ~ #1 ~ ?}
1763         \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1764             \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1765             \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1766                 \stex_debug:nn{sref}{success!}

```

```

1767     \__stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
1768     \endinput
1769 }
1770 }
1771 }
1772 }
```

The actual macros:

```

1773 \NewDocumentCommand \sref { O{} m O{} }{
1774   \stex_keys_set:nn { sref / 1 }{ #1 }
1775   \__stex_refs_find_uri:n { #2 }
1776   \__stex_refs_do_sref:nn{#2}{#3}
1777 }
1778 \NewDocumentCommand \extref { O{} m m }{
1779   \stex_keys_set:nn { sref / 1 }{ #1 }
1780   \__stex_refs_find_uri:n { #2 }
1781   \__stex_refs_set_keys_b:n{ #3 }
1782   \str_if_empty:NT \l_stex_key_file_str {
1783     \msg_error:nn{stex}{error/extrefmissing}
1784   }
1785   \__stex_refs_do_sref_in:n{#2}
1786 }
```

(End of definition for `\sref` and `\extref`. These functions are documented on page 84.)

### \stex\_ref\_new\_sym\_target:n

```

1787 \cs_new_protected:Nn \stex_ref_new_symbol:n {
1788   \cs_if_exist:cF{r@sref@sym@tl_to_str:n{#1}}{
1789     \__stex_refs_new_symbol:n{#1}
1790   }
1791 }
1792
1793 \cs_new_protected:Nn \stex_sref_do_aux:n {
1794   #1 \iow_now:Nn \auxout {#1}
1795 }
1796
1797 \cs_new_protected:Nn \__stex_refs_new_symbol:n {
1798   \prop_if_exist:NTF \l_stex_current_archive_prop {
1799     \prop_get:NnTF \l_stex_current_archive_prop {docurl} \l__stex_refs_str {
1800       \exp_args:Ne \stex_sref_do_aux:n {
1801         \STEXInternalSetSrefSymURL{#1}{\l__stex_refs_str / symbol? #1}
1802       }
1803     }{
1804       \stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
1805     }
1806   }{
1807     \stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
1808   }
1809 }
1810
1811 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1812   \cs_if_exist:NT \hypertarget{
1813     \exp_args:Ne \hypertarget{\tl_to_str:n{sref@sym@ #1}}{}
1814     \str_gset:cx{\tl_to_str:n{r@sref@sym@ #1}}{\tl_to_str:n{sref@sym@ #1}}
1815   }
```

```

1816 }
1817
1818 \cs_new_protected:Nn \stex_ref_new_sym_target:nn {
1819   \str_if_eq:nnTF{#1}{#2} {
1820     \stex_ref_new_sym_target:n{#1}
1821   }{
1822     \str_gset:cn{g_stex_sref_sym_ #1 _label}{#2}
1823   }
1824 }

```

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

### `\srefsym`

```

1825 \NewDocumentCommand \srefsym { m m }{
1826   \stex_get_symbol:n { #1 }
1827   \exp_args:Ne
1828   \__stex_refs_sym_aux:nn{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
1829 }
1830
1831 \cs_new_protected:Npn \__stex_refs_do_internal_link:nn #1 {
1832   \cs_if_exist:NTF \hyperlink {
1833     \hyperlink{#1}
1834   }\use:n
1835 }
1836
1837 \cs_new_protected:Npn \__stex_refs_do_url_link:nn {
1838   \cs_if_exist:NTF \href \href \use_i:nn
1839 }
1840
1841 \cs_new_protected:Npn \__stex_refs_sym_aux:nn #1 {
1842   \cs_if_exist:cTF{\tl_to_str:n{r@sref@sym@#1}}{
1843     \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1844   }{
1845     \str_if_exist:cTF{g_stex_sref_sym_#1_label}{
1846       \exp_args:Ne \__stex_refs_sym_aux:nn{\use:c{g_stex_sref_sym_#1_label}}
1847     }{
1848       \str_if_empty:cTF{g_stex_sref_sym_#1_target}{
1849         \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1850       }{
1851         \exp_args:Ne \__stex_refs_do_url_link:nn{\use:c{g_stex_sref_sym_#1_target}}
1852       }
1853     }
1854   }
1855 }

```

(End of definition for `\srefsym`. This function is documented on page 89.)

### `\srefsymuri`

```

1856 \cs_new_protected:Npn \srefsymuri #1 {
1857   \__stex_refs_sym_aux:nn{#1}
1858 }

```

(End of definition for `\srefsymuri`. This function is documented on page 89.)

### 13.4.4 Inputs

```

1859 <@=stex_inputs>

\stex_resolve_path_pair:Nnn
\stex_resolve_path_pair:Nxx
1860 \cs_new_protected:Nn \stex_resolve_path_pair:Nnn {
1861   \stex_debug:nn{resolving-path}{#3~in-[#2]}
1862   \str_if_empty:nTF{#2} {
1863     \prop_if_exist:NTF \l_stex_current_archive_prop {
1864       \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1865       \prop_item:Nn \l_stex_current_archive_prop { id } / source /
1866       #3}
1867       \stex_debug:nn{resolving-path}{In-current-archive-
1868       \prop_item:Nn \l_stex_current_archive_prop { id }
1869       ;~result:~#1}
1870     }{
1871       \str_set:Nx #1 {\stex_file_use:N \c_stex_pwd_file / .. / #3 }
1872       \stex_debug:nn{resolving-path}{No-current-archive;~result:~#1}
1873     }
1874   }{
1875     \stex_require_archive:n { #2 }
1876     \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1877     \prop_item:cn {c_stex_mathhub_#2_manifest_prop} { id } / source /
1878     #3}
1879     \stex_debug:nn{resolving-path}{result:~#1}
1880   }
1881 }
1882 \cs_generate_variant:Nn \stex_resolve_path_pair:Nnn {Nxx}

```

(End of definition for \stex\_resolve\_path\_pair:Nnn. This function is documented on page 134.)

```

\inputref
\mhinput
\ifinputref
1883 \newif \ifinputref \inputreffalse
1884
1885 \cs_new_protected:Nn \__stex_inputs_mhinput:nn {
1886   \stex_in_archive:nn {#1} {
1887     \ifinputref
1888       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1889     \else
1890       \inputreftrue
1891       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1892       \inputreffalse
1893     \fi
1894   }
1895 }
1896
1897 \NewDocumentCommand \mhinput { O{} m }{
1898   \exp_args:NNx \exp_args:Nnx \__stex_inputs_mhinput:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2} }
1899 }
1900
1901 \cs_new_protected:Nn \__stex_inputs_inputref_html:nn {
1902   \str_clear:N \l_tmpa_str
1903   \prop_get:NnNF \l_stex_current_archive_prop { narr } \l_tmpa_str {
1904     \prop_get:NnNF \l_stex_current_archive_prop { ns } \l_tmpa_str {}
1905   }

```

```

1906 \tl_if_empty:nTF{ #1 }{
1907   \IfFileExists{#2}{
1908     \ifvmode\noindent\fi\stex_annotation_invisible:nn{shtml:inputref=}
1909     \l_tmpa_str / #2
1910   }{}}
1911 }{
1912   \stex_input_with_hooks:n{#2}
1913 }
1914 }{
1915 \IfFileExists{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }{
1916   \ifvmode\noindent\fi\stex_annotation_invisible:nn{shtml:inputref=}
1917   \l_tmpa_str / #2
1918 }{}}
1919 }{
1920   \input{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1921 }
1922 }
1923 }
1924
1925 \cs_new_protected:Nn \__stex_inputs_inputref_pdf:nn {
1926   \begingroup
1927     \inputreftrue
1928     \tl_if_empty:nTF{ #1 }{
1929       \stex_input_with_hooks:n{#2}
1930     }{
1931       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1932     }
1933   \endgroup
1934 }
1935
1936 \cs_new_protected:Nn \__stex_inputs_inputref:nn {
1937   \stex_in_archive:nn {#1} {
1938     \stex_if_html_backend:TF
1939       \__stex_inputs_inputref_html:nn
1940       \__stex_inputs_inputref_pdf:nn
1941       {##1}{#2}
1942   }
1943 }
1944
1945 \NewDocumentCommand \inputref { O{} m }{
1946   \exp_args:NNx \exp_args:Nnx \__stex_inputs_inputref:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2} }
1947 }

(End of definition for \inputref, \mhinput, and \ifinputref. These functions are documented on page 83.)
```

### \addmhbibresource

```

1948 \cs_new_protected:Nn \__stex_inputs_bibresource:n {
1949   \__stex_inputs_up_archive:nn{#1}{bib}
1950   \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
1951     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\addmhbibresource}{#1.bib}
1952   }{
1953     \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
1954       \addbibresource{ ##1 }
```

```

1955     }
1956   }
1957 }
1958 \newcommand\addmhbibresource[2][]{%
1959   \tl_if_empty:nTF{#1}{%
1960     \__stex_inputs_bibresource:n{#2}%
1961   }{%
1962     \stex_in_archive:nn{#1}{\__stex_inputs_bibresource:n{#2}}%
1963   }%
1964 }

```

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

### \IfInputref

```

1965 \stex_if_html_backend:TF{%
1966   \newcommand \IfInputref[2]{%
1967     \stex_annotate:nn{shtml:ifinputref=true}{#1}%
1968     \stex_annotate:nn{shtml:ifinputref=false}{#2}%
1969   }%
1970 }{%
1971   \newcommand \IfInputref[2]{%
1972     \ifinputref #1 \else #2 \fi%
1973   }%
1974 }

```

(End of definition for `\IfInputref`. This function is documented on page 83.)

### \libinput

```

1975 \cs_new_protected:Nn \__stex_inputs_up_archive:nn {%
1976   \prop_if_exist:NF \l_stex_current_archive_prop {%
1977     \msg_error:nnn{stex}{error/notinarchive}\libinput%
1978   }%
1979   \prop_get:NnNF \l_stex_current_archive_prop {id} \l__stex_inputs_id_str {%
1980     \msg_error:nnn{stex}{error/notinarchive}\libinput%
1981   }%
1982   \seq_clear:N \l__stex_inputs_libinput_files_seq%
1983   \seq_set_eq:NN \l__stex_inputs_path_seq \c_stex_mathhub_file%
1984   \seq_set_split:NnV \l__stex_inputs_id_seq / \l__stex_inputs_id_str%
1985 %
1986   \bool_while_do:nn { ! \seq_if_empty_p:N \l_stex_inputs_id_seq }{%
1987     \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / meta-i}%
1988     \IfFileExists{ \l__stex_inputs_path_str }{%
1989       \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_s%
1990       \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str%
1991     }%
1992   }{%
1993     \seq_pop_left:NN \l__stex_inputs_id_seq \l__stex_inputs_path_str%
1994     \seq_put_right:No \l__stex_inputs_path_seq \l__stex_inputs_path_str%
1995   }%
1996 %
1997   \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / lib / #1}%
1998   \IfFileExists{ \l__stex_inputs_path_str }{%
1999     \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_s%
2000     \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str%
2001   }%

```

```

2002     }{}
2003 }
2004
2005 \cs_new_protected:Nn \__stex_inputs_libinput:n {
2006   \__stex_inputs_up_archive:nn{#1}{tex}
2007   \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
2008     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
2009   }{
2010     \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
2011       \input{ ##1 }
2012     }
2013   }
2014 }
2015
2016 \newcommand \libinput [2] [] {
2017   \tl_if_empty:nTF{#1} {
2018     \__stex_inputs_libinput:n{#2}
2019   }{
2020     \stex_in_archive:nn{#1}{\__stex_inputs_libinput:n{#2}}
2021   }
2022 }

```

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

### `\libusepackage`

```

2023 \newcommand\libusepackage[2] [] {
2024   \__stex_inputs_up_archive:nn{#2}{sty}
2025   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2026     \str_set:Nx \l__stex_inputs_tmp_str {\seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
2027     \exp_args:Nne \use:n {\usepackage[#1]} {
2028       \str_range:Nnn\l__stex_inputs_tmp_str 1 {-5}
2029     }
2030   }{
2031     \msg_fatal:nnnn{stex}{error/nofile}{\libusepackage}{#1.sty}
2032   }
2033 }

```

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

```

\mhgraphics
\cmhgraphics
\lstinputmhlisting
\clstinputmhlisting
\mhtikzinput
\cmhtikzinput
2034 \str_new:N \l__stex_inputs_gin_repo_str
2035 \ltx@ifpackageloaded{graphicx}{\use:n}{\AtEndOfPackageFile{graphicx}}{
2036   \define@key{Gin}[archive]{%
2037     \tl_set:Nx\Gin@mrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2038   }
2039   \providecommand\mhgraphics[2][]{%
2040     \tl_set:Nx\Gin@mrepos{%
2041       \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id}
2042     }
2043     \setkeys{Gin}{#1}
2044     \includegraphics[#1]{ \Gin@mrepos #2 }
2045   }
2046   \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
2047 }
2048

```

```

2049 \ltx@ifpackageloaded{listings}{\use:n}{\AtEndOfPackageFile{listings}}{
2050   \define@key{lst}{archive}{
2051     \def\lst@mrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2052   }
2053 \newcommand\lstinputmhlisting[2][]{%
2054   \def\lst@mrepos{
2055     \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id}
2056   }
2057   \setkeys{lst}{#1}%
2058   \lstinputlisting[#1]{\lst@mrepos #2}}
2059 \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
2060 }
2061
2062 \ltx@ifpackageloaded{tikzinput}{\use:n}{\AtEndOfPackageFile{tikzinput}}{
2063   \define@key{Gin}{archive}{
2064     \str_set:Nn \l__stex_inputs_gin_repo_str {#1}
2065     \def\Gin@mrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2066   }
2067 \newcommand\mhtikzinput[2][]{%
2068   \str_clear:N \l__stex_inputs_gin_repo_str
2069   \def\Gin@mrepos{
2070     \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id}
2071   }
2072   \setkeys{Gin}{#1}%
2073   \exp_args:No \stex_in_archive:nn \l__stex_inputs_gin_repo_str {
2074     \tikzinput[#1]{\Gin@mrepos #2}
2075   }
2076 }
2077 \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
2078 }

```

(End of definition for `\mhgraphics` and others. These functions are documented on page 83.)

### `\libusetikzlibrary`

```

2079 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary_i:nn {
2080   \pgfkeys@spdef\pgf@temp{#1}
2081   \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
2082   \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfutil@empty
2083   \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode`@}
2084   \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode`\|}%
2085   \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode`\$}
2086   \catcode`\@=11
2087   \catcode`\|=12
2088   \catcode`\$=3
2089   \pgfutil@InputIfFileExists{#2}{}{%
2090     \catcode`\@=\csname tikz@library@#1@atcode\endcsname
2091     \catcode`\|= \csname tikz@library@#1@barcode\endcsname
2092     \catcode`\$= \csname tikz@library@#1@dollarcode\endcsname
2093   }
2094
2095 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary:n{
2096   \__stex_inputs_up_archive:nn{tikzlibrary#1}{code.tex}
2097   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2098     \exp_args:Nne \__stex_inputs_usetikzlibrary_i:nn{#1}{\seq_item:Nn \l__stex_inputs_libin

```

```

2099 }{
2100   \msg_fatal:nnn{stex}{error/nofile}{\libusetikzlibrary}{tikzlibrary#1.code.tex}
2101 }
2102 }
2103
2104 \newcommand \libusetikzlibrary [2] [] {
2105   \cs_if_exist:N \usetikzlibrary {
2106     \msg_error:nnx{stex}{error/notikz}{\tl_to_str:n{\libusetikzlibrary}}
2107   }
2108   \tl_if_empty:nTF{#1} {
2109     \__stex_inputs_usetikzlibrary:n{#2}
2110   }{
2111     \stex_in_archive:nn{#1}{\__stex_inputs_usetikzlibrary:n{#2}}
2112   }
2113 }

```

(End of definition for `\libusetikzlibrary`. This function is documented on page 118.)

## 13.5 SMS Mode

```
2114 <@=stex_smsmode>
```

Macros and environments allowed in sms mode:

```

2115 \tl_new:N \g__stex_smsmode_allowed_tl
2116 \tl_new:N \g__stex_smsmode_allowed_escape_tl
2117 \seq_new:N \g__stex_smsmode_allowedenvs_seq

```

```
\stex_sms_allow:N
```

```
\stex_sms_allow_escape:N
```

```
\stex_sms_allow_env:n
```

```

2118 \cs_new_protected:Nn \stex_sms_allow:N {
2119   \tl_gput_right:Nn \g__stex_smsmode_allowed_tl {#1}
2120 }
2121
2122 \cs_new_protected:Nn \stex_sms_allow_escape:N {
2123   \tl_gput_right:Nn \g__stex_smsmode_allowed_escape_tl {#1}
2124 }
2125
2126 \cs_new_protected:Nn \stex_sms_allow_env:n {
2127   \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowedenvs_seq {\tl_to_str:n{#1}}
2128 }

```

(End of definition for `\stex_sms_allow:N`, `\stex_sms_allow_escape:N`, and `\stex_sms_allow_env:n`. These functions are documented on page 135.)

Some initial allowed macros:

```

2129 \stex_sms_allow:N \makeatletter
2130 \stex_sms_allow:N \makeatother
2131 \stex_sms_allow:N \ExplSyntaxOn
2132 \stex_sms_allow:N \ExplSyntaxOff
2133 \stex_sms_allow:N \rustexBREAK

```

```
\stex_if_smsmode_p:
```

```
\stex_if_smsmode:TF
```

```

2134 \bool_new:N \g__stex_smsmode_bool
2135 \bool_set_false:N \g__stex_smsmode_bool
2136 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2137   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2138 }

```

(End of definition for \stex\_if\_smsmode:TF. This function is documented on page 135.)

```
\stex_sms_allow_import:Nn
  \stex_sms_allow_import_env:nn
    2139  \tl_new:N \g__stex_smsmode_allowed_import_tl
    2140  \seq_new:N \g__stex_smsmode_allowed_import_env_seq
    2141  \cs_new_protected:Nn \stex_sms_allow_import:Nn {
    2142    \tl_gput_right:Nn \g__stex_smsmode_allowed_import_tl {#1}
    2143    \tl_gset:cn{\tl_to_str:n{#1}~~~smsmode} {#2}
    2144  }
    2145  \cs_new_protected:Nn \stex_sms_allow_import_env:nn {
    2146    \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowed_import_env_seq {\tl_to_str:n{#1}
    2147      \tl_gset:cn{\tl_to_str:n{#1}~~~env~~~smsmode} {#2}
    2148  }
    2149
    2150  \tl_new:N \g_stex_sms_import_code
```

(End of definition for \stex\_sms\_allow\_import:Nn and \stex\_sms\_allow\_import\_env:nn. These functions are documented on page 136.)

```
\stex_file_in_smsmode:nn
\stex_file_in_smsmode:on
  2151  \cs_new_protected:Nn \__stex_smsmode_in_smsmode:n { \stex_suppress_html:n {
  2152    \vbox_set:Nn \l_tmpa_box {
  2153      \bool_set_true:N \g__stex_smsmode_bool
  2154      \bool_set_false:N \stex_html_do_output_bool
  2155      #1
  2156    }
  2157    \%box_clear:N \l_tmpa_box
  2158  } }
  2159
  2160  \quark_new:N \q__stex_smsmode_break
  2161
  2162  \cs_new_protected:Nn \__stex_smsmode_start_smsmode:n {
  2163    \everyeof{\q__stex_smsmode_break\exp_not:N}
  2164    \let\stex_smsmode_do:\__stex_smsmode_smsmode_do:
  2165    \exp_after:wN \exp_after:wN \exp_after:wN
  2166    \stex_smsmode_do:
  2167    \cs:w @ input\cs_end: "#1" \relax
  2168  }
  2169
  2170  \cs_new_protected:Nn \stex_file_in_smsmode:nn {
  2171    \seq_gclear:N \l__stex_smsmode_importmodules_seq
  2172    \seq_gclear:N \l__stex_smsmode_sigmodules_seq
  2173    \tl_clear:N \g_stex_sms_import_code
  2174    \group_begin:
  2175      \let \l_stex_metatheory_uri \c_stex_default_metatheory
  2176      \cs_set:Npn \stex_check_term:n {#1 {}}
  2177      \seq_clear:N \l_stex_all_modules_seq
  2178      \str_clear:N \l_stex_current_module_str
  2179      #2
  2180      \stex_filestack_push:n{#1}
  2181      \__stex_smsmode_in_smsmode:n {
  2182        \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_imports:N
  2183        \tl_map_inline:Nn \g__stex_smsmode_allowed_import_tl {
  2184          \use:c{\tl_to_str:n{##1}~~~smsmode}
```

```

2185     }
2186     \seq_map_inline:Nn \g__stex_smsmode_allowed_import_env_seq {
2187         \use:c{\tl_to_str:n{##1}~env~~~smsmode}
2188     }
2189     \__stex_smsmode_start_smsmode:n{#1}
2190 }
2191 \__stex_smsmode_in_smsmode:n \g_stex_sms_import_code
2192 \__stex_smsmode_in_smsmode:n {
2193     \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_normal:N
2194     \__stex_smsmode_start_smsmode:n{#1}
2195 }
2196 \stex_filestack_pop:
2197 \group_end:
2198 }
2199 \cs_generate_variant:Nn \stex_file_in_smsmode:nn {on}

```

(End of definition for `\stex_file_in_smsmode:nn`. This function is documented on page 135.)

### `\stex_smsmode_do:`

```

2200 \cs_new_protected:Nn \__stex_smsmode_smsmode_do: {
2201     \% \stex_if_smsmode:T {
2202         \__stex_smsmode_do:w
2203     %}
2204 }
2205 \let\stex_smsmode_do:\relax
2206
2207
2208 \cs_new:Nn \__stex_smsmode_check_cs:NNn {
2209     \exp_after:wN\if\exp_after:wN\relax\exp_not:N#3
2210     \exp_after:wN#1\exp_after:wN#3\else
2211     \exp_after:wN#2\fi
2212 }
2213
2214 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2215     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2216         \__stex_smsmode_check_cs:NNn \__stex_smsmode_do_aux:N \__stex_smsmode_do:w { #1 }
2217     }{
2218         \__stex_smsmode_do:w
2219     }
2220 }
2221
2222 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2223     \cs_if_eq:NNF #1 \q__stex_smsmode_break {
2224         \__stex_smsmode_do_aux_curr:N #1
2225     }
2226 }
2227
2228 \cs_new_protected:Nn \__stex_smsmode_do_aux_imports:N {
2229     \% \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~import}
2230     \tl_if_in:NnTF \g__stex_smsmode_allowed_import_tl {#1} {
2231         \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}~in~import}
2232         #1
2233     }{
2234         \cs_if_eq:NNTF \begin{ #1 {

```

```

2235     \__stex_smsemode_check_begin:Nn \g__stex_smsemode_allowed_import_env_seq
2236     }{
2237         \cs_if_eq:NNTF \end #1 {
2238             \__stex_smsemode_check_end:Nn \g__stex_smsemode_allowed_import_env_seq
2239         }{
2240             \__stex_smsemode_do:w
2241         }
2242     }
2243 }
2244 }
2245
2246 \cs_new_protected:Nn \__stex_smsemode_do_aux_normal:N {
2247     % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}-in-sms-mode}
2248     \tl_if_in:NnTF \g__stex_smsemode_allowed_tl {#1} {
2249         \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}}
2250         #1\__stex_smsemode_do:w
2251     }{
2252         \tl_if_in:NnTF \g__stex_smsemode_allowed_escape_tl {#1} {
2253             \stex_debug:nn{sms}{Executing~escaped~\tl_to_str:n{#1}}
2254             #1
2255         }{
2256             \cs_if_eq:NNTF \begin #1 {
2257                 \__stex_smsemode_check_begin:Nn \g__stex_smsemode_allowedenvs_seq
2258             }{
2259                 \cs_if_eq:NNTF \end #1 {
2260                     \__stex_smsemode_check_end:Nn \g__stex_smsemode_allowedenvs_seq
2261                 }{
2262                     \__stex_smsemode_do:w
2263                 }
2264             }
2265         }
2266     }
2267 }
2268
2269 \cs_new_protected:Nn \__stex_smsemode_check_begin:Nn {
2270     % \stex_debug:nn{sms}{Checking~environment~#2}
2271     \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2272         \stex_debug:nn{sms}{Environment~#2}
2273         \begin{#2}
2274     }{
2275         \__stex_smsemode_do:w
2276     }
2277 }
2278 \cs_new_protected:Nn \__stex_smsemode_check_end:Nn {
2279     % \stex_debug:nn{sms}{Checking~end~environment~#2}
2280     \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2281         \stex_debug:nn{sms}{End-Environment~#2}
2282         \end{#2}\__stex_smsemode_do:w
2283     }{
2284         \%str_if_eq:nnTF{#2}{document} \endinput
2285         \__stex_smsemode_do:w
2286     }
2287 }

```

(End of definition for `\stex_smsemode_do`. This function is documented on page 136.)

## 13.6 Modules

### 13.6.1 The smodule-environment

2288 `<@@=stex_modules>`

`\l_stex_current_module_str` The current module:

2289 `\str_new:N \l_stex_current_module_str`

(End of definition for `\l_stex_current_module_str`. This variable is documented on page 122.)

`\l_stex_all_modules_seq` Stores all modules currently in scope

2290 `\seq_new:N \l_stex_all_modules_seq`

(End of definition for `\l_stex_all_modules_seq`. This variable is documented on page 122.)

`\stex_every_module:n`

2291 `<@@=stex_module_setup>`  
2292 `\tl_clear:N \g_stex_every_module_tl {`  
2293 `}`  
2294 `\cs_new_protected:Nn \stex_every_module:n {`  
2295  `\tl_gput_right:Nn \g_stex_every_module_tl { #1 }`  
2296 `}`

(End of definition for `\stex_every_module:n`. This function is documented on page 122.)

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

2297 `\cs_new_protected:Npn \stex_module_setup:n {`  
2298  `\stex_if_in_module:TF \__stex_module_setup_setup_nested:n \__stex_module_setup_setup_top:n`  
2299 `}`  
2300 `\cs_new_protected:Nn \__stex_module_setup_setup_top:n {`  
2301  `\__stex_module_setup_get_uri_str:n{#1}`  
2302  `\stex_debug:nn{module}{Module~URI:~\l__stex_module_setup_ns_str?#1}`  
2303  `\str_if_empty:NTF \l_stex_key_sig_str`  
2304  `\stex_module_setup_top_nosig:n \__stex_module_setup_setup_top_sig:n {\l__stex_module_setu`  
2305  `\stex_metagroup_new:o \l_stex_current_module_str`  
2306  `\g_stex_every_module_tl`  
2307  `\stex_execute_in_module:x {`  
2308  `\stex_do_deprecation:n{#1}`  
2309  `}`  
2310  `\__stex_module_setup_load_meta:`  
2311  `}`  
2312   
2313 `\cs_new_protected:Nn \stex_module_setup_top_nosig:n {`  
2314  `\stex_if_module_exists:nTF{#1}{`  
2315  `\stex_debug:nn{modules}{(already exists)}`  
2316  `}{`  
2317  `\tl_gclear:c{c_stex_module_ #1 _code}`  
2318  `\prop_gclear:c{c_stex_module_ #1 _morphisms_prop }`  
2319  `\prop_gclear:c{c_stex_module_ #1 _symbols_prop }`  
2320  `\prop_gclear:c{c_stex_module_ #1 _notations_prop }`  
2321  `}`  
2322  `\str_set:Nx \l_stex_current_module_str {#1}`  
2323  `\seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str`

```

2325 }
2326
2327 \cs_new_protected:Nn \__stex_module_setup_top_sig:n {
2328   \stex_if_module_exists:nTF{#1} {
2329     \stex_debug:nn{modules}{(already exists)}
2330   }{
2331     \stex_debug:nn{modules}{(needs loading)}
2332     \__stex_module_setup_load_sig:
2333   }
2334 \% \stex_if_smsmode:F { % WHY?
2335   \stex_activate_module:x {
2336     #1
2337   }
2338 %}
2339 \str_set:Nx\l_stex_current_module_str{#1}
2340 }
2341
2342 \cs_new_protected:Nn \__stex_module_setup_load_sig: {
2343   \stex_file_split_off_ext:NN \l__stex_module_setup_sigfile \g_stex_current_file
2344   \stex_file_split_off_lang:NN \l__stex_module_setup_sigfile \l__stex_module_setup_sigfile
2345   \exp_args:Ne \stex_file_in_smsmode:nn {
2346     \stex_file_use:N \l__stex_module_setup_sigfile . \l_stex_key_sig_str . tex
2347   }{}
2348 }
2349
2350 \cs_new_protected:Nn \__stex_module_setup_setup_nested:n {
2351   \exp_after:wN
2352     \__stex_module_setup_split_module:n \l_stex_current_module_str \__stex_module_setup_end:
2353   \stex_debug:nn{module}{Nested~Module~URI:~\l_stex_current_module_str}
2354   \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
2355   \stex_metagroup_new:o \l_stex_current_module_str
2356 }
2357
2358
2359 \cs_new_protected:Nn \__stex_module_setup_get_uri_str:n {
2360   \str_clear:N \l__stex_module_setup_ns_str
2361   \stex_map_uri:Nnnnn \l_stex_current_ns_uri {
2362     \str_set:Nx \l__stex_module_setup_ns_str{##1\c_colon_str/}
2363   }{
2364     \seq_set_split:Nnn \l__stex_module_setup_seq / {##1}
2365     \seq_pop_right:NN \l__stex_module_setup_seq \l__stex_module_setup_seg
2366     \exp_args:No \str_if_eq:nnF \l__stex_module_setup_seg {#1} {
2367       \seq_put_right:No \l__stex_module_setup_seq \l__stex_module_setup_seg
2368     }
2369     \tl_put_right:Nx \l__stex_module_setup_ns_str {\seq_use:Nn \l__stex_module_setup_seq /}
2370   }{}{ }
2371 }
2372
2373 \cs_new_protected:Npn \__stex_module_setup_split_module:n #1#2 \__stex_module_setup_end: #3
2374   \stex_module_setup_top_nosig:n { #1 ? #2 / #3}
2375 }
2376
2377 \bool_new:N \l_stex_in_meta_bool
2378 \bool_set_false:N \l_stex_in_meta_bool

```

```

2379 \cs_new_protected:Nn \__stex_module_setup_load_meta: {
2380   \tl_if_empty:NF \l_stex_metatheory_uri {
2381     \stex_execute_in_module:x{
2382       \stex_pseudogroup_with:nn{\l_stex_in_meta_bool} {
2383         \stex_activate_module:n {\l_stex_uri_use:N \l_stex_metatheory_uri }
2384       }
2385     }
2386   }
2387 }
2388 }
2389
2390 (@@=stex_modules)

```

(End of definition for `\stex_module_setup:n`. This function is documented on page 122.)

### `\stex_close_module:`

```

2391 \cs_new:Nn \stex_close_module: {
2392   \bool_if:NT \c_stex_persist_write_mode_bool \__stex_modules_persist_module:
2393   \stex_debug:nn{module} {
2394     Closing~module~\l_stex_current_module_str^~J
2395     Code:~\expandafter\meaning\csname c_stex_module_\l_stex_current_module_str _code\endcsna
2396     Imports:~\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_str
2397     Declarations:~\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_
2398     Notations:~\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_
2399   }
2400 }
2401
2402 \cs_new_protected:Nn \__stex_modules_persist_module: {
2403   \stex_persist:e {
2404     \__stex_modules_restore_module:nnnn {\l_stex_current_module_str} {
2405       \exp_after:wN \prop_to_keyval:N \cs:w
2406         c_stex_module_\l_stex_current_module_str _morphisms_prop
2407       \cs_end:
2408     }{
2409       \exp_after:wN \prop_to_keyval:N \cs:w
2410         c_stex_module_\l_stex_current_module_str _symbols_prop
2411       \cs_end:
2412     }{
2413       \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
2414       \exp_after:wN \exp_after:wN \exp_after:wN
2415       { \cs:w c_stex_module_\l_stex_current_module_str _code \cs_end: }
2416     }{}
2417     \prop_map_function:cN{c_stex_module_\l_stex_current_module_str _notations_prop}
2418       \__stex_modules_persist_nots_i:nn
2419     \exp_not:N \STEXRestoreNotsEnd {}
2420   }
2421 }
2422
2423 \cs_new_protected:Nn \__stex_modules_restore_module:nnnn {
2424   \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_morphisms_prop}{#2}
2425   \cs_set:Npn \__stex_modules_tl {#3}
2426   \exp_args:Nno \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}\__stex_
2427   \prop_map_inline:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}{%
2428     \stex_ref_new_symbol:n{#1?##1}

```

```

2429 }
2430 \cs_gset:cpn{c_stex_module_ \tl_to_str:n{#1}_code}{#4}
2431 \prop_gclear:c{c_stex_module_ \tl_to_str:n{#1} _notations_prop}
2432 \str_set:Nn \l_stex_modules_restore_mod_str {#1}
2433 \group_begin:
2434   \catcode`_=8\relax
2435   \catcode`:=12\relax
2436   \__stex_modules_restore_nots:n
2437 }
2438
2439 \cs_new:Nn \__stex_modules_persist_nots_i:nn {
2440   \exp_not:n{#2}
2441 }
2442
2443 \quark_new:N \STEXRestoreNotsEnd
2444
2445 \cs_new_protected:Nn \__stex_modules_restore_nots:n {
2446   \__stex_modules_restore_nots_i:n
2447 }
2448
2449 \cs_new_protected:Nn \__stex_modules_restore_nots_i:n {
2450   \tl_if_eq:nnTF{#1}{\STEXRestoreNotsEnd}{
2451     \group_end:
2452   }{
2453     \__stex_modules_restore_nots_ii:nnnn {#1}
2454   }
2455 }
2456
2457 \cs_new_protected:Nn \__stex_modules_restore_nots_ii:nnnn {
2458   \cs_set:Npn \l_stex_modules_tl {{#4}{#5}}
2459   \exp_args:NNe\use:nn\prop_gput:cnn{
2460     {c_stex_module_ \l_stex_modules_restore_mod_str _notations_prop}
2461     {\tl_to_str:n{#1!#2}}{
2462       {\tl_to_str:n{#1}}{\tl_to_str:n{#2}}{#3}
2463       \exp_args:No \exp_not:n \l_stex_modules_tl
2464     }
2465   }
2466   \__stex_modules_restore_nots_i:n
2467 }

```

(End of definition for `\stex_close_module`. This function is documented on page 122.)

`\l_stex_metatheory_uri`

```
2468 \tl_new:N \l_stex_metatheory_uri
```

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

`\setmetatheory`

```

2469 \cs_new_protected:Nn \__stex_modules_set_matatheory:nn {
2470   \group_begin:
2471     \stex_debug:nn{metatheory}{Setting-metatheory~[#1]#2}
2472     \stex_import_module_uri:nn { #1 } { #2 }
2473     \stex_debug:nn{metatheory}{Here:^^J
2474       \l_stex_import_archive_str^^J
2475       \l_stex_import_path_str^^J

```

```

2476     \l_stex_import_name_str^^J
2477   }
2478 \stex_import_require_module:ooo
2479   \l_stex_import_archive_str
2480   \l_stex_import_path_str
2481   \l_stex_import_name_str
2482 \stex_debug:nn{metatheory}{Found:~\l_stex_import_ns_str}
2483 \exp_args:Nne \use:nn {
2484   \group_end: \stex_uri_resolve:Nn \l_stex_metalanguage_uri
2485 }{\l_stex_import_ns_str}
2486 }
2487
2488 \NewDocumentCommand \setmetatheory {O{} m}{
2489   \__stex_modules_set_metalanguage:nn { #1 }{ #2 }
2490   \stex_smsmode_do:
2491 }
2492 \stex_sms_allow_escape:N \setmetatheory

```

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

Keys and key handling:

```

2493 \stex_keys_define:nnnn{smodule}{
2494   \str_clear:N \l_stex_key_sig_str
2495 }{
2496   meta .code:n = {
2497     \str_if_empty:nTF {#1} {
2498       \tl_clear:N \l_stex_metalanguage_uri
2499     }{
2500       \stex_uri_resolve:Nx \l_stex_metalanguage_uri { #1 }
2501     }
2502   },
2503   ns .code:n = {
2504     \stex_uri_resolve:Nx \l_stex_current_ns_uri { #1 }
2505   },
2506   lang .code:n = {
2507     \stex_set_language:n { #1 }
2508   },
2509   sig .str_set_x:N = \l_stex_key_sig_str ,
2510   creators .code:n = {} , % todo ?
2511   contributors .code:n = {} , % todo ?
2512   srccite .code:n = {} % todo ?
2513 }{id, title, style, deprecate}

```

smodule (env.)

```

2514 \stex_new_stylable_env:nnnnnnn {module} {O{} m} {
2515   \stex_keys_set:nn { smodule }{ #1 }
2516   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
2517   \tl_if_empty:NF \thistitle {
2518     \exp_args:No \stexdoctitle \thistitle
2519   }
2520   \exp_args:Nx \stex_module_setup:n { \tl_to_str:n{ #2 } }
2521
2522 \stex_if_do_html:T {
2523   \exp_args:Nne \begin{stex_annotation_env} {
2524     shtml:theory={\l_stex_current_module_str},

```

```

2525     shtml:language={ \l_stex_current_language_str},
2526     shtml:signature={\l_stex_key_sig_str}
2527     \tl_if_empty:NF \l_stex_metatheory_uri {
2528         shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
2529     }
2530 }
2531 \stex_annotation_invisible:n{}
2532 }
2533 \stex_if_smsmode:F {
2534     \str_set_eq:NN \thismoduleuri \l_stex_current_module_str
2535     \tl_set:Nn \thismodulename {#2}
2536     \stex_style_apply:
2537 }
2538 \stex_smsmode_do:
2539 }{
2540     \stex_close_module:
2541     \stex_if_smsmode:F \stex_style_apply:
2542     \stex_if_do_html:T{ \end{stex_annotation_env} }
2543 }{}{}{s}
2544
2545 \stex_sms_allow_env:n{smodule}

```

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

**\stex\_if\_in\_module:TF**

```

2546 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
2547     \str_if_empty:NTF \l_stex_current_module_str
2548         \prg_return_false: \prg_return_true:
2549 }

```

(End of definition for `\stex_if_in_module:TF`. This function is documented on page 122.)

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

**\stex\_if\_module\_exists:nTF**

```

2550 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
2551     \tl_if_exist:cTF { c_stex_module_#1_code }
2552         \prg_return_true: \prg_return_false:
2553 }

```

(End of definition for `\stex_if_module_exists:nTF`. This function is documented on page 122.)

**\stex\_do\_up\_to\_module:n** Execute code in the current module (i.e. as if the `\begin{smodule}`) was the current tex group)

```

2554 \cs_new_protected:Nn \stex_do_up_to_module:n {
2555     \exp_args:No \stex_metagroup_do_in:nn \l_stex_current_module_str {#1}
2556 }
2557 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}

```

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

**\stex\_module\_add\_code:n**

**\stex\_module\_add\_code:x**

```

2558 \cs_new_protected:Nn \stex_module_add_code:n {
2559     \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
2560 }
2561 \cs_generate_variant:Nn \stex_module_add_code:n {x}

```

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

```

\stex_execute_in_module:n
\stex_execute_in_module:x 2562 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:TF {
2563   \stex_module_add_code:n { #1 }
2564   \stex_do_up_to_module:n { #1 }
2565 }{ #1 } }
2566 \cs_generate_variant:Nn \stex_execute_in_module:n {x}

```

(End of definition for `\stex_execute_in_module:n`. This function is documented on page 122.)

### \STEXexport

```

2567 \NewDocumentCommand \STEXexport {} {
2568   \ExplSyntaxOn
2569   \__stex_modules_export:n
2570 }
2571 \cs_new_protected:Nn \__stex_modules_export:n {
2572   \stex_ignore_spaces_and_pars:#1\ExplSyntaxOff
2573   \stex_module_add_code:n { \stex_ignore_spaces_and_pars:#1}
2574   \stex_smsmode_do:
2575 }

```

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

```

2576 \stex_deactivate_macro:Nn \STEXexport {module~environments}
2577 \stex_sms_allow_escape:N \STEXexport
2578 \stex_every_module:n {\stex_reactivate_macro:N \STEXexport}

```

(End of definition for `\STEXexport`. This function is documented on page 86.)

### \stex\_module\_add\_morphism:nnnn

```

\stex_module_add_morphism:nonn
\stex_module_add_morphism:ooox
2579 \cs_new_protected:Nn \stex_module_add_morphism:nnnn {
2580   \exp_args:Nne \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _morphisms_prop}{%
2581     \tl_if_empty:nTF{#1}{[#2]}{#1}
2582   }{#1}{#2}{#3}{#4}
2583 }
2584 \cs_generate_variant:Nn \stex_module_add_morphism:nnnn {nonn,ooox}

```

(End of definition for `\stex_module_add_morphism:nnnn`. This function is documented on page 130.)

### \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

2585 \cs_new_protected:Nn \stex_module_add_symbol:nnnnnnN {
2586   \stex_debug:nn{declaration}{New~declaration:~\l_stex_current_module_str?#2^^J
2587     Macro:#1^^JArity:#3~(#4)^^^J
2588     Def:~\tl_to_str:n{#5}^^J
2589     Type:~\tl_to_str:n{#6}^^J
2590     Returns:~\tl_to_str:n{#7}
2591   }
2592   \%prop_gput:cnx{c_stex_module_\l_stex_current_module_str _symbols_prop}
2593   %{#2}{\exp_not:n{#1}{#2}{#3}{#4}{#5}{#6}{#7}\exp_not:n{#8}}

```

```

2594 \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
2595 {##2}{##1}{##2}{##3}{##4}{##5}{##6}{##7}{##8}}
2596 \tl_if_empty:nF{#1} {
2597   \stex_execute_in_module:n {
2598     \__stex_modules_activate_sym:n {#2}
2599   }
2600 }
2601 }
2602
2603 \cs_new_protected:Nn \__stex_modules_activate_sym:n {
2604   \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _symbols_prop}{%
2605     \str_if_eq:nnT{#1}{##1}{%
2606       \__stex_modules_activate_i:nnnnnnnn ##2
2607     }
2608   }
2609 }
2610 \cs_new_protected:Nn \__stex_modules_activate_i:nnnnnnnn {
2611   \stex_debug:nn{activating}{#1:\l_stex_current_module_str^~J}
2612   \tl_to_str:n{##2}{##3}{##4}{##5}{##6}{##7}{##8}
2613 }
2614 \cs_set:cp{#1} {
2615   \stex_invoke_symbol:nnnnnnnN
2616   {\l_stex_current_module_str}
2617   \exp_not:n{##2}{##3}{##4}{##5}{##6}{##7}{##8}
2618 }
2619 \stex_debug:nn{activating}{done}
2620 \prop_map_break:
2621 }

```

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

```

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

2622 \cs_new_protected:Nn \stex_module_add_notation:nnnnn {
2623   \stex_debug:nn{notations}{Adding~notation:~^~J}
2624   #1~\c_hash_str#2~#3~^~J
2625   to~\l_stex_current_module_str
2626 }
2627 \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _notations_prop}
2628 {##1!##2}{##1}{##2}{##3}{##4}{##5}
2629 \stex_execute_in_module:n {
2630   \__stex_modules_activate_not:nn{##1}{##2}
2631 }
2632 }
2633 \cs_generate_variant:Nn \stex_module_add_notation:nnnnn {eoeoo}
2634
2635
2636 \cs_new_protected:Nn \__stex_modules_activate_not:nn {
2637   \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _notations_prop}{%
2638     \str_if_eq:nnT{##1!##2}{##1}{%
2639       \prop_map_break:n{\stex_set_notation_macro:nnnnn ##2 } }
```

```

2640     }
2641   }
2642 }
```

(End of definition for `\stex_module_add_notation:nnnnn` and `\stex_set_notation_macro:nnnnn`. These functions are documented on page 125.)

```
\stex_set_notation_macro:nnnnn
\stex_set_notation_macro:eoexo
2643 \cs_new_protected:Nn \stex_set_notation_macro:nnnnn {
2644   \tl_set:cn {l_stex_notation_#1#2_cs}{#4}
2645   \cs_if_exist:cF{l_stex_notation_#1__cs} {
2646     \tl_set:cn {l_stex_notation_#1__cs}{#4}
2647   }
2648   \tl_if_empty:nF{#5} {
2649     \tl_set:cn{l_stex_notation_#1_op_#2_cs}{#5}
2650     \cs_if_exist:cF{l_stex_notation_#1_op__cs} {
2651       \cs_set_eq:cc {l_stex_notation_#1_op__cs}{l_stex_notation_#1_op_#2_cs}
2652     }
2653   }
2654 }
2655 \cs_generate_variant:Nn \stex_set_notation_macro:nnnnn {eoexo}
```

(End of definition for `\stex_set_notation_macro:nnnnn`. This function is documented on page 126.)

```
\stex_activate_module:n
\stex_activate_module:o
\stex_activate_module:x
2656 \cs_new_protected:Nn \stex_activate_module:n {
2657   \seq_if_in:Nnf \l_stex_all_modules_seq { #1 } {
2658     \stex_debug:nnf{modules}{Activating-module~#1^J\expandafter\meaning\csname c_stex_module
2659     \seq_put_right:Nn \l_stex_all_modules_seq { #1 }
2660     \stex_pseudogroup:nn{
2661       \str_set:Nn \l_stex_current_module_str {#1}
2662       \use:c{ c_stex_module_#1_code }
2663     }{
2664       \stex_pseudogroup_restore:N \l_stex_current_module_str
2665     }
2666   }
2667 }
2668 \cs_generate_variant:Nn \stex_activate_module:n {o,x}
```

(End of definition for `\stex_activate_module:n`. This function is documented on page 122.)

Iterating:

```
2669 <@=stex_iterate>
```

`\stex_iterate_symbols:n`

```
2670 \cs_new_protected:Nn \stex_iterate_symbols:n {
2671   \stex_pseudogroup_with:nn{\_\stex_iterate_sym_cs:nnnnnnnnN\stex_iterate_break:\stex_iterat
2672   \cs_set:Npn \_\stex_iterate_sym_cs:nnnnnnnnN
2673   ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
2674   \cs_set:Npn \stex_iterate_break: {
2675     \prop_map_break:n{\seq_map_break:}
2676   }
2677   \cs_set:Npn \stex_iterate_break:n ##1 {
2678     \prop_map_break:n{\seq_map_break:n{##1}}
2679 }
```

```

2680     \seq_map_inline:Nn \l_stex_all_modules_seq {
2681         \prop_map_inline:cn{c_stex_module_##1_symbols_prop}{
2682             \__stex_iterate_sym_cs:nnnnnnnnN {##1} ####2
2683         }
2684     }
2685 }
2686 }
```

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

### \stex\_iterate\_symbols:nn

```

2687 \cs_new_protected:Nn \stex_iterate_symbols:nn {
2688     \seq_clear:N \l_stex_iterate_mods_seq
2689     \stex_pseudogroup_with:nn{\__stex_iterate_sym_cs:nnnnnnnnN}{
2690         \cs_set:Npn \__stex_iterate_sym_cs:nnnnnnnnN
2691             ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #2 }
2692         \clist_map_function:nN {#1} \__stex_iterate_it_decl_i:n
2693     }
2694 }
2695
2696 \cs_new_protected:Nn \__stex_iterate_it_decl_i:n {
2697     \seq_if_in:NnF \l_stex_iterate_mods_seq {#1} {
2698         \seq_put_left:Nn \l_stex_iterate_mods_seq {#1}
2699         \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2700             \__stex_iterate_it_decl_check:nnnn ##2
2701         }
2702         \prop_map_inline:cn{c_stex_module_#1_symbols_prop}{
2703             \__stex_iterate_sym_cs:nnnnnnnnN {##1} ##2
2704         }
2705     }
2706 }
2707 \cs_new_protected:Nn \__stex_iterate_it_decl_check:nnnn {
2708     \tl_if_empty:nT{#1}{%
2709         \__stex_iterate_it_decl_i:n {#2}
2710     }
2711 }
```

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

### \stex\_iterate\_notations:nn

```

2712 \cs_new_protected:Nn \stex_iterate_notations:nn {
2713     \seq_clear:N \l_stex_iterate_mods_seq
2714     \stex_pseudogroup_with:nn{\__stex_iterate_not_cs:nnnn}{%
2715         \cs_set:Npn \__stex_iterate_not_cs:nnnn
2716             ##1 ##2 ##3 ##4 ##5 { #2 }
2717         \clist_map_function:nN {#1} \__stex_iterate_it_not_i:n
2718     }
2719 }
2720
2721 \cs_new_protected:Nn \__stex_iterate_it_not_i:n {
2722     \seq_if_in:NnF \l_stex_iterate_mods_seq {#1} {
2723         \seq_put_left:Nn \l_stex_iterate_mods_seq {#1}
2724         \prop_map_inline:cn{c_stex_module_#1_notations_prop}{
2725             \__stex_iterate_not_cs:nnnn ##2
2726         }
2727 }
```

```

2727   \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2728     \__stex_iterate_it_not_check:nnnn ##2
2729   }
2730 }
2731 }
2732 \cs_new_protected:Nn \__stex_iterate_it_not_check:nnnn {
2733   \tl_if_empty:nT{#1}{%
2734     \__stex_iterate_it_not_i:n {#2}
2735   }
2736 }

```

(End of definition for `\stex_iterate_notations:nn`. This function is documented on page 126.)

### `\stex_iterate_morphisms:nn`

```

2737 \cs_new_protected:Nn \stex_iterate_morphisms:nn {
2738   \seq_clear:N \l__stex_iterate_mods_seq
2739   \bool_set_true:N \l__stex_iterate_continue_bool
2740   \cs_set:Npn \__stex_iterate_morphism_cs:nnnn ##1 ##2 ##3 ##4 ##5 {
2741     #2
2742     \bool_if:NT \l__stex_iterate_continue_bool {
2743       \str_if_eq:nnTF{##1}{##2}{%
2744         \tl_put_right:Nn \l__stex_iterate_todo_tl {
2745           \__stex_iterate_iterate_morphism:nn{##5}{##2}
2746         }
2747       }{
2748         \tl_put_right:Nn \l__stex_iterate_todo_tl {
2749           \__stex_iterate_iterate_morphism:nn{##5 / ##1}{##2}
2750         }
2751       }
2752     }
2753   }
2754   \cs_set:Npn \stex_iterate_break:n ##1 {
2755     \bool_set_false:N \l__stex_iterate_continue_bool
2756     \prop_map_break:n{##1}
2757   }
2758   \__stex_iterate_iterate_morphism:nn{}{##1}
2759 }
2760
2761 \cs_new_protected:Nn \__stex_iterate_iterate_morphism:nn {
2762   \tl_clear:N \l__stex_iterate_todo_tl
2763   \seq_if_in:NnF \l__stex_iterate_mods_seq {##1 ##2}{%
2764     \seq_put_right:Nn \l__stex_iterate_mods_seq {##1 ##2}
2765     \prop_map_inline:cn{c_stex_module_#2_morphisms_prop}{
2766       \__stex_iterate_morphism_cs:nnnn ##2 {##1}
2767       % TODO
2768       % ##1: name or [mpath]
2769       % ##2 = {####1}{####2}{####3}{####4}
2770       % ####1 = name
2771       % ####2 = mpath
2772       % ####3 = type
2773       % ####4 = {origname}{newname}*
2774     }
2775     \bool_if:NT \l__stex_iterate_continue_bool \l__stex_iterate_todo_tl
2776   }
2777 }

```

(End of definition for \stex\_iterate\_morphisms:nn. This function is documented on page ??.)

### 13.6.2 Structural Features

2778 `<@@=stex_features>`

```
\stex_structural_feature_module:nn
\stex_structural_feature_module_end:
2779 \cs_new_protected:Nn \stex_structural_feature_module:nn {
2780   \stex_if_do_html:TF {
2781     \exp_args:Nne \begin{stex_annotation_env} {
2782       \shtml:feature-#2={
2783         \l_stex_current_module_str/#1}
2784       \str_if_empty:NF \l_stex_mroname_str {
2785         \shtml:macroname={\l_stex_mroname_str}
2786       }
2787     }
2788     \stex_annotation_invisible:n{}
2789   }\group_begin:
2790   \stex_module_setup:n {#1-module}
2791 }
2792
2793 \cs_new_protected:Nn \stex_structural_feature_module_end: {
2794   \tl_gset_eq:NN \g_stex_last_feature_str \l_stex_current_module_str
2795   \stex_close_module:
2796   \stex_if_do_html:TF{
2797     \end{stex_annotation_env}
2798   }\group_end:
2799 }
```

(End of definition for \stex\_structural\_feature\_module:nn and \stex\_structural\_feature\_module\_end:. These functions are documented on page 128.)

\stex\_structural\_feature\_morphism:nnn

\stex\_structural\_feature\_morphism\_end:

```
2800 \bool_new:N \l__stex_features_implicit_bool
2801 \cs_new_protected:Nn \stex_structural_feature_morphism:nnnnn {
2802   \str_clear:N \l_stex_current_domain_str
2803   \tl_if_empty:nT{#3} {
2804     \l_stex_get_mathstructure:n{#4}
2805     \str_set_eq:NN \l_stex_current_domain_str \l_stex_get_structure_module_str
2806   }
2807   \str_if_empty:NT \l_stex_current_domain_str {
2808     \stex_import_module_uri:nn { #3 }{ #4 }
2809     \group_begin:
2810     \stex_import_require_module:ooo
2811       \l_stex_import_archive_str
2812       \l_stex_import_path_str
2813       \l_stex_import_name_str
2814     \exp_args:Nnx \use:nn \group_end: {
2815       \str_set:Nn \exp_not:N\l_stex_current_domain_str {\l_stex_import_ns_str}
2816     }
2817   }
2818   \tl_if_empty:nTF{#1} {
2819     \bool_set_true:N \l__stex_features_implicit_bool
2820     \str_set:Nx \l_tmpa_str {[ \l_stex_current_domain_str ] }
2821   }
}
```

```

2822   \bool_set_false:N \l__stex_features_implicit_bool
2823   \str_set:Nn \l_tmpa_str {#1}
2824 }
2825
2826 \stex_if_do_html:TF {
2827   \begin{stex_annotation_env} {
2828     \shtml:feature-#2={\l_stex_current_module_str?\l_tmpa_str},
2829     \shtml:domain={\l_stex_current_domain_str}
2830     #5
2831   }
2832   \stex_annotation_invisible:n{}
2833 } \group_begin:
2834 \str_set:Nn \l__stex_features_feature_str {#2}
2835 \str_set_eq:NN \l_stex_feature_name_str \l_tmpa_str
2836 \__stex_features_setup:
2837 \__stex_features_reactivate:
2838 %^A\stex_activate_module:o \l_stex_current_domain_str
2839 \exp_args:Ne \stex_metagroup_new:n {\l_stex_current_module_str / \l_stex_feature_name_str}
2840 }
2841
2842 \cs_new_protected:Nn \__stex_features_do_for_list: {
2843   \seq_clear:N \l_stex_fors_seq
2844   \clist_map_inline:Nn \l_stex_key_for_clist {
2845     \exp_args:Ne \stex_get_in_morphism:n{\tl_to_str:n{##1}}
2846     \seq_put_right:Nx \l_stex_fors_seq
2847     {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
2848   }
2849 }
2850
2851 \cs_new_protected:Nn \__stex_features_add_definiens:nn {
2852   \__stex_features_set_definiens_macros: #1\__stex_features_break:
2853   \stex_assign_do:n{#2}
2854   #2
2855 }
2856 \cs_new_protected:Npn \__stex_features_set_definiens_macros: #1?#2?#3\__stex_features_break:
2857   \str_set:Nn \l_stex_get_symbol_mod_str {#1?#2}
2858   \str_set:Nn \l_stex_get_symbol_name_str {#3}
2859   \exp_args:Nne \use:nn{\__stex_features_set_definiens_macros_i:nnnnnnn}{%
2860     \prop_item:Nn \l_stex_morphism_symbols_prop {[#1?#2]/[#3]}}
2861   }
2862 }
2863 \cs_new_protected:Nn \__stex_features_set_definiens_macros_i:nnnnnnn {
2864   \tl_set:Nn \l_stex_get_symbol_def_tl{#4}
2865 }
2866
2867 \cs_new_protected:Nn \stex_structural_feature_morphism_end: {
2868   \str_gset_eq:NN \l_stex_feature_name_str \l_stex_feature_name_str
2869   \str_gset_eq:NN \l_stex_current_domain_str \l_stex_current_domain_str
2870   \seq_gset_eq:NN \l_stex_morphism_symbols_prop \l_stex_morphism_symbols_prop
2871   \seq_gset_eq:NN \l_stex_morphism_renames_prop \l_stex_morphism_renames_prop
2872   \seq_gset_eq:NN \l_stex_morphism_morphisms_seq \l_stex_morphism_morphisms_seq
2873   \__stex_features_do_elaboration:
2874   \stex_if_do_html:TF{
2875     \end{stex_annotation_env}

```

```

2876   }\group_end:
2877 }
2878
2879 \cs_new_protected:Nn \__stex_features_setup: {
2880   \prop_clear:N \l_stex_morphism_symbols_prop
2881   \prop_clear:N \l_stex_morphism_renames_prop
2882   \seq_clear:N \l_stex_morphism_morphisms_seq
2883   \__stex_features_do_decls:
2884   \exp_args:No \__stex_features_do_morphisms:n \l_stex_current_domain_str
2885 }
2886
2887 \cs_new_protected:Nn \__stex_features_rename_all: {
2888 }
2889
2890
2891 \cs_new:Npn \__stex_features_clean:nnw [#1] / #2 \_stex_end: {
2892   [#1]/[#2]
2893 }
2894
2895 \cs_new_protected:Nn \__stex_features_do_decls: {
2896   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_domain_str {
2897     \stex_if_starts_with:nnTF{##3}[{
2898       \exp_args:NNe \prop_put:Nnn \l_stex_morphism_symbols_prop {
2899         \__stex_features_clean:nnw ##3 \_stex_end:
2900       }
2901     }{
2902       \prop_put:Nnn \l_stex_morphism_symbols_prop
2903         {[##1]/[##3]}
2904     }{
2905       {##2}{##4}{##5}{##6}{##7}{##8}##9
2906     }
2907   }
2908 }
2909
2910 \cs_new_protected:Nn \stex_structural_feature_morphism_check_total: {
2911   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2912     \__stex_features_total_check: ##1 ##2
2913   }
2914 }
2915
2916 \cs_new_protected:Npn \__stex_features_total_check: [#1]/[#2] #3 #4 #5 #6 #7 #8 #9 {
2917   \tl_if_empty:nT{#6}{%
2918     \msg_error:nnxx{\stex}{error/needsdefiniens}{#1?#2}{total-morphism}
2919   }
2920 }
2921
2922 \cs_new:Npn \__stex_features_split_qm:w #1 ? #2 ? #3 { #3 }
2923 \cs_new_protected:Nn \__stex_features_do_elaboration: {
2924   \stex_debug:nn{morphisms}{%
2925     Elaborating:^^J\prop_to_keyval:N \l_stex_morphism_symbols_prop
2926     ^^J
2927     Renamings:^^J
2928     \prop_to_keyval:N \l_stex_morphism_renames_prop
2929   }

```

```

2930 \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2931   \__stex_features_elab_check: ##1 ##2
2932 }
2933 \exp_args:No\stex_iterate_notations:nn\l_stex_current_domain_str{
2934   \prop_get:NnNTF \l_stex_morphism_renames_prop {##1}\l_stex_features_tmp {
2935     \exp_args:Ne \stex_module_add_notation:nnnnn
2936     {\l_stex_current_module_str ? \exp_after:wN \use_i:nn \l_stex_features_tmp}
2937   }{
2938     \exp_args:Ne \stex_module_add_notation:nnnnn
2939     {\l_stex_current_module_str ? \l_stex_feature_name_str
2940       / \__stex_features_split_qm:w ##1}
2941   }{##2}{##3}{##4}{##5}
2942 }
2943 \stex_module_add_morphism:ooox
2944   \l_stex_feature_name_str
2945   \l_stex_current_domain_str
2946   \l_stex_features_feature_str
2947   {\prop_map_function:NN \l_stex_morphism_renames_prop \__stex_features_rename:nn}
2948 }
2949 \cs_new:Nn \__stex_features_rename:nn{
2950   {#1}{\use_i:nn#2}
2951 }
2952 }
2953 \cs_new_protected:Npn \__stex_features_elab_check: [#1]/[#2] #3 {
2954   \prop_get:NnNTF \l_stex_morphism_renames_prop {#1##2} \l_stex_features_tmp {
2955     \stex_debug:nn{morphisms}{Generating~\l_stex_features_tmp}
2956     \exp_after:wN \stex_module_add_symbol:nnnnnnnnN \l_stex_features_tmp
2957   }{
2958     \bool_if:NTF \l_stex_features_implicit_bool {
2959       \stex_debug:nn{morphisms}{Generating~#3:~\l_stex_feature_name_str / #2}
2960       \exp_args:Nno \stex_module_add_symbol:nnnnnnnnN {#3}{\l_stex_feature_name_str / #2}
2961     }{
2962       \stex_debug:nn{morphisms}{Generating~\l_stex_feature_name_str / #2}
2963       \exp_args:Nno \stex_module_add_symbol:nnnnnnnnN {}{\l_stex_feature_name_str / #2}
2964     }
2965   }
2966 }
2967 }
2968 \cs_new_protected:Nn \__stex_features_do_morphisms:n {
2969   \prop_map_inline:cn {c_stex_module_#1_morphisms_prop} {
2970     \__stex_features_do_morph:nnnn ##2
2971   }
2972 }
2973 }
2974 \cs_new_protected:Nn \__stex_features_do_morph:nnnn {
2975   \tl_if_empty:nF{#3} {
2976     \seq_put_right:Nn \l_stex_morphism_morphisms_seq {{#1}{#2}{#3}}
2977   }
2978   \__stex_features_do_morphisms:n{#2}
2979 }
2980 }
2981 \cs_new_protected:Npn \__stex_features_reactivate: {
2982   \stex_deactivate_macro:Nn \symdecl {module~environments}

```

```

2984 \stex_deactivate_macro:Nn \textsymdecl {module-environments}
2985 \stex_deactivate_macro:Nn \symdef {module-environments}
2986 \stex_deactivate_macro:Nn \notation {module-environments}
2987 \stex_deactivate_macro:Nn \importmodule {module-environments}
2988 \stex_deactivate_macro:Nn \requiremodule {module-environments}
2989 \stex_deactivate_macro:Nn \smodule {outside-of-morphisms}
2990 \stex_reactivate_macro:N \assign
2991 \stex_reactivate_macro:N \assignMorphism
2992 \stex_reactivate_macro:N \renamedecl
2993 \cs_set_eq:NN \stex_do_for_list: \stex_features_do_for_list:
2994 \cs_set_eq:NN \stex_add_definiens:nn \stex_features_add_definiens:nn
2995 }

```

(End of definition for \stex\_structural\_feature\_morphism:nnn and \stex\_structural\_feature\_morphism\_end:. These functions are documented on page ??.)

### \stex\_get\_in\_morphism:n

```

2996 \cs_new_protected:Nn \stex_get_in_morphism:n {
2997   \str_clear:N \l_stex_get_symbol_name_str
2998   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2999     \exp_args:Nx \stex_features_get_check:nnnn{\tl_to_str:n{#1}}##1##2
3000   }
3001   \str_if_empty:NT \l_stex_get_symbol_name_str {
3002     \prop_map_inline:Nn \l_stex_morphism_renames_prop {
3003       \stex_features_renamed_check:nnnnn{#1}##1=##2
3004     }
3005     \str_if_empty:NT \l_stex_get_symbol_name_str {
3006       \msg_error:nnxx{\stex}{error/unknownsymbolin}{#1}{
3007         morphism-\l_stex_feature_name_str
3008       }
3009     }
3010   }
3011 }
3012
3013 \cs_new_protected:Npn \stex_features_renamed_check:nnnnn #1##2##3##4##5##6 {
3014   \str_if_eq:nnTF{#1}{#5} {
3015     \exp_args:Nnx \use:nn{\stex_features_check_break:nnnnnnnn{#2##3}{#4}}{
3016       \prop_item:Nn \l_stex_morphism_symbols_prop {[#2##3]/[#4]}
3017     }
3018   }{
3019     \str_if_eq:nnT{#1}{#6} {
3020       \exp_args:Nnx \use:nn{\stex_features_check_break:nnnnnnnn{#2##3}{#4}}{
3021         \prop_item:Nn \l_stex_morphism_symbols_prop {[#2##3]/[#4]}
3022       }
3023     }
3024   }
3025 }
3026
3027 \cs_new_protected:Npn \stex_features_get_check:nnnn #1[#2]/[#3]##4 {
3028   \str_if_eq:nnTF{#1}{#3} {
3029     \stex_features_check_break:nnnnnnnn{#2}{#3}{#4}
3030   }{
3031     \str_if_eq:nnTF{#1}{#4} {
3032       \stex_features_check_break:nnnnnnnn{#2}{#3}{#4}

```

```

3033     }{
3034         \use_none:nnnnnn
3035     }
3036 }
3037 }
3038
3039 \cs_new_protected:Nn \__stex_features_check_break:nnnnnnnn {
3040     \prop_map_break:n{
3041         \str_set:Nn \l_stex_get_symbol_mod_str{\#1}
3042         \str_set:Nn \l_stex_get_symbol_name_str{\#2}
3043         \str_set:Nn \l_stex_get_symbol_macro_str{\#3}
3044         \int_set:Nn \l_stex_get_symbol_arity_int {\#4}
3045         \tl_set:Nn \l_stex_get_symbol_args_tl {\#5}
3046         \tl_set:Nn \l_stex_get_symbol_def_tl {\#6}
3047         \tl_set:Nn \l_stex_get_symbol_type_tl {\#7}
3048         \tl_set:Nn \l_stex_get_symbol_return_tl {\#8}
3049         \tl_set:Nn \l_stex_get_symbol_invoke_cs {\#9}
3050     }
3051 }

```

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

## 13.7 Inheritance

### 13.7.1 \importmodule/\usemodule

```

3052 <@=stex_importmodule>
\usemodule
3053 \stex_new_styable_cmd:nnnn {usemodule} { 0{} m } {
3054     \stex_import_module_uri:nn { #1 }{ #2 }
3055     \stex_import_require_module:ooo
3056         \l_stex_import_archive_str
3057         \l_stex_import_path_str
3058         \l_stex_import_name_str
3059     \stex_if_do_html:T {
3060         \hbox{\stex_annotation_invisible:nn
3061             {shtml:usemodule=\l_stex_import_ns_str} {}}
3062     }
3063     \stex_if_smsmode:F{
3064         \group_begin:
3065         \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3066         \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3067         \tl_clear:N \thisstyle
3068         \stex_style_apply:
3069         \group_end:
3070     }
3071 }{}

```

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

```
\stex_import_module_uri:nn
3072 \cs_new_protected:Nn \stex_import_module_uri:nn {
3073     \stex_debug:nn{importmodule}{URI:~>#1<~>#2<}
```

```

3074 \exp_args:NNnx \seq_set_split:Nnn \__stex_importmodule_seq ? { \tl_to_str:n{ #2 } }
3075 \seq_pop_right:NN \__stex_importmodule_seq \l_stex_import_name_str
3076 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \__stex_importmodule_seq ? }
3077 \tl_if_empty:nTF { #1 } {
3078   \stex_debug:nn{importmodule}{No~archive}
3079   \prop_if_exist:NTF \l_stex_current_archive_prop {
3080     \stex_debug:nn{importmodule}{Picking~current~archive}
3081     \str_set:Nx \l_stex_import_archive_str {
3082       \prop_item:Nn \l_stex_current_archive_prop { id }
3083     }
3084   }{
3085     \str_clear:N \l_stex_import_archive_str
3086     \str_set:Nn \l_stex_import_uri_str {file:}
3087     \str_if_empty:NTF \l_stex_import_path_str {
3088       \stex_debug:nn{importmodule}{Empty~Path}
3089       \stex_file_split_off_ext:NN \l__stex_importmodule_path_seq \g_stex_current_file
3090       \stex_file_split_off_lang:NN \l__stex_importmodule_path_seq \l__stex_importmodule_pa
3091       \str_set:Nx \l_stex_import_path_str {
3092         \stex_file_use:N \l__stex_importmodule_path_seq
3093       }
3094     }{
3095       \stex_debug:nn{importmodule}{Resolving~path~\l_stex_import_path_str~relative~to~\ste
3096       \stex_file_resolve:Nx \l__stex_importmodule_seq { \stex_file_use:N \g_stex_current_f
3097       \str_set:Nx \l_stex_import_path_str {
3098         \stex_file_use:N \l__stex_importmodule_seq
3099       }
3100       \stex_debug:nn{importmodule}{...yields~\l_stex_import_path_str}
3101     }
3102   }
3103 }{
3104   \stex_debug:nn{importmodule}{Archive~#1}
3105   \str_set:Nx \l_stex_import_archive_str { #1 }
3106   \stex_require_archive:o \l_stex_import_archive_str
3107   \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ \l_stex_import_archi
3108 }
3109 }

```

(End of definition for `\stex_import_module:nn`. This function is documented on page 130.)

```

\stex_import_require_module:nnn
\stex_import_require_module:ooo
3110 \cs_new_protected:Npn \stex_import_require_module:nnn #1 {
3111   \tl_if_empty:nTF { #1 } {
3112     \str_clear:N \l__stex_importmodule_archive_str
3113     \str_set:Nn \l_stex_import_uri_str {file:}
3114     \__stex_importmodule_get_module:nnn {}
3115   }{
3116     \stex_require_archive:n { #1 }
3117     \str_set:Nx \l__stex_importmodule_archive_str {#1}
3118     \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ #1 _manifest_prop}{}
3119     \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
3120     \exp_args:No \__stex_importmodule_get_module:nnn \l__stex_importmodule_str
3121   }
3122 }
3123 \cs_generate_variant:Nn \stex_import_require_module:nnn {ooo}

```

```

3124 \cs_new_protected:Npn \stex_import_require_module_safe:nnn #1 {
3125   \tl_if_empty:nTF { #1 } {
3126     \str_clear:N \l__stex_importmodule_archive_str
3127     \str_set:Nn \l_stex_import_uri_str {file:}
3128     \__stex_importmodule_get_module_safe:nnn {}
3129   }{
3130     \stex_require_archive:n { #1 }
3131     \str_set:Nx \l__stex_importmodule_archive_str {#1}
3132     \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ #1 _manifest_prop}{}
3133     \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
3134     \exp_args:No \__stex_importmodule_get_module_safe:nnn \l__stex_importmodule_str
3135   }
3136 }
3137 }
3138
3139 \cs_new_protected:Nn \__stex_importmodule_get_module_uri:nnn {
3140   \tl_if_empty:nF {#2} {
3141     \str_set:Nx \l_stex_import_uri_str { \l_stex_import_uri_str / #2}
3142   }
3143   \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
3144     Current~file:\stex_file_use:N \g_stex_current_file^^J
3145     Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3146   }
3147   \stex_if_module_exists:nTF { \l_stex_import_uri_str?#3} {
3148     \str_set:Nx \l_stex_import_ns_str { \l_stex_import_uri_str?#3}
3149   }{
3150     \stex_if_module_exists:nTF{ \stex_uri_use:N \l_stex_current_ns_uri ? #3} {
3151       \str_set:Nx \l_stex_import_ns_str { \stex_uri_use:N \l_stex_current_ns_uri ? #3}
3152     }{
3153       \__stex_importmodule_get_from_file:nnn{#1}{#2}{#3}
3154       \str_set:Nx \l_stex_import_ns_str { \l_stex_import_uri_str?#3}
3155     }
3156   }
3157 }
3158 \cs_new_protected:Nn \__stex_importmodule_get_module_uri_safe:nnn {
3159   \tl_if_empty:nF {#2} {
3160     \str_set:Nx \l_stex_import_uri_str { \l_stex_import_uri_str / #2}
3161   }
3162   \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
3163     Current~file:\stex_file_use:N \g_stex_current_file^^J
3164     Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3165   }
3166   \stex_if_module_exists:nTF { \l_stex_import_uri_str?#3} {
3167     \str_set:Nx \l_stex_import_ns_str { \l_stex_import_uri_str?#3}
3168   }{
3169     \stex_if_module_exists:nTF{ \stex_uri_use:N \l_stex_current_ns_uri ? #3} {
3170       \str_set:Nx \l_stex_import_ns_str { \stex_uri_use:N \l_stex_current_ns_uri ? #3}
3171     }{
3172       \__stex_importmodule_get_from_file_safe:nnn{#1}{#2}{#3}
3173       \str_set:Nx \l_stex_import_ns_str { \l_stex_import_uri_str?#3}
3174     }
3175   }
3176 }
3177

```

```

3178 \cs_new_protected:Nn \__stex_importmodule_get_module:nnn {
3179   \stex_debug:nn{importmodule}{Requiring~>[#1]#2?#3<}
3180   \__stex_importmodule_get_module_uri:nnn{#1}{#2}{#3}
3181   \stex_activate_module:o \l_stex_import_ns_str
3182 }
3183
3184 \cs_new_protected:Nn \__stex_importmodule_get_module_safe:nnn {
3185   \stex_debug:nn{importmodule}{Requiring~>[#1]#2?#3<}
3186   \__stex_importmodule_get_module_uri_safe:nnn{#1}{#2}{#3}
3187 }
3188
3189 \cs_new_protected:Nn \__stex_importmodule_get_from_file:nnn {
3190   \stex_file_resolve:Nx \l_stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3191   \str_set:Nx \l_stex_importmodule_str {\stex_file_use:N \l_stex_importmodule_seq}
3192   \stex_debug:nn{imports}{Looking-for-\l_stex_import_uri_str?#3...}
3193   \__stex_importmodule_check_file:nn{ /#3.tex }{
3194     \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}){
3195       \__stex_importmodule_check_file:nn{/#3.en.tex} {
3196         \__stex_importmodule_check_file:nn{.tex} {
3197           \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex} {
3198             \__stex_importmodule_check_file:nn{.en.tex} {
3199               \msg_error:nnx{\stex}{error/unknownmodule}{\l_stex_import_uri_str?#3}
3200             }
3201           }
3202         }
3203       }
3204     }
3205   }
3206 \stex_if_smsmode:TF{
3207   \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l_stex_importmodule_str}{\stex_file_use:N
3208     \stex_debug:nn{imports}{Skipping-current-file}
3209   }{
3210     \__stex_importmodule_load_file:n{#3}
3211   }
3212 }{
3213   \__stex_importmodule_load_file:n{#3}
3214 }
3215 }
3216 \cs_new_protected:Nn \__stex_importmodule_get_from_file_safe:nnn {
3217   \stex_file_resolve:Nx \l_stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3218   \str_set:Nx \l_stex_importmodule_str {\stex_file_use:N \l_stex_importmodule_seq}
3219   \stex_debug:nn{imports}{Looking-for-\l_stex_import_uri_str?#3...}
3220   \__stex_importmodule_check_file:nn{ /#3.tex }{
3221     \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
3222       \__stex_importmodule_check_file:nn{/#3.en.tex} {
3223         \__stex_importmodule_check_file:nn{.tex} {
3224           \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex} {
3225             \__stex_importmodule_check_file:nn{.en.tex} {
3226               }
3227             }
3228           }
3229         }
3230       }
3231     }

```

```

3232 \stex_if_smsmode:TF{
3233   \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
3234     \stex_debug:nn{imports}{Skipping-current-file}
3235   }{
3236     \IfFileExists{ \l__stex_importmodule_str }{
3237       \__stex_importmodule_load_file:n{#3}
3238     }{}
3239   }
3240 }{
3241   \IfFileExists{ \l__stex_importmodule_str }{
3242     \__stex_importmodule_load_file:n{#3}
3243   }{}
3244 }
3245 }
3246
3247 \cs_new_protected:Nn \__stex_importmodule_load_file:n {
3248   \stex_file_in_smsmode:on \l__stex_importmodule_str {
3249     \str_if_empty:NF \l__stex_importmodule_archive_str {
3250       \stex_set_current_archive:n \l__stex_importmodule_archive_str
3251     }
3252     \stex_debug:nn{modules}{Loading~\l__stex_importmodule_str}
3253   }
3254   \stex_if_module_exists:nF {\l_stex_import_uri_str?#1} {
3255     \msg_error:nnx{stex}{error/unknownmodule}{\l_stex_import_uri_str?#1}
3256   }
3257 }
3258
3259 \cs_new_protected:Npn \__stex_importmodule_check_file:nn #1 {
3260   \stex_debug:nn{imports}{Checking~\l__stex_importmodule_str #1}
3261   \IfFileExists{ \l__stex_importmodule_str #1 }{
3262     \stex_debug:nnf{imports}{Success}
3263     \str_set:Nx \l__stex_importmodule_str { \l__stex_importmodule_str #1 }
3264   }
3265 }

```

(End of definition for `\stex_import_require_module:nnn`. This function is documented on page 130.)

### \importmodule

```

3266 \stex_new_stylable_cmd:nnnn{importmodule} { O{} m } {
3267   \__stex_importmodule_import_module:nn {#1}{#2}
3268   \stex_smsmode_do:
3269 }{}
3270 \stex_deactivate_macro:Nn \importmodule {module-environments}
3271
3272 \cs_new_protected:Nn \__stex_importmodule_import_module:nn {
3273   \stex_import_module_uri:nn { #1 }{ #2 }
3274   \stex_import_require_module:ooo
3275     \l_stex_import_archive_str
3276     \l_stex_import_path_str
3277     \l_stex_import_name_str
3278   \stex_execute_in_module:x{
3279     \stex_activate_module:n{\l_stex_import_ns_str}
3280   }
3281   \stex_module_add_morphism:nonn

```

```

3282     {}{\l_stex_import_ns_str}{import}{}}
3283 \stex_if_do_html:T {
3284   \stex_annotation_invisible:nn
3285   {shtml:import=\l_stex_import_ns_str} {}
3286 }
3287 \stex_if_smsmode:F{
3288   \group_begin:
3289   \tl_set:Nn \thisarchive {\#1}
3290   \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3291   \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3292   \tl_clear:N \thisstyle
3293   \stex_style_apply:
3294   \group_end:
3295 }
3296 }
3297
3298 \cs_new_protected:Nn \__stex_importmodule_import_module_presms:nn {
3299   \stex_import_module_uri:nn { #1 }{ #2 }
3300   \tl_gput_right:Nx \g_stex_sms_import_code {
3301     \stex_import_require_module_safe:nnn
3302       {\l_stex_import_archive_str}
3303       {\l_stex_import_path_str}
3304       {\l_stex_import_name_str}
3305   }
3306 }
3307
3308 \stex_sms_allow_escape:N \importmodule
3309 \stex_every_module:n {\stex_reactivate_macro:N \importmodule}
3310 \stex_sms_allow_import:Nn \importmodule {
3311   \stex_reactivate_macro:N \importmodule
3312   \let \__stex_importmodule_import_module:nn \__stex_importmodule_import_module_presms:nn
3313 }
3314
3315 \stex_new_stylable_cmd:nnnn[requiremodule] { 0{} m } {
3316   \stex_import_module_uri:nn { #1 }{ #2 }
3317   \stex_import_require_module:ooo
3318     \l_stex_import_archive_str
3319     \l_stex_import_path_str
3320     \l_stex_import_name_str
3321   \stex_do_up_to_module:x{
3322     \stex_activate_module:n{\l_stex_import_ns_str}
3323   }
3324   \stex_if_do_html:T {
3325     \stex_annotation_invisible:nn
3326     {shtml:import=\l_stex_import_ns_str} {}
3327   }
3328 \stex_if_smsmode:F{
3329   \group_begin:
3330   \tl_set:Nn \thisarchive {\#1}
3331   \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3332   \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3333   \tl_clear:N \thisstyle
3334   \stex_style_apply:
3335   \group_end:

```

```

3336   }
3337   \stex_smsmode_do:
3338 }{}
3339 \stex_deactivate_macro:Nn \requiremodule {module-environments}

```

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

### 13.7.2 Theory Morphisms

```

3340 <@=stex_morphisms>
3341 \stex_new_stylable_cmd:nnnn {assign} { m m }{
3342   \stex_get_in_morphism:n{#1}
3343   \stex_assign_do:n{#2}
3344   \stex_smsmode_do:
3345 }{}
3346 \stex_sms_allow_escape:N\assign
3347
3348 \cs_new_protected:Nn \stex_assign_do:n{
3349   \stex_debug:nn{assign}{Assigning~\l_stex_get_symbol_name_str~to~\tl_to_str:n{#1}}
3350   \tl_if_empty:NF \l_stex_get_symbol_def_tl {
3351     \%msg_error:nnxx{stex}{error/symbolalreadydefined}{\l_stex_get_symbol_name_str}{
3352       % morphism~\l_stex_feature_name_str
3353       %}
3354   }
3355   \stex_check_term:n{#1}
3356   \stex_debug:nn{HERE!}){
3357     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str^J
3358     \tl_to_str:n{#1}
3359   }
3360   \stex_if_do_html:T{
3361     \stex_annotation_invisible:nn{shtml:assign={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_
3362       \stex_annotation_force_break:n{
3363         \mode_if_math:T\hbox{\$ \stex_annotation:nn{shtml:definiens={}{}{#1} \$}
3364       }
3365     }
3366   }
3367   \exp_args:Ne \stex_metagroup_do_in:nx{
3368     \l_stex_current_module_str / \l_stex_feature_name_str
3369   }{
3370     \prop_put:Nnn \exp_not:N \l_stex_morphism_symbols_prop
3371     {[ \l_stex_get_symbol_mod_str ]/[ \l_stex_get_symbol_name_str ]}
3372     {
3373       {\l_stex_get_symbol_macro_str}
3374       {\int_use:N \l_stex_get_symbol_arity_int}
3375       {\l_stex_get_symbol_args_tl}
3376       {\exp_not:n{#1}}
3377       {\exp_args:No\exp_not:n\l_stex_get_symbol_type_tl}
3378       {\exp_args:No\exp_not:n\l_stex_get_symbol_return_tl}
3379       {\l_stex_get_symbol_invoke_cs}
3380     }
3381   }
3382 }
3383
3384

```

```

3385 \stex_new_stylable_cmd:nnnn {renamedecl} { m 0{} m }{
3386   \stex_get_in_morphism:n{#1}
3387   \_stex_renamedecl_do:nn{#2}{#3}
3388   \stex_smsmode_do:
3389 }{}
3390 \stex_sms_allow_escape:N\renamedecl
3391
3392 \cs_new_protected:Nn \_stex_renamedecl_do:nn {
3393   \stex_debug:nn{renamedecl}{Renaming~\l_stex_get_symbol_name_str~to~[#1]{#2}}
3394   \stex_if_do_html:T{
3395     \exp_args:Ne \stex_annotation_invisible:nn{
3396       shtml:rename={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str},
3397       shtml:macroname={#2}
3398       \str_if_empty:nF{#1}{ ,shtml:to={#1} }
3399     }{}
3400   }
3401   \exp_args:Ne \stex_metagroup_do_in:nx{
3402     \l_stex_current_module_str / \l_stex_feature_name_str
3403   }{
3404     \prop_put:Nnn \exp_not:N \l_stex_morphism_renames_prop
3405     {\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str}{##2}{%
3406       \tl_if_empty:nTF{#1}{\l_stex_feature_name_str/\l_stex_get_symbol_name_str}{#1}
3407     }%
3408   }
3409 }
3410
3411 \stex_new_stylable_cmd:nnnn {assignMorphism} { m m }{
3412   \str_clear:N \l__stex_morphisms_morphism_dom_str
3413   \stex_iterate_morphisms:nn\l_stex_current_domain_str{
3414     \stex_debug:nn{assignMorphism}{%
3415       Checking:~#1-vs:^^J##1^^J##2^^J##3^^J##4
3416     }
3417     \str_if_eq:nnTF{#1}{##1}{%
3418       \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3419     }{
3420       \stex_str_if_ends_with:nnT{##2}{#1}{%
3421         \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3422       }
3423     }
3424   }
3425   \str_if_empty:NT \l__stex_morphisms_morphism_dom_str {
3426     \msg_error:nnn{\stex}{error/nomorphism}{#1}
3427   }
3428   \bool_set_false:N \l_tmpa_bool
3429   \stex_iterate_morphisms:nn \l_stex_current_module_str {
3430     \stex_debug:nn{assignMorphism}{%
3431       Checking:~#2-vs:^^J##1^^J##2^^J##3^^J##4
3432     }
3433     \str_if_eq:nnTF{#2}{##1}{%
3434       \stex_debug:nn{assignMorphism}{match!}
3435       \stex_iterate_break:n{
3436         \stex_annotation_invisible:nn{
3437           shtml:assignMorphismFrom={\l__stex_morphisms_morphism_dom_str}
3438           ahtml:assignMorphismTo={\l_stex_current_module_str?##1}

```

```

3439     }{}
3440     \bool_set_true:N \l_tmpa_bool
3441   }
3442 }{
3443   \stex_if_ends_with:nnT{##2}{#2}{
3444     \stex_debug:nn{assignMorphism}{match!}
3445     \stex_iterate_break:n{
3446       \stex_annotation_invisible:nn{
3447         shtml:assignMorphismFrom={\l_stex_morphisms_morphism_dom_str},
3448         shtml:assignMorphismTo={\l_stex_current_module_str?##1}
3449       }{}
3450       \bool_set_true:N \l_tmpa_bool
3451     }
3452   }
3453 }
3454 }
3455 \bool_if:NF \l_tmpa_bool {
3456   \msg_error:nnn{\stex}{error/nomorphism}{#2}
3457 }
3458 }{}
3459 \cs_new_protected:Nn \__stex_morphisms_do_morph_assign:nnn {
3460   \stex_iterate_break:n{
3461     \str_set:Nx \l_stex_morphisms_morphism_dom_str { \l_stex_current_domain_str ? #1 }
3462     \stex_debug:nn{assignMorphism}{match!}
3463     \stex_iterate_symbols:nn{#2}{{
3464       \stex_debug:nn{assignMorphism}{removing~##1?##3}
3465       % TODO: non-trivial assignments
3466       \prop_remove:NN \l_stex_morphism_symbols_prop {
3467         [##1]/[##3]
3468       }
3469     }
3470   }
3471 }
3472
3473 \stex_deactivate_macro:Nn \assign {morphism~environments}
3474 \stex_deactivate_macro:Nn \renamedecl {morphism~environments}
3475 \stex_deactivate_macro:Nn \assignMorphism {morphism~environments}
3476 \stex_new_stylable_env:nnnnnnn {copymodule}{m 0{} m}{%
3477
3478   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3479
3480   \stex_if_smsmode:F {
3481     \tl_set:Nn \thiscopyname { #1 }
3482     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3483     \stex_style_apply:
3484   }
3485   \stex_smsmode_do:
3486 }{
3487   \stex_if_smsmode:F {
3488     \stex_style_apply:
3489   }
3490   \stex_structural_feature_morphism_end:
3491 }{}{}{}%
3492 \stex_deactivate_macro:Nn \copymodule {module~environments}

```

```

3493 \stex_every_module:n {
3494   \stex_reactivate_macro:N \copymodule
3495 }
3496 \stex_sms_allow_env:n{copymodule}
3497
3498 \stex_new_stylable_env:nnnnnn {interpretmodule}{m 0{} m}{
3499   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3500   \stex_if_smsmode:F {
3501     \tl_set:Nn \thiscopyname { #1 }
3502     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3503     \stex_style_apply:
3504   }
3505   \stex_smsmode_do:
3506 }{
3507   \stex_structural_feature_morphism_check_total:
3508   \stex_if_smsmode:F {
3509     \stex_style_apply:
3510   }
3511   \stex_structural_feature_morphism_end:
3512 }{}{}{}
3513 \stex_deactivate_macro:Nn \interpretmodule {module~environments}
3514 \stex_every_module:n {
3515   \stex_reactivate_macro:N \interpretmodule
3516 }
3517 \stex_sms_allow_env:n{interpretmodule}
3518 \stex_new_stylable_env:nnnnnn {realization}{0{} m}{

3519
3520   \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}
3521   \% \stex_execute_in_module:x{
3522   % \stex_activate_module:n{\l_stex_current_domain_str}
3523   %}
3524   \stex_if_smsmode:F {
3525     \tl_set:Nn \thiscopyname { #2 }
3526     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3527     \stex_style_apply:
3528   }
3529   \stex_smsmode_do:
3530 }{
3531   \stex_structural_feature_morphism_check_total:
3532   \stex_if_smsmode:F {
3533     \stex_style_apply:
3534   }
3535   \stex_structural_feature_morphism_end:
3536 }{}{}{}
3537 \stex_deactivate_macro:Nn \realization {module~environments}
3538 \stex_every_module:n {
3539   \stex_reactivate_macro:N \realization
3540 }
3541 \stex_sms_allow_env:n{realization}
3542 \cs_new_protected:Nn \__stex_morphisms_parse_assign:n {
3543   \str_clear:N \l__stex_morphisms_name_str
3544   \str_clear:N \l__stex_morphisms_newname_str
3545   \tl_clear:N \l__stex_morphisms_ass_tl

```

```

3546 \stex_debug:nn{morphisms}{Parsing-#1}
3547 \exp_args:NNe \seq_set_split:Nnn \l__stex_morphisms_seq {\tl_to_str:n{0}} {#1}
3548 \int_compare:nNnTF {\seq_count:N \l__stex_morphisms_seq} = 1 {
3549   \stex_debug:nn{morphisms}{No~@}
3550   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_next_tl
3551 }{
3552   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3553   \stex_debug:nmf{morphisms}{Name:~\l__stex_morphisms_name_str}
3554   \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3555   \tl_set:Nx \l__stex_morphisms_next_tl {\seq_use:Nn \l__stex_morphisms_seq @}
3556 }
3557 \exp_args:NNNo \seq_set_split:Nnn \l__stex_morphisms_seq = \l__stex_morphisms_next_tl
3558 \str_if_empty:NTF \l__stex_morphisms_name_str {
3559   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3560   \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3561   \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3562 }{
3563   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_newname_str
3564   \exp_args:NNo \str_set:Nn \l__stex_morphisms_newname_str \l__stex_morphisms_newname_str
3565   \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3566 }
3567 \__stex_morphisms_do_parsed_assign:
3568 }
3569
3570 \cs_new_protected:Nn \__stex_morphisms_do_parsed_assign: {
3571   \exp_args:No \stex_get_in_morphism:n \l__stex_morphisms_name_str
3572   \str_if_empty:NF \l__stex_morphisms_newname_str {
3573     \exp_after:wN \__stex_morphisms_do_parsed_newname: \l__stex_morphisms_newname_str \__ste
3574   }
3575   \tl_if_empty:NF \l__stex_morphisms_ass_tl {
3576     \exp_args:No \stex_assign_do:n \l__stex_morphisms_ass_tl
3577   }
3578 }
3579
3580 \cs_new_protected:Nn \__stex_morphisms_do_parsed_newname: {
3581   \peek_charcode:NTF [ {
3582     \__stex_morphisms_do_parsed_newname:w
3583   }{
3584     \__stex_morphisms_do_parsed_newname:w []
3585   }
3586 }
3587
3588 \cs_new_protected:Npn \__stex_morphisms_do_parsed_newname:w [#1] #2 \__stex_morphisms_end: {
3589   \stex_renamedecl_do:nn{#1}{#2}
3590 }
3591
3592 \stex_new_stylable_cmd:nnnn{copymod}{m 0{} m m}{
3593   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3594
3595   \clist_map_function:nN{#4}\__stex_morphisms_parse_assign:n
3596
3597   \stex_if_smsmode:F {
3598     \tl_set:Nn \thiscopyname { #1 }
3599     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str

```

```

3600     \stex_style_apply:
3601 }
3602 \stex_structural_feature_morphism_end:
3603 \stex_smsmode_do:
3604 }{}
3605 \stex_deactivate_macro:Nn \copymod {module~environments}
3606 \stex_every_module:n {
3607   \stex_reactivate_macro:N \copymod
3608 }
3609 \stex_sms_allow_escape:N\copymod
3610
3611
3612 \stex_new_stylable_cmd:nnnn{interpretmod}{0{} m m}{
3613   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3614
3615 \clist_map_function:nN{#4}\_stex_morphisms_parse_assign:n
3616
3617 \stex_if_smsmode:F {
3618   \tl_set:Nn \thiscopynname { #1 }
3619   \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3620   \stex_style_apply:
3621 }
3622 \stex_structural_feature_morphism_check_total:
3623 \stex_structural_feature_morphism_end:
3624 \stex_smsmode_do:
3625 }{}
3626 \stex_deactivate_macro:Nn \interpretmod {module~environments}
3627 \stex_every_module:n {
3628   \stex_reactivate_macro:N \interpretmod
3629 }
3630 \stex_sms_allow_escape:N\interpretmod
3631
3632
3633 \stex_new_stylable_cmd:nnnn{realize}{0{} m m}{
3634   \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}
3635
3636 \clist_map_function:nN{#3}\_stex_morphisms_parse_assign:n
3637
3638 \stex_if_smsmode:F {
3639   \tl_set:Nn \thiscopynname { #1 }
3640   \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3641   \stex_style_apply:
3642 }
3643 \stex_structural_feature_morphism_check_total:
3644 \stex_structural_feature_morphism_end:
3645 \stex_smsmode_do:
3646 }{}
3647 \stex_deactivate_macro:Nn \realize {module~environments}
3648 \stex_every_module:n {
3649   \stex_reactivate_macro:N \realize
3650 }
3651 \stex_sms_allow_escape:N\realize

```

## 13.8 Symbols

### 13.8.1 Declarations

3652 `<@@=stex_symdecl>`

Some setup:

```
\stex_if_check_terms_p:
\stex_if_check_terms:TF
3653 \stex_if_html_backend:TF {
3654   \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3655     \prg_return_false:
3656   }
3657 }{
3658   \stex_get_env:Nn\__stex_symdecl_env_str{STEX_CHECKTERMS}
3659   \str_if_empty:NF\__stex_symdecl_env_str{
3660     \exp_args:No \str_if_eq:nnF \__stex_symdecl_env_str{false}{
3661       \bool_set_true:N \c_stex_check_terms_bool
3662     }
3663   }
3664   \bool_if:NTF \c_stex_check_terms_bool {
3665     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3666       \prg_return_true:
3667     }
3668   }{
3669     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3670       \prg_return_false:
3671     }
3672   }
3673 }
```

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

`\stex_check_term:n`

```
3674 \stex_if_check_terms:TF{
3675   \cs_new_protected:Nn \stex_check_term:n {
3676     \hbox_set:Nn \l_tmpa_box {
3677       \group_begin:
3678         $$#1$$
3679       \group_end:
3680     }
3681   }
3682 }{
3683   \cs_new_protected:Nn \stex_check_term:n {}
3684 }
```

(End of definition for `\stex_check_term:n`. This function is documented on page 124.)

symdecl arguments:

```
3685 \stex_keys_define:nnnn{symargs}{
3686   \str_clear:N \l_stex_key_args_str
3687   \str_clear:N \l_stex_key_role_str
3688   \str_clear:N \l_stex_key_reorder_str
3689   \str_clear:N \l_stex_key_assoc_str
3690 }{
3691   args      .str_set:N  = \l_stex_key_args_str ,
3692   reorder   .str_set:N  = \l_stex_key_reorder_str ,
```

```

3693     assoc      .choices:nnn = {bin,binl,binr,pre,conj,pwconj}
3694     {\str_set:Nx \l_stex_key_assoc_str \l_keys_choice_tl},
3695     role       .str_set:N      = \l_stex_key_role_str
3696 }{}
3697
3698 \stex_keys_define:nnnn{decl}{%
3699   \str_clear:N \l_stex_key_name_str
3700   \str_clear:N \l_stex_key_args_str
3701   \tl_clear:N \l_stex_key_type_tl
3702   \tl_clear:N \l_stex_key_def_tl
3703   \tl_clear:N \l_stex_key_return_tl
3704   \str_clear:N \l_stex_key_wikidata_str
3705   \clist_clear:N \l_stex_key_argtypes_clist
3706 }%
3707   name      .str_set:N = \l_stex_key_name_str ,
3708
3709   return    .tl_set:N      = \l_stex_key_return_tl ,
3710   argtypes  .clist_set:N  = \l_stex_key_argtypes_clist ,
3711   wikidata  .str_set:N  = \l_stex_key_wikidata_str ,
3712
3713   type      .tl_set:N      = \l_stex_key_type_tl ,
3714   def       .tl_set:N      = \l_stex_key_def_tl ,
3715
3716   align     .code:n       = {},
3717   gfc      .code:n       = {}
3718 }{style,deprecate,symargs}
3719 % \_stex_do_deprecation:n{#2}

\symdecl
3720 \str_new:N \l_stex_mroname_str
3721 \stex_new_stylable_cmd:nnnn {symdecl} { s m O{} } {
3722   \stex_keys_set:nn{decl}{#3}
3723   \IfBooleanTF #1 {
3724     \str_clear:N \l_stex_mroname_str
3725   }{
3726     \str_set:Nx \l_stex_mroname_str { #2 }
3727   }
3728 \stex_symdecl_top:n{#2}
3729
3730 \stex_if_smsmode:F{
3731   \group_begin:
3732   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3733   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3734   \tl_set_eq:NN \thistype \l_stex_key_type_tl
3735   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
3736   \tl_set_eq:NN \thisargs \l_stex_key_args_str
3737   \tl_clear:N \thisstyle
3738   \stex_style_apply:
3739   \group_end:
3740 }
3741 \stex_smsmode_do:
3742 }{}
3743 \stex_deactivate_macro:Nn \symdecl {module~environments}
3744 \stex_every_module:n {\stex_reactivate_macro:N \symdecl}

```

```

3745 \stex_sms_allow_escape:N \symdecl
(End of definition for \symdecl. This function is documented on page 87.)
```

### \stex\_symdecl\_top:n

```

3746 \cs_new_protected:Nn \stex_symdecl_top:n {
3747   \str_if_empty:NT \l_stex_key_name_str {
3748     \str_set:Nx \l_stex_key_name_str { #1 }
3749   }
3750   \stex_symdecl_do:
3751   \_stex_symdecl_check_terms:
3752   \_stex_symdecl_add_decl:
3753   \stex_if_do_html:T {
3754     \_stex_symdecl_html:
3755   }
3756 }
3757
3758 \cs_new_protected:Nn \_stex_symdecl_add_decl: {
3759   \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnnN}{
3760     {\l_stex_mroname_str}
3761     {\l_stex_key_name_str}
3762     {\int_use:N \l_stex_get_symbol_arity_int}
3763     {\l_stex_get_symbol_args_tl}
3764     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }%{\exp_args:No \exp_not:n \l_stex_key_def_tl
3765     {}%{\exp_args:No \exp_not:n \l_stex_key_type_tl}
3766     {}%{\exp_args:No \exp_not:n \l_stex_key_return_tl}
3767     \stex_invoke_symbol:
3768   }
3769   \exp_args:Ne \stex_ref_new_symbol:n
3770     {\l_stex_current_module_str?\l_stex_key_name_str}
3771 }
3772
3773 \cs_new:Nn \stex_return_args:nn {
3774   {\svar{ARGUMENT_#1}\_stex_eat_exclamation_point:}
3775 }
3776
3777 \cs_new_protected:Nn \stex_symdecl_html: {
3778   \exp_args:Ne \stex_annotation_invisible:nn {
3779     \shtml:symdecl = {\l_stex_current_module_str ? \l_stex_key_name_str},
3780     \shtml:args = {\l_stex_key_args_str}
3781     \str_if_empty:NF \l_stex_mroname_str {
3782       \shtml:macroname={\l_stex_mroname_str}
3783     }
3784     \str_if_empty:NF \l_stex_key_wikidata_str {
3785       \shtml:wikidata={\l_stex_key_wikidata_str}
3786     }
3787     \str_if_empty:NF \l_stex_key_assoc_str {
3788       \shtml:assocotype={\l_stex_key_assoc_str}
3789     }
3790     \str_if_empty:NF \l_stex_key_reorder_str {
3791       \shtml:reorderargs={\l_stex_key_reorder_str}
3792     }
3793     \str_if_empty:NF \l_stex_key_role_str {
3794       \shtml:role={\l_stex_key_role_str}
```

```

3795     }
3796 }{\hbox\bgroup\stex_annotation_force_break:n{
3797   \bool_set_true:N \stex_in_invisible_html_bool
3798   \tl_if_empty:NF \l_stex_key_type_tl {
3799     $\stex_annotation:nn{shtml:type={}}{\l_stex_key_type_tl}$
3800   }
3801   \tl_if_empty:NF \l_stex_key_def_tl {
3802     $\stex_annotation:nn{shtml:definiens={}}{\l_stex_key_def_tl}$
3803   }
3804   \tl_if_empty:NF \l_stex_key_return_tl{
3805     \exp_args:Nno \use:n{
3806       \cs_generate_from_arg_count:NNnn \l_stex_symdecl_cs
3807       \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
3808       \tl_set:Nx \l_stex_symdecl_args_tl {\l_stex_map_args:N \l_stex_return_args:nn}
3809       $\stex_annotation:nn{shtml:returntype={}}{
3810         \exp_after:wN \l_stex_symdecl_cs \l_stex_symdecl_args_tl!
3811       }$}
3812   }
3813   \clist_if_empty:NF \l_stex_key_argtypes_clist {
3814     \stex_annotation:nn{shtml:argtypes={}}{\l_stex_annotation_force_break:n{
3815       \clist_map_inline:Nn \l_stex_key_argtypes_clist {
3816         $\stex_annotation:nn{shtml:type={}}{\##1}$
3817       }
3818     }}
3819   }
3820 } \egroup
3821 }

```

(End of definition for `\stex_symdecl_top:n`. This function is documented on page [125](#).)

**\stex\_symdecl\_do:** Requires the above keys and `\l_stex_macroname_str` to be set first

```

3822 \cs_new_protected:Nn \stex_symdecl_do: {
3823   \stex_do_deprecation:n \l_stex_key_name_str
3824   \__stex_symdecl_parse_arity:
3825   \__stex_symdecl_do_args:
3826 }
3827
3828 \int_new:N \l_stex_assoc_args_count
3829
3830 \cs_new_protected:Nn \__stex_symdecl_parse_arity: {
3831   \int_zero:N \l_stex_get_symbol_arity_int
3832   \int_zero:N \l_stex_assoc_args_count
3833   \str_map_inline:Nn \l_stex_key_args_str {
3834     \str_case:nnF \##1 {
3835       0 { \str_map_break: }
3836       1 { \str_map_break:n{
3837         \int_set:Nn \l_stex_get_symbol_arity_int {1}
3838         \str_set:Nn \l_stex_key_args_str {i}
3839       } }
3840       2 { \str_map_break:n{
3841         \int_set:Nn \l_stex_get_symbol_arity_int {2}
3842         \str_set:Nn \l_stex_key_args_str {ii}
3843       } }
3844       3 { \str_map_break:n{

```

```

3845     \int_set:Nn \l_stex_get_symbol_arity_int {3}
3846     \str_set:Nn \l_stex_key_args_str {iii}
3847   } }
3848   4 { \str_map_break:n{
3849     \int_set:Nn \l_stex_get_symbol_arity_int {4}
3850     \str_set:Nn \l_stex_key_args_str {iiii}
3851   } }
3852   5 { \str_map_break:n{
3853     \int_set:Nn \l_stex_get_symbol_arity_int {5}
3854     \str_set:Nn \l_stex_key_args_str {iiiii}
3855   } }
3856   6 { \str_map_break:n{
3857     \int_set:Nn \l_stex_get_symbol_arity_int {6}
3858     \str_set:Nn \l_stex_key_args_str {iiiiii}
3859   } }
3860   7 { \str_map_break:n{
3861     \int_set:Nn \l_stex_get_symbol_arity_int {7}
3862     \str_set:Nn \l_stex_key_args_str {iiiiiii}
3863   } }
3864   8 { \str_map_break:n{
3865     \int_set:Nn \l_stex_get_symbol_arity_int {8}
3866     \str_set:Nn \l_stex_key_args_str {iiiiiiii}
3867   } }
3868   9 { \str_map_break:n{
3869     \int_set:Nn \l_stex_get_symbol_arity_int {9}
3870     \str_set:Nn \l_stex_key_args_str {iiiiiiiii}
3871   } }
3872   i {\int_incr:N \l_stex_get_symbol_arity_int}
3873   b {\int_incr:N \l_stex_get_symbol_arity_int}
3874   a {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3875   B {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3876 }{
3877   \msg_error:nnnx{stex}{error/wrongargs}{
3878     \l_stex_current_module_str ? \l_stex_key_name_str
3879   }{##1}
3880 }
3881 }
3882 }
3883
3884 \cs_new_protected:Nn \__stex_symdecl_do_args: {
3885   \tl_clear:N \l_stex_get_symbol_args_tl
3886   \int_step_inline:nn \l_stex_get_symbol_arity_int {
3887     \tl_put_right:Nn \l_stex_get_symbol_args_tl {##1}
3888     \tl_put_right:Nx \l_stex_get_symbol_args_tl {
3889       \str_item:Nn \l_stex_key_args_str {##1}
3890     }
3891   }
3892 }

```

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

### `\stex_symdecl_check_terms:`

```

3893 \cs_new_protected:Nn \stex_symdecl_check_terms: {
3894   \stex_check_term:n{

```

```

3895   \stex_debug:nn{check_terms}{Checking~type...}
3896   \group_begin:\l_stex_key_type_tl\group_end:
3897   \stex_debug:nn{check_terms}{Checking~definiens...}
3898   \group_begin:\l_stex_key_def_tl\group_end:
3899   \stex_debug:nn{check_terms}{Checking~return...}
3900   \group_begin:\l_stex_key_return_tl!\group_end:
3901   \stex_debug:nn{check_terms}{Checking~argument-types...}
3902   \group_begin:\l_stex_key_argtypes_clist\group_end:
3903 }
3904 }

```

(End of definition for `\_stex_symdecl_check_terms`. This function is documented on page 125.)

### `\textsymdecl`

```

3905
3906 \stex_keys_define:nnnn{textsymdecl}{
3907   \str_clear:N \l_stex_key_name_str
3908   \tl_clear:N \l_stex_key_type_tl
3909   \tl_clear:N \l_stex_key_def_tl
3910 }{
3911   name      .str_set:N  = \l_stex_key_name_str ,
3912   type      .tl_set:N    = \l_stex_key_type_tl ,
3913   def       .tl_set:N    = \l_stex_key_def_tl
3914 }{style,deprecate}
3915
3916 \stex_new_stylable_cmd:nnnn {textsymdecl} {m 0{} m} {
3917   \stex_keys_set:nn{symdef}{}
3918   \stex_keys_set:nn{textsymdecl}{#2}
3919   \str_set:Nx \l_stex_macroname_str { #1 }
3920   \str_if_empty:NT \l_stex_key_name_str {
3921     \str_set:Nn \l_stex_key_name_str {#1}
3922   }%
3923   % \str_set:Nx \l_stex_key_name_str {\l_stex_key_name_str-sym}
3924   %
3925   \str_set:Nn \l_stex_key_role_str {textsymdecl}
3926
3927   \stex_symdecl_do:
3928   \_stex_symdecl_check_terms:
3929   \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnnN} {
3930     {\l_stex_macroname_str}
3931     {\l_stex_key_name_str}
3932     {0}{}
3933     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
3934     {}% type
3935     {\use:c{\#1name_nospace}}% return
3936     \stex_invoke_text_symbol:
3937   }
3938   \exp_args:Ne \stex_ref_new_symbol:n
3939     {\l_stex_current_module_str?\l_stex_key_name_str}
3940   \stex_if_do_html:T {
3941     \_stex_symdecl_html:
3942   }
3943
3944   \int_set:Nn \l_stex_get_symbol_arity_int 0

```

```

3945 \tl_clear:N \l_stex_key_op_tl
3946 \str_clear:N \l_stex_key_intent_str
3947 \str_clear:N \l_stex_key_prec_str
3948 \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
3949 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
3950 \stex_notation_parse:n{\hbox{#3}}
3951 \stex_notation_add:
3952 \stex_if_do_html:T {
3953   \def\comp{\_comp}
3954   \stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
3955 }
3956 \stex_execute_in_module:xf{
3957   \__stex_symdecl_set_textsymdecl_macro:nnn{#1}{\l_stex_current_module_str?\l_stex_key_name_str}
3958   \exp_not:n{#3}
3959 }
3960
3961 \stex_if_smsmode:F{
3962   \group_begin:
3963   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3964   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3965   \tl_clear:N \thisstyle
3966   \stex_style_apply:
3967   \group_end:
3968 }
3969 \stex_smsmode_do:
3970 }{}
3971 \stex_deactivate_macro:Nn \textsymdecl {module-environments}
3972 \stex_every_module:n {\stex_reactivate_macro:N \textsymdecl}
3973 \stex_sms_allow_escape:N \textsymdecl
3974
3975 \cs_new_protected:Nn \__stex_symdecl_set_textsymdecl_macro:nnn {
3976   \cs_set_protected:cpn{#1name_nospace}{#3}
3977   \cs_set_protected:cpn{#1name} {
3978     \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3979     \mode_if_math:T\hbox{\let\xspace\relax #3}
3980     \mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3981   }
3982 }
3983
3984 \cs_new_protected:Nn \stex_invoke_text_symbol: {
3985   \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3986   \stex_term_oms_or_omv:nnn{}{}{\maincomp{\let\xspace\relax\l_stex_current_return_t1}}
3987   \group_end:\mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3988 }

```

(End of definition for \textsymdecl. This function is documented on page 88.)

```
\stex_get_symbol:n
3989 \cs_new_protected:Nn \stex_get_symbol:n {
3990   \stex_get_symbol:n{ #1 }
3991   \str_if_empty:NT \l_stex_get_symbol_name_str {
3992     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3993   }
3994 }
```

```

3995 \cs_new_protected:Nn \_stex_get_symbol:n {
3996   \str_clear:N \l_stex_get_symbol_mod_str
3997   \str_clear:N \l_stex_get_symbol_name_str
3998   \cs_if_exist:cTF { #1 }{
3999     \cs_set_eq:Nc \l_stex_symdecl_cs { #1 }
4000     % command name
4001     \exp_args:Ne \tl_if_empty:nTF { \cs_argument_spec:N \l_stex_symdecl_cs }{
4002       % ...that takes no arguments
4003       \exp_args:Ne \cs_if_eq:NNTF { \tl_head:N \l_stex_symdecl_cs }
4004         \l_stex_invoke_symbol:nnnnnnN
4005         \l_stex_symdecl_get_symbol_from_cs:
4006         {\l_stex_symdecl_get_symbol_from_string:n { #1 }}
4007     }{
4008       \l_stex_symdecl_get_symbol_from_string:n { #1 }
4009     }
4010   }{
4011     \l_stex_symdecl_get_symbol_from_string:n { #1 }
4012   }
4013 }
4014 }
4015
4016 \int_new:N \l_stex_get_symbol_arity_int
4017 \cs_new_protected:Nn \l_stex_symdecl_get_symbol_from_cs: {
4018   \stex_debug:nn{symbols}{Getting~from~cs...}
4019   \stex_pseudogroup_with:nn{\l_stex_invoke_symbol:nnnnnnN}{
4020     \cs_set:Npn \l_stex_invoke_symbol:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {
4021       \str_set:Nn \l_stex_get_symbol_mod_str {##1}
4022       \str_set:Nn \l_stex_get_symbol_name_str {##2}
4023       \int_set:Nn \l_stex_get_symbol_arity_int {##3}
4024       \tl_set:Nn \l_stex_get_symbol_args_tl {##4}
4025       \tl_set:Nn \l_stex_get_symbol_def_tl {##5}
4026       \tl_set:Nn \l_stex_get_symbol_type_tl {##6}
4027       \tl_set:Nn \l_stex_get_symbol_return_tl {##7}
4028       \tl_set:Nn \l_stex_get_symbol_invoke_cs {##8}
4029     }
4030     \l_stex_symdecl_cs
4031   }
4032 }
4033
4034 \cs_new_protected:Nn \l_stex_symdecl_get_symbol_from_string:n {
4035   \stex_debug:nn{symbols}{Getting~from~string~#1...}
4036   \seq_set_split:Nnn \l_stex_symdecl_seq ? {#1}
4037   \seq_pop_right:NN \l_stex_symdecl_seq \l_stex_symdecl_name
4038   \seq_if_empty:NTF \l_stex_symdecl_seq {
4039     \exp_args:No \l_stex_symdecl_get_from_one_string:n {#1}
4040   }{
4041     \exp_args:NNe \exp_args:Nno \l_stex_symdecl_get_symbol_from_modules:nn {
4042       \seq_use:Nn \l_stex_symdecl_seq ?
4043     } \l_stex_symdecl_name
4044   }
4045 }
4046
4047 \cs_new_protected:Nn \l_stex_symdecl_sym_from_str_i:nnnn {
4048   \bool_lazy_any:nTF{

```

```

4049 {\str_if_eq_p:nn{#2}{#3}}
4050 {\str_if_eq_p:nn{#2}{#4}}
4051 {\stex_str_if_ends_with_p:nn{#4}{/#2}}
4052 }{
4053     \__stex_symdecl_sym_i_finish:nnnnnnn{#1}{#4}
4054 }{
4055     \__stex_symdecl_sym_i_gobble:nnnnnn
4056 }
4057 }
4058 \cs_new_protected:Nn \__stex_symdecl_sym_i_gobble:nnnnnn {}
4059
4060 \cs_new_protected:Nn \__stex_symdecl_sym_i_finish:nnnnnnn {
4061     \prop_map_break:n{\seq_map_break:n{
4062         \str_set:Nn \l_stex_get_symbol_mod_str {#1}
4063         \str_set:Nn \l_stex_get_symbol_name_str {#2}
4064         \int_set:Nn \l_stex_get_symbol_arity_int {#3}
4065         \tl_set:Nn \l_stex_get_symbol_args_tl {#4}
4066         \tl_set:Nn \l_stex_get_symbol_def_tl {#5}
4067         \tl_set:Nn \l_stex_get_symbol_type_tl {#6}
4068         \tl_set:Nn \l_stex_get_symbol_return_tl {#7}
4069         \tl_set:Nn \l_stex_get_symbol_invoke_cs {#8}
4070     }}
4071 }
4072
4073 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_modules:nn {
4074     \stex_debug:nn{symbols}{Getting~#2~in~#1...}
4075     \seq_map_inline:Nn \l_stex_all_modules_seq {
4076         \stex_str_if_ends_with:nnT{##1}{#1}{
4077             \prop_map_inline:cn{c_stex_module_##1_symbols_prop}){
4078                 \__stex_symdecl_sym_from_str_i:nnn{##1}{#2} ####2
4079             }
4080         }
4081     }
4082 }
4083
4084 \cs_new_protected:Nn \__stex_symdecl_get_from_one_string:n {
4085     \stex_debug:nn{symbols}{Getting~#1~anywhere...}
4086     \stex_iterate_symbols:n{
4087         \%stex_debug:nn{symbols}{>#1==##2~|~#1==##3<...}
4088         \bool_lazy_any:nT{
4089             {\str_if_eq_p:nn{#1}{##2}}
4090             {\str_if_eq_p:nn{#1}{##3}}
4091             {\stex_str_if_ends_with_p:nn{##3}{/#1}}
4092         }{
4093             \stex_iterate_break:n{
4094                 \str_set:Nn \l_stex_get_symbol_mod_str {##1}
4095                 \str_set:Nn \l_stex_get_symbol_name_str {##3}
4096                 \int_set:Nn \l_stex_get_symbol_arity_int {##4}
4097                 \tl_set:Nn \l_stex_get_symbol_args_tl {##5}
4098                 \tl_set:Nn \l_stex_get_symbol_def_tl {##6}
4099                 \tl_set:Nn \l_stex_get_symbol_type_tl {##7}
4100                 \tl_set:Nn \l_stex_get_symbol_return_tl {##8}
4101                 \tl_set:Nn \l_stex_get_symbol_invoke_cs {##9}
4102             }
4103         }
4104     }

```

```

4103     }
4104   }
4105 }
```

(End of definition for `\stex_get_symbol:n`. This function is documented on page 124.)

### 13.8.2 Notations

```
4106 <@@=stex_notations>
```

```

\_stex_map_args:N
\_stex_map_notation_args:N
4107 \cs_new:Nn \_stex_map_args:N {
4108   \tl_if_empty:NF \l_stex_get_symbol_args_tl {
4109     \exp_after:wN \__stex_notations_map_args_i:w \exp_after:wN
4110     #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
4111   }
4112 }
4113 \cs_new:Npn \__stex_notations_map_args_i:w #1 #2 #3 #4 \__stex_notations_args_end: {
4114   #1 #2 #3
4115   \tl_if_empty:nF{#4} {
4116     \__stex_notations_map_args_i:w #1 #4 \__stex_notations_args_end:
4117   }
4118 }
4119
4120 \cs_new:Nn \_stex_map_notation_args:N {
4121   \tl_if_empty:NF \l_stex_notation_args_tl {
4122     \exp_after:wN \__stex_notations_map_args_ii:w \exp_after:wN
4123     #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
4124   }
4125 }
4126 \cs_new:Npn \__stex_notations_map_args_ii:w #1 #2 #3 #4 #5 #6 \__stex_notations_args_end: {
4127   #1 #2 #3 #4 #5
4128   \tl_if_empty:nF{#6} {
4129     \__stex_notations_map_args_ii:w #1 #6 \__stex_notations_args_end:
4130   }
4131 }
```

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

notation arguments:

```

4132 \stex_keys_define:nnnn{notation} {
4133   \str_clear:N \l_stex_key_variant_str
4134   \str_clear:N \l_stex_key_prec_str
4135   \str_clear:N \l_stex_key_op_tl
4136   \str_clear:N \l_stex_key_intent_str
4137   \clist_clear:N \l_stex_key_intent_args_clist
4138 }{
4139   variant      .str_set_x:N = \l_stex_key_variant_str ,
4140   prec         .str_set_x:N = \l_stex_key_prec_str ,
4141   op           .tl_set:N    = \l_stex_key_op_tl ,
4142   intent        .str_set:N = \l_stex_key_intent_str ,
4143   argnames     .clist_set:N = \l_stex_key_intent_args_clist ,
4144   unknown       .code:n     = {
4145     \str_if_empty:NTF \l_keys_key_str {
4146       \str_set:Nx \l_stex_key_variant_str {\l_keys_key_tl}
```

```

4147    }{
4148      \str_set_eq:NN \l_stex_key_variant_str \l_keys_key_str
4149    }
4150  }
4151 }{style}

\notation

4152 \stex_new_stylable_cmd:nnnn {notation} { s m O{} m} {
4153   \stex_keys_set:nn{notation}{#3}
4154   \stex_get_symbol:n{#2}
4155   \stex_notation_parse:n{#4}
4156   \stex_if_check_terms:T{ \stex_notation_check: }
4157   \stex_notation_add:
4158   \stex_if_do_html:T {
4159     \def\comp{\_comp}
4160     \stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4161   }
4162 \IfBooleanTF#1{
4163   \stex_notation_set_default:n{
4164     \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4165   }
4166 }{}
4167 \stex_if_smsmode:F{
4168   \group_begin:
4169   \__stex_notations_styledefs:
4170   \stex_style_apply:
4171   \group_end:
4172 }
4173 \stex_smsmode_do:
4174 }{}
4175
4176 \cs_new_protected:Nn \__stex_notations_styledefs: {
4177   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4178   \str_set:Nn \thisdeclname \l_stex_get_symbol_name_str
4179   \tl_set:Nx \thisdecluri {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4180   \def\thisnotation{
4181     $
4182     \tl_set_eq:NN \l_stex_current_symbol_str\thisdecluri
4183     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs} {
4184       \stex_notation_make_args:
4185     }$
4186   }
4187 }
4188
4189 \stex_deactivate_macro:Nn \notation {module~environments}
4190 \stex_every_module:n {\stex_reactivate_macro:N \notation}
4191 \stex_sms_allow_escape:N \notation

```

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

\stex\_notation\_parse:n requires the above keys, \l\_stex\_get\_symbol\_arity\_int, and \l\_stex\_get\_symbol\_args\_tl

```

4192 \cs_new_protected:Nn \stex_notation_parse:n {
4193   \tl_if_empty:NF \l_stex_key_op_tl {

```

```

4194   \tl_set:Nx \l_stex_key_op_tl { \exp_not:N\maincomp {
4195     \exp_args:No \exp_not:n \l_stex_key_op_tl
4196   } }
4197 }
4198 \seq_clear:N \l__stex_notations_precs_seq
4199 \tl_clear:N \l_stex_notation_args_tl
4200 \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0 {
4201   \__stex_notations_const_precs:
4202   \tl_if_empty:NT \l_stex_key_op_tl {
4203     \tl_set:Nn \l_stex_key_op_tl { \maincomp{#1} }
4204   }
4205 }{
4206   \__stex_notations_fun_precs:
4207   \str_set:Nn \l__stex_notations_missing_str {#1}
4208   \tl_clear:N \l_stex_notations_missing_tl
4209   \stex_map_args:N \__stex_notations_add_missing_args:nn
4210   \tl_if_empty:NT \l_stex_key_op_tl {
4211     \hbox_set:Nn \l_tmpa_box {
4212       \str_set:Nn \l_stex_current_symbol_str {}
4213       \cs_set:Npn \l_tmpa_cs ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
4214       \cs_set:Npn \maincomp ##1 {
4215         \tl_gset:Nn \l_stex_key_op_tl { \maincomp{##1} }
4216         ##1
4217       }
4218       \cs_set:Npn \argsep ##1 ##2 {##1 ##2}
4219       \cs_set:Npn \argmap ##1 ##2 ##3 {##1 ##3}
4220       \cs_set:Npn \argarraymap ##1 ##2 ##3 ##4 {
4221         ##1 ##2
4222       }
4223       \stex_suppress_html:n{$\l_tmpa_cs abcdefghj$}
4224     }
4225   }
4226 }
4227 \exp_args:NNe
4228 \tl_set:Nn \l_stex_notation_macrocode_cs {
4229   \STEXInternalNotation
4230   { \l_stex_key_variant_str }
4231   { \l__stex_notations_oppref_tl }
4232   { \l_stex_key_intent_str }
4233   { \l_stex_notation_args_tl }
4234   {
4235     \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0
4236     { \exp_not:n { \maincomp{ #1 } } }
4237     { \exp_not:n { #1 } \l__stex_notations_missing_tl }
4238   }
4239 }
4240 \stex_debug:nn{notation}{Notation:~\meaning\l_stex_notation_macrocode_cs}
4241 }
4242
4243 \cs_new_protected:Nn \__stex_notations_const_precs: {
4244   \str_if_empty:NTF \l_stex_key_prec_str {
4245     \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4246   }{
4247     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
```

```

4248     \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4249   }{
4250     \tl_set_eq:NN \l__stex_notations_oppref_tl \l_stex_key_prec_str
4251   }
4252 }
4253 }
4254
4255 \cs_new_protected:Nn \__stex_notations_fun_precs: {
4256   \str_if_empty:NTF \l_stex_key_prec_str {
4257     \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4258   }{
4259     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4260       \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4261     }{
4262       \tl_set_eq:NN \l__stex_notations_oppref_tl \l_stex_key_prec_str
4263     }
4264   }
4265   \str_if_empty:NTF \l_stex_key_prec_str {
4266     \tl_set:Nn \l__stex_notations_oppref_tl { 0 }
4267     \int_step_inline:nn \l_stex_get_symbol_arity_int {
4268       \seq_put_right:Nn \l__stex_notations_precs_seq {0}
4269     }
4270   }{
4271     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4272       \stex_debug:nn{notation}{No~brackets}
4273       \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4274       \int_step_inline:nn \l_stex_get_symbol_arity_int {
4275         \exp_args:NNo \seq_put_right:Nn \l__stex_notations_precs_seq \infprec
4276       }
4277     } \__stex_notations_parse_precs:
4278   }
4279   \__stex_notations_do_argnames:
4280 }
4281
4282 \cs_new_protected:Nn \__stex_notations_parse_precs: {
4283   \stex_debug:nn{notation}{parsing~precedence~\l_stex_key_prec_str}
4284   \seq_set_split:NnV \l__stex_notations_seq ; \l_stex_key_prec_str
4285   \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4286     \tl_set_eq:NN \l__stex_notations_oppref_tl \l__stex_notations_str
4287     \seq_pop_left:NNT \l__stex_notations_seq \l__stex_notations_str {
4288       \exp_args:NNo \seq_set_split:NnV \l__stex_notations_seq
4289       {\tl_to_str:n{x}} \l__stex_notations_str
4290     }
4291   }{
4292     \tl_set:No \l__stex_notations_oppref_tl { 0 }
4293   }
4294   \int_step_inline:nn \l_stex_get_symbol_arity_int {
4295     \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4296       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_str
4297     }{
4298       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_oppref_tl
4299     }
4300   }
4301 }

```

```

4302 \cs_new_protected:Nn \__stex_notations_do_argnames: {
4303   \tl_clear:N \l_stex_notation_args_tl
4304   \stex_map_args:N \__stex_notations_do_argname:nn
4305 }
4306 }
4307
4308 \cs_new_protected:Nn \__stex_notations_do_argname:nn {
4309   \clist_if_empty:NTF \l_stex_key_intent_args_clist {
4310     \tl_put_right:Nx \l_stex_notation_args_tl {
4311       #1#2{\seq_item:Nn \l_stex_notations_precs_seq #1} {
4312         \str_if_empty:NF \l_stex_key_intent_str {#1}
4313       }
4314     }
4315   }{
4316     \tl_put_right:Nx \l_stex_notation_args_tl {
4317       #1#2{\seq_item:Nn \l_stex_notations_precs_seq #1}
4318       {\c_dollar_str\clist_item:Nn \l_stex_key_intent_args_clist 1}
4319     }
4320     \clist_pop:NN \l_stex_key_intent_args_clist \l_tmpa_tl
4321   }
4322 }
4323
4324 \cs_new:Nn \__stex_notations_add_missing_args:nn {
4325   \exp_args:NNe \str_if_in:NnF \l_stex_notations_missing_str {\c_hash_str\c_hash_str#1} {
4326     \tl_put_right:Nn \l_stex_notations_missing_tl{\STEXinvis{## #1}}
4327   }
4328 }

```

(End of 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: {
4329   \stex_check_term:n{
4330     \str_set:Nn \l_stex_current_symbol_str {test}
4331     \cs_set:Npn \comp {##1 {##1}}
4332     \stex_debug:nn{check_terms}{Checking~notation...}
4333     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}){
4334       \stex_notation_make_args:
4335     }
4336   }
4337 }
4338 }
4339
4340 \cs_new_protected:Nn \stex_notation_add: {
4341   \stex_module_add_notation:oeoo{
4342     \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4343     }\l_stex_key_variant_str
4344     {\int_use:N \l_stex_get_symbol_arity_int}
4345     \l_stex_notation_macrocode_cs
4346     \l_stex_key_op_tl
4347 }
4348
4349 \cs_new_protected:Nn \stex_notation_do_html:n {
4350   \hbox{\stex_annotation_invisible:nn {
4351     shtml:notation={#1},

```

```

4352     shtml:notationfragment={\l_stex_key_variant_str},
4353     shtml:precedence={\l__stex_notations_opprec_tl},
4354     shtml:argprecs={\seq_use:Nn \l__stex_notations_precs_seq ,}
4355 }{
4356     \cs_set_protected:Npn \argsep ##1 ##2 {
4357         \stex_annotation:nn{shtml:argsep={}{}{
4358             ##1 ##2
4359         }
4360     }
4361     \cs_set_protected:Npn \argmap ##1 ##2 ##3 {
4362         \cs_set:Npn \l__stex_notations_map_cs: #####1 { ##2 }
4363         \stex_annotation:nn{shtml:argmap={}{}{
4364             \l__stex_notations_map_cs:{##1} ##3
4365         }
4366     }
4367     \cs_set_protected:Npn \maincomp {
4368         \do_comp:nNn {maincomp}\compemph@uri
4369     }
4370     $
4371     \str_set:Nx \l_stex_current_symbol_str {#1}
4372     \stex_annotation:nn{shtml:notationcomp={}{}{
4373         \exp_args:Nne \use:nn {
4374             \l_stex_notation_macrocode_cs {}
4375         }
4376         \l_stex_map_args:N \l__stex_notations_make_arg_html:nn
4377     }
4378     }
4379     $
4380     \tl_if_empty:NF \l_stex_key_op_tl {
4381         $
4382         \str_set:Nx \l_stex_current_symbol_str {#1}
4383         \stex_annotation:nn{shtml:notationopcomp={}{}{
4384             \l_stex_term_oms:nnn{\l_stex_key_variant_str}{}{\l_stex_key_op_tl}
4385         }
4386         $
4387     }
4388 }
4389 }
4390
4391 \cs_new:Nn \l__stex_notations_make_arg_html:nn {
4392 %   \str_case:nnF #2 {
4393 %       a {{
4394 %           \stex_annotation:nn{shtml:argnum=#1a}{x},
4395 %           \stex_annotation:nn{shtml:argnum=#1b}{x}
4396 %       }}
4397 %       B {{
4398 %           \stex_annotation:nn{shtml:argnum=#1a}{x},
4399 %           \stex_annotation:nn{shtml:argnum=#1b}{x}
4400 %       }}
4401 %   }{
4402 %   {
4403 %       \stex_annotation:nn{shtml:argnum=#1}{x}
4404 %   }
4405 % }

```

```

4406 }
4407
4408 \cs_new:Nn \_stex_notation_make_args: {
4409   \_stex_map_notation_args:N \_stex_notations_make_arg:nnnn
4410 }
4411
4412
4413 \cs_new:Nn \_stex_notations_make_arg:nnnn {
4414   \str_case:nnF #2 {
4415     a {
4416       a\c_math_subscript_token{#1,1},
4417       a\c_math_subscript_token{#1,2}
4418     }
4419     B {
4420       B\c_math_subscript_token{#1,1},
4421       B\c_math_subscript_token{#1,2}
4422     }
4423   }{
4424     \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}
4425     {{#2}\c_math_subscript_token{#1}}
4426   }
4427 }

```

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

### `\setnotation`

```

\_stex_notation_set_default:n
4428 \cs_new_protected:Npn \setnotation #1 #2 {
4429   \stex_get_symbol:n{#1}
4430   \cs_if_exist:cTF{l_stex_notation_}
4431     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4432     _#2_cs
4433   }{
4434     \tl_set_eq:Nc \l_stex_notation_macrocode_cs {l_stex_notation_
4435       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4436       _#2_cs
4437     }
4438     \cs_if_exist:cTF{l_stex_notation_}
4439       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4440       _op_#2_cs
4441   }{
4442     \tl_set_eq:Nc \l_stex_key_op_tl {l_stex_notation_
4443       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4444       _op_#2_cs
4445     }
4446   }{
4447     \tl_clear:N \l_stex_key_op_tl
4448   }
4449   \_stex_notation_set_default:n{
4450     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4451   }
4452 }{
4453   \msg_error:nnxx{stex}{unknownnotation}{#2}{
4454     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str

```

```

4455     }
4456   }
4457 }
4458
4459 \cs_new_protected:Nn \_stex_notation_set_default:n{
4460   \stex_if_in_module:TF{
4461     \stex_module_add_notation:eooeo{#1}{}
4462     {int_use:N \l_stex_get_symbol_arity_int}
4463     \l_stex_notation_macrocode_cs
4464     \l_stex_key_op_tl
4465   }{
4466     \cs_set_eq:cN {\l_stex_notation_
4467       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4468       __cs}\l_stex_notation_macrocode_cs
4469     \tl_if_empty:NF \l_stex_key_op_tl {
4470       \cs_set_eq:cN{\l_stex_notation_
4471         \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4472         _op__cs}\l_stex_key_op_tl
4473     }
4474   }
4475 }
```

(End of definition for `\setnotation` and `\_stex_notation_set_default:n`. These functions are documented on page 91.)

### \varnotation

```

4476 \stex_new_stylable_cmd:nnnn {varnotation} { s m 0{} m} {
4477   \stex_keys_set:nn{notation}{#3}
4478   \stex_get_var:n{#2}
4479   \str_set_eq:NN \l_stex_key_name_str \l_stex_get_symbol_name_str
4480   \stex_notation_parse:n{#4}
4481   \stex_if_check_terms:T{ \_stex_notation_check: }
4482   \_stex_vardecl_notation_macro:
4483   \IfBooleanTF{#1{
4484     \_stex_notation_set_default:n{\l_stex_get_symbol_name_str}
4485   }{}}
4486   \group_begin:
4487   \tl_set_eq:NN \thisvarname \l_stex_get_symbol_name_str
4488   \tl_clear:N \thisstyle
4489   \str_set_eq:NN \thisnotationvariant\l_stex_key_variant_str
4490   \def\thisnotation{
4491     $ \let\l_stex_current_symbol_str\thisvarname
4492       \def\comp{\_varcomp}\exp_args:Nne \use:n{n{\l_stex_notation_macrocode_cs{}}}{}
4493       \_stex_notation_make_args:
4494     }$
4495   }
4496   \stex_style_apply:
4497   \group_end:
4498 }
```

(End of definition for `\varnotation`. This function is documented on page 95.)

### \symdef

```

4499 \stex_keys_define:nnnn{symdef}{}{}{decl,notation}
4500 }
```

```

4501 \cs_new_protected:Nn \_stex_symdef_styledefs: {
4502   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
4503   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
4504   \tl_set_eq:NN \thistype \l_stex_key_type_tl
4505   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
4506   \tl_set_eq:NN \thisargs \l_stex_key_args_str
4507   \tl_clear:N \thisstyle
4508   \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4509   \def\thisnotation{
4510     $ \let\l_stex_current_symbol_str \thisdecluri
4511       \def\comp{\_comp}\exp_args:Nne \use:n{n{ \l_stex_notation_macrocode_cs}{}
4512         \l_stex_notation_make_args:
4513       }$}
4514   }
4515 }
4516
4517 \stex_new_stylable_cmd:nnnn {symdef} { m 0{} m} {
4518   \stex_keys_set:nn{symdef}{#2}
4519   \str_set:Nx \l_stex_mroname_str { #1 }
4520   \stex_symdecl_top:n{#1}
4521   \stex_debug:nn{symdef}{Doing~\l_stex_current_module_str ? \l_stex_key_name_str}
4522   \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
4523   \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
4524   \stex_notation_parse:n{#3}
4525   \stex_debug:nn{Here!}{\meaning\l_stex_notation_args_tl}
4526   \stex_notation_check:
4527   \stex_notation_add:
4528   \stex_if_do_html:T{
4529     \stex_notation_do_html:n{ \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4530   }
4531   \stex_if_smsmode:F{
4532     \group_begin:
4533     \stex_symdef_styledefs:
4534     \stex_style_apply:
4535     \group_end:
4536   }
4537   \stex_smsmode_do:
4538 }{}
4539
4540 \stex_deactivate_macro:Nn \symdef {module~environments}
4541 \stex_every_module:n {\stex_reactivate_macro:N \symdef}
4542 \stex_sms_allow_escape:N \symdef

```

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

`\stex_do_default_notation_op:`

```

4543 \cs_new_protected:Nn \stex_do_default_notation: {
4544   \stex_do_default_notation_op:
4545   \tl_if_empty:NTF \l_stex_current_args_tl {
4546     \tl_clear:N \l__stex_notations_args_tl
4547   }{
4548     \__stex_notations_make_name:
4549     \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
4550     \tl_set:Nx \l__stex_notations_args_tl {

```

```

4551     \_stex_map_args:N \__stex_notations_augment_arg:nn
4552 }
4553 \tl_put_right:Nn \l_stex_default_notation {\comp{}} % 1: variant
4554 \seq_clear:N \l_tmpa_seq
4555 \int_step_inline:nn \l_stex_current_arity_str {
4556     \seq_put_right:Nn \l_tmpa_seq {#### ##1}
4557 }
4558 \tl_put_right:Nx \l_stex_default_notation {
4559     \seq_use:Nn \l_tmpa_seq {\mathpunct{\comp{,}}}}
4560 }
4561 \tl_put_right:Nn \l_stex_default_notation {\comp{}}
4562 }
4563 \tl_set:Nx \l_stex_default_notation {\STEXInternalNotation{}{0}{}{\l__stex_notations_args_}
4564     \exp_args:No \exp_not:n \l_stex_default_notation
4565 }
4566 }
4567
4568 \cs_new:Nn \__stex_notations_augment_arg:nn {
4569     #1#2{0}{}
4570 }
4571
4572 \cs_new_protected:Nn \__stex_notations_make_name: {
4573     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq ? \l_stex_current_symbol_str
4574     \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
4575     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq / \l__stex_notations_name_str
4576     \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
4577 }
4578
4579 \cs_new_protected:Nn \stex_do_default_notation_op: {
4580     \__stex_notations_make_name:
4581     \tl_set:Nx \l_stex_default_notation {\exp_not:N \maincomp{ \exp_not:N \mathrm{f} \l__stex_no_}
4582 }

```

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

### \STEXInternalNotation

```

4583 % 1: variant 2: operator precedence 3: intent 4: arguments 5: code 6: next
4584
4585 \cs_new_protected:Npn \STEXInternalNotation #1 #2 #3 #4 #5 #6 {
4586     \__stex_notations_process_notation:nnnnnn{#1}{#2}{#3}{#4}{#5}{%
4587         \l__stex_notations_code_tl
4588         #6
4589     }
4590 }
4591
4592 \cs_new_protected:Npn \__stex_notations_process_notation:nnnnnn #1 #2 #3 #4 {
4593     \tl_if_empty:nTF{#4}{%
4594         \__stex_notations_simple:nnnnn{#1}{#2}{#3}{%
4595             \__stex_notations_complex:nnnnnn{#1}{#2}{#3}{#4}{%
4596             }%
4597         }%
4598     }%
4599
4600 \cs_new_protected:Nn \__stex_notations_simple:nnnnn {

```

```

4601 \stex_debug:nn{Notation~code}{\tl_to_str:n{#4}}
4602 \tl_set:Nn \l__stex_notations_code_tl {
4603   \cs_set:Npn \l__stex_notations_cs {
4604     \stex_maybe_brackets:nn{#2}{
4605       \stex_term_oms_or_omv:nnn{#1}{#3}{#4}
4606     }
4607   }
4608   \l__stex_notations_cs
4609 }
4610 \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4611 #5
4612 }
4613
4614 \cs_new_protected:Nn \__stex_notations_complex:nnnnnn {
4615   \stex_debug:nn{Notation~code}{\tl_to_str:n{#5}}
4616   \int_zero:N \l_tmpa_int
4617   \tl_set:Nn \l__stex_notations_pre_tl {\cs_set_eq:NN \stex_term_oma_or_omb:nnn \stex_term}
4618   \tl_set:Nn \l__stex_notations_code_tl {
4619     \cs_generate_from_arg_count:NNnn \l__stex_notations_cs \cs_set:Npn \l_tmpa_int
4620     {
4621       \stex_maybe_brackets:nn{#2}{
4622         \stex_term_oma_or_omb:nnn{#1}{#3}{#4}
4623         \bool_set_false:N \l_stex_brackets_dones_bool
4624         #5
4625       }
4626     }
4627   }
4628   \l__stex_notations_cs
4629 }
4630 \tl_set:Nn \l__stex_notations_after_tl{
4631   \exp_args:NNo
4632   \tl_put_left:Nn \l__stex_notations_code_tl \l__stex_notations_pre_tl
4633   \tl_put_left:Nx \l__stex_notations_code_tl {
4634     \int_set:Nn \l_tmpa_int {\int_use:N \l_tmpa_int}
4635   }
4636   \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4637 #6
4638 }
4639 \__stex_notations_parse_notation_args:nnnw #4 \__stex_notations_args_end:
4640 }
4641
4642 \cs_new_protected:Npn \__stex_notations_parse_notation_args:nnnw #1 #2 #3 #4 #5 \__stex_no
4643   \tl_if_empty:nTF{#5}{
4644     \__stex_notations_add_last:nnnnn{#1}{#2}{#3}{#4}{#5}
4645   }{
4646     \__stex_notations_add_next:nnnnnn{#1}{#2}{#3}{#4}{#5}
4647   }
4648 }
4649
4650 \cs_new_protected:Nn \__stex_notations_add_next:nnnnnn {
4651   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#6}
4652   \__stex_notations_parse_notation_args:nnnw #5 \__stex_notations_args_end:
4653 }
4654

```

```

4655 \cs_new_protected:Nn \__stex_notations_add_last:nnnnn {
4656   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#5}
4657   \l__stex_notations_after_tl
4658 }
4659
4660 \cs_new_protected:Nn \__stex_notations_add:nnnnn {
4661   \int_incr:N \l_tmpa_int
4662   \str_case:nn{#2}{f
4663     i {
4664       \tl_put_right:Nn \l__stex_notations_code_tl {
4665         {\__stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4666       }
4667     }
4668     b {
4669       \tl_set:Nn \l__stex_notations_pre_tl {
4670         \cs_set_eq:NN \__stex_term_oma_or_omb:nnn \__stex_term_omb:nnn
4671       }
4672       \tl_put_right:Nn \l__stex_notations_code_tl {
4673         {\__stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4674       }
4675     }
4676     a {
4677       \tl_put_right:Nn \l__stex_notations_code_tl {
4678         {\__stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4679       }
4680     }
4681     B {
4682       \tl_set:Nn \l__stex_notations_pre_tl {
4683         \cs_set_eq:NN \__stex_term_oma_or_omb:nnn \__stex_term_omb:nnn
4684       }
4685       \tl_put_right:Nn \l__stex_notations_code_tl {
4686         {\__stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4687       }
4688     }
4689   }
4690 }

```

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

## a/B-mode argument handling

### \argsep

```

4691 \cs_new_protected:Nn \__stex_notations_check_aB_arg:Nn {
4692   \exp_args:Ne \cs_if_eq:NNF {\tl_head:n{#2}}
4693   \__stex_term_arg_aB:nnnnn {
4694     \msg_error:nnx{\stex}{error/assocarg}{\tl_to_str:n{#1}}
4695   }
4696 }
4697
4698 \cs_new_protected:Npn \argsep #1 #2 {
4699   \__stex_notations_check_aB_arg:Nn\argsep{#1}
4700   \stex_pseudogroup_with:nn{\__stex_term_do_aB_clist:} {
4701     \tl_set:Nn \__stex_term_do_aB_clist: {
4702       \seq_use:Nn \l_stex_aB_args_seq {#2}

```

```

4703      }
4704      #1
4705    }
4706 }

```

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

### `\argmap`

```

4707 \cs_new_protected:Npn \argmap #1 #2 #3 {
4708   \__stex_notations_check_aB_arg:Nn\argmap{#1}
4709   \stex_pseudogroup_with:nn{
4710     \_stex_term_do_aB_clist:
4711     \__stex_notations_map_cs:
4712   }{
4713     \cs_set:Npn \__stex_notations_map_cs: ##1 { #2 }
4714     \tl_set:Nn \_stex_term_do_aB_clist: {
4715       \seq_clear:N \l_tmpa_seq
4716       \seq_map_inline:Nn \l_stex_aB_args_seq {
4717         \tl_if_eq:nnTF{##1}{\ellipses}{
4718           \seq_put_right:Nn \l_tmpa_seq \ellipses
4719         }{
4720           \seq_put_right:Nx \l_tmpa_seq {
4721             \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
4722           }
4723         }
4724       }
4725       \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4726       \seq_use:Nn \l_stex_aB_args_seq {#3}
4727     }
4728   #1
4729 }
4730 }

```

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

### `\argarraymap`

```

4731 \int_new:N \l__stex_notations_clist_count_int
4732 \cs_new_protected:Npn \argarraymap #1 #2 #3 #4 {
4733   \__stex_notations_check_aB_arg:Nn\argarraymap{#1}
4734   \stex_pseudogroup_with:nn{
4735     \_stex_term_do_aB_clist:
4736     \__stex_notations_map_cs:
4737   }{
4738     \cs_set:Npn \__stex_notations_map_cs: ##1 { #3 }
4739     \int_set:Nn \l__stex_notations_clist_count_int {\exp_args:No\clist_count:nf\tl_to_str:nf}
4740     \tl_set:Nn \_stex_term_do_aB_clist: {
4741       \tl_clear:N \l_tmpa_tl
4742       \int_zero:N \l_tmpa_int
4743       \seq_map_inline:Nn \l_stex_aB_args_seq {
4744         \int_incr:N \l_tmpa_int
4745         \int_compare:nNnT \l_tmpa_int > \l__stex_notations_clist_count_int {
4746           \int_set:Nn \l_tmpa_int 1
4747         }
4748         \tl_put_right:Nx \l_tmpa_tl {
4749           \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }

```

```

4750           \clist_item:nn{#4}\l_tmpa_int
4751       }
4752   }
4753   \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4754   \begin{array}{#2}
4755     \l_tmpa_t1
4756   \end{array}
4757 }
4758 #1
4759 }
4760 }
```

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

### 13.8.3 Variables

```

4761 <@=stex_vars>
4762 \vardef
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
5001
5002
5003
5004
5005
5006
5007
5008
5009
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039
5040
5041
5042
5043
5044
5045
5046
5047
5048
5049
5050
5051
5052
5053
5054
5055
5056
5057
5058
5059
5060
5061
5062
5063
5064
5065
5066
5067
5068
5069
5070
5071
5072
5073
5074
5075
5076
5077
5078
5079
5080
5081
5082
5083
5084
5085
5086
5087
5088
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5099
5099
5100
5101
5102
5103
5104
5105
5106
5107
5108
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119
5120
5121
5122
5123
5124
5125
5126
5127
5128
5129
5130
5131
5132
5133
5134
5135
5136
5137
5138
5139
5140
5141
5142
5143
5144
5145
5146
5147
5148
5149
5150
5151
5152
5153
5154
5155
5156
5157
5158
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5199
5200
5201
5202
5203
5204
5205
5206
5207
5208
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228
5229
5230
5231
5232
5233
5234
5235
5236
5237
5238
5239
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249
5249
5250
5251
5252
5253
5254
5255
5256
5257
5258
5259
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378
5379
5379
5380
5381
5382
5383
5384
5385
5386
5387
5388
5389
5389
5390
5391
5392
5393
5394
5395
5396
5397
5398
5399
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5409
5410
5411
5412
5413
5414
5415
5416
5417
5418
5419
5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5429
5430
5431
5432
5433
5434
5435
5436
5437
5438
5439
5439
5440
5441
5442
5443
5444
5445
5446
5447
5448
5449
5449
5450
5451
5452
5453
5454
5455
5456
5457
5458
5459
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468
5469
5469
5470
5471
5472
5473
5474
5475
5476
5477
5478
5479
5479
5480
5481
5482
5483
5484
5485
5486
5487
5488
5489
5489
5490
5491
5492
5493
5494
5495
5496
5497
5498
5499
5499
5500
5501
5502
5503
5504
5505
5506
5507
5508
5509
5509
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528
5529
5529
5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548
5549
5549
5550
5551
5552
5553
5554
5555
5556
5557
5558
5559
5559
5560
5561
5562
5563
5564
5565
5566
5567
5568
5569
5569
5570
5571
5572
5573
5574
5575
5576
5577
5578
5579
5579
5580
5581
5582
5583
5584
5585
5586
5587
5588
5589
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5599
5599
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5609
5610
5611
5612
5613
5614
5615
5616
5617
5618
5619
5619
5620
5621
5622
5623
5624
5625
5626
5627
5628
5629
5629
5630
5631
5632
5633
5634
5635
5636
5637
5638
5639
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649
5649
5650
5651
5652
5653
5654
5655
5656
5657
5658
5659
5659
5660
5661
5662
5663
5664
5665
5666
5667
5668
5669
5669
5670
5671
5672
5673
5674
5675
5676
5677
5678
5679
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5699
5699
5700
5701
5702
5703
5704
5705
5706
5707
5708
5709
5709
5710
5711
5712
5713
5714
5715
5716
5717
5718
5719
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729
5729
5730
5731
5732
5733
5734
5735
5736
5737
5738
5739
5739
5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5749
5750
5751
5752
5753
5754
5755
5756
5757
5758
5759
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778
5779
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5799
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5809
5810
5811
5812
5813
5814
5815
5816
5817
5818
5819
5819
5820
5821
5822
5823
5824
5825
5826
5827
5828
5829
5829
5830
5831
5832
5833
5834
5835
5836
5837
5838
5839
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5849
5850
5851
5852
5853
5854
5855
5856
5857
5858
5859
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999
5999
6000
6001
6002
6003
6004
6005
6006
6007
6008
6009
6009
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6049
6050
6051
6052
6053
6054
6055
6056
6057
6058
6059
6059
6060
6061
6062
6063
6064
6065
6066
6067
6068
6069
6069
6070
6071
6072
6073
6074
6075
6076
6077
6078
6079
6079
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6089
6090
6091
6092
6093
6094
6095
6096
6097
6098
6099
6099
6100
6101
6102
6103
6104
6105
6106
6107
6108
6109
6109
6110
6111
6112
6113
6114
6115
6116
6117
6118
6119
6119
6120
6121
6122
6123
6124
6125
6126
6127
6128
6129
6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6139
6140
6141
6142
6143
6144
6145
6146
6147
6148
6149
6149
6150
6151
6152
6153
6154
6155
6156
6157
6158
6159
6159
6160
6161
6162
6163
6164
6165
6166
6167
6168
6169
6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179
6179
6180
6181
6182
6183
6184
6185
6186
6187
6188
6189
6189
6190
6191
6192
6193
6194
6195
6196
6197
6198
6199
6199
6200
6201
6202
6203
6204
6205
6206
6207
6208
6209
6209
6210
6211
6212
6213
6214
6215
6216
6217
6218
6219
6219
6220
6221
6222
6223
6224
6225
6226
6227
6228
6229
6229
6230
6231
6232
6233
6234
6235
6236
6237
6238
6239
6239
6240
6241
6242
6243
6244
6245
6246
6247
6248
6249
6249
6250
6251
6252
6253
6254
6255
6256
6257
6258
6259
6259
6260
6261
6262
6263
6264
6265
6266
6267
6268
6269
6269
6270
6271
6272
6273
6274
6275
6276
6277
6278
6279
6279
6280
6281
6282
6283
6284
6285
6286
6287
6288
6289
6289
6290
6291
6292
6293
6294
6295
6296
6297
6298
6299
6299
6300
6301
6302
6303
6304
6305
6306
6307
6308
6309
6309
6310
6311
6312
6313
6314
6315
6316
6317
6318
6319
6319
6320
6321
6322
6323
6324
6325
6326
6327
6328
6329
6329
6330
6331
6332
6333
6334
6335
6336
6337
6338
6339
6339
6340
6341
6342
6343
6344
6345
6346
6347
6348
6349
6349
6350
6351
6352
6353
6354
6355
6356
6357
6358
6359
6359
6360
6361
6362
6363
6364
6365
6366
6367
6368
6369
6369
6370
6371
6372
6373
6374
6375
6376
6377
6378
6379
6379
6380
6381
6382
6383
6384
6385
6386
6387
6388
6389
6389
6390
6391
6392
6393
6394
6395
6396
6397
6398
6399
6399
6400
6401
6402
6403
6404
6405
6406
6407
6408
6409
6409
6410
6411
6412
6413
6414
6415
6416
6417
6418
6419
6419
6420
6421
6422
6423
6424
6425
6426
6427
6428
6429
6429
6430
6431
6432
6433
6434
6435
6436
6437
6438
6439
6439
6440
6441
6442
6443
6444
6445
6446
6447
6448
6449
6449
6450
6451
6452
6453
6454
6455
6456
6457
6458
6459
6459
6460
6461
6462
6463
6464
6465
6466
6467
6468
6469
6469
6470
6471
6472
6473
6474
6475
6476
6477
6478
6479
6479
6480
6481
6482
6483
6484
6485
6486
6487
6488
6489
6489
6490
6491
6492
6493
6494
6495
6496
6497
6498
6499
6499
6500
6501
6502
6503
6504
6505
6506
6507
6508
6509
6509
6510
6511
6512
6513
6514
6515
6516
6517
6518
6519
6519
6520
6521
6522
6523
6524
6525
6526
6527
6528
6529
6529
6530
6531
6532
6533
6534
6535
6536
6537
6538
6539
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6549
6550
6551
6552
6553
6554
6555
6556
6557
6558
6559
6559
6560
6561
6562
6563
6564
6565
6566
6567
6568
6569
6569
6570
6571
6572
6573
6574
6575
6576
6577
6578
6579
6579
6580
6581
6582
6583
6584
6585
6586
6587
6588
6589
6589
6590
6591
6592
6593
6594
6595
6596
6597
6598
6599
6599
6600
6601
6602
6603
6604
6605
6606
6607
6608
6609
6609
6610
6611
6612
6613
6614
6615
6616
6617
6618
6619
6619
6620
6621
6622
6623
6624
6625
6626
6627
6628
6629
6629
6630
6631
6632
6633
6634
6635
6636
6637
6638
6639
6639
6640
6641
6642
6643
6644
6645
6646
6647
6648
6649
6649
6650
6651
6652
6653
6654
6655
6656
6657
6658
6659
6659
6660
6661
6662
6663
6664
6665
6666
6667
6668
6669
6669
6670
6671
6672
6673
6674
6675
6676
6677
6678
6679
6679
6680
6681
6682
6683
6684
6685
6686
6687
6688
6689
6689
6690
6691
6692
6693
6694
6695
6696
6697
6698
6699
6699
6700
6701
6702
6703
6704
6705
6706
6707
6708
6709
6709
6710
6711
6712
6713
6714
6715
6716
6717
6718
6719
6719
6720
6721
6722
6723
6724
6725
6726
6727
6728
6729
6729
6730
6731
6732
6733
6734
6735
6736
6737
6738
6739
6739
6740
6741
6742
6743
6744
6745
6746
6747
6748
6749
6749
6750
6751
6752
6753
6754
6755
6756
6757
6758
6759
6759
6760
6761
6762
6763
6764
6765
6766
6767
6768
6769
6769
6770
6771
6772
6773
6774
6775
6776
6777
6778
6779
6779
6780
6781
6782
6783
6784
6785
6786
6787
```

```

4798 \def\thisnotation{
4799   $\let\l_stex_current_symbol_str\thisvarname
4800   \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_mmacrocode_{}}{
4801     \l_stex_notation_make_args:
4802   }$}
4803 }
4804 \stex_style_apply:
4805 \group_end:\ignorespaces
4806 }{}}
4807
4808 \cs_new_protected:Nn \__stex_vars_add: {
4809   \exp_args:NNNo \exp_args:NNnx
4810   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4811     {\l_stex_macroname_str}
4812     {\l_stex_key_name_str}
4813     {\int_use:N \l_stex_get_symbol_arity_int}
4814     {\l_stex_get_symbol_args_tl}
4815     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4816     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4817     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4818     \stex_invoke_symbol:
4819   }
4820 }
4821
4822 \cs_new_protected:Nn \__stex_vars_macro: {
4823   \tl_set:cx{\l_stex_macroname_str}{%
4824     \l_stex_invoke_variable:nnnnnnN
4825     {\l_stex_key_name_str}
4826     {\int_use:N \l_stex_get_symbol_arity_int}
4827     {\l_stex_get_symbol_args_tl}
4828     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4829     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4830     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4831     \stex_invoke_symbol:
4832   }
4833 }
4834
4835 \cs_new_protected:Nn \__stex_vars_html: {
4836   \stex_if_do_html:T {
4837     \hbox\bgroup\exp_args:Ne \stex_annotation_invisible:nn {
4838       \shtml:vardef = {\l_stex_key_name_str},
4839       \shtml:args = {\l_stex_key_args_str}
4840       \str_if_empty:NF \l_stex_macroname_str {,
4841         \shtml:macroname={\l_stex_macroname_str}
4842       }
4843       \str_if_empty:NF \l_stex_key_assoc_str {,
4844         \shtml:assocotype={\l_stex_key_assoc_str}
4845       }
4846       \str_if_empty:NF \l_stex_key_role_str {,
4847         \shtml:role={\l_stex_key_role_str}
4848       }
4849       \str_if_empty:NF \l_stex_key_reorder_str {,
4850         \shtml:reorderargs={\l_stex_key_reorder_str}
4851     }

```

```

4852     \bool_if:NT \l_stex_vars_bind_bool {
4853         shtml:bind={}
4854     }
4855 }{
4856     \stex_annotate_force_break:n{
4857         \bool_set_true:N \stex_in_invisible_html_bool
4858         \tl_if_empty:NF \l_stex_key_type_tl {
4859             \stex_annotate:nn{shtml:type={}}{$\l_stex_key_type_tl$}
4860         }
4861         \tl_if_empty:NF \l_stex_key_def_tl {
4862             \stex_annotate:nn{shtml:definiens={}}{$\l_stex_key_def_tl$}
4863         }
4864         \tl_if_empty:NF \l_stex_key_return_tl{
4865             \exp_args:Nno \use:n{
4866                 \cs_generate_from_arg_count:NNnn \l_stex_vars_cs
4867                 \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4868                 \tl_set:Nx \l_stex_vars_args_tl {\l_stex_map_args:N \l_stex_return_args:nn}
4869                 $ \stex_annotate:nn{shtml:returnrtype={}}{
4870                     \exp_after:wN \l_stex_vars_cs \l_stex_vars_args_tl!}$}
4871         }
4872         \tl_if_empty:NF \l_stex_key_argtypes_clist {
4873             \stex_annotate:nn{shtml:argtypes={}}{
4874                 \stex_annotate_force_break:n{
4875                     \clist_map_inline:Nn \l_stex_key_argtypes_clist {
4876                         $ \stex_annotate:nn{shtml:type={}}{\#\#1}$}
4877                     }
4878                 }
4879             }
4880         }
4881     }
4882 }
```

(End of definition for `\vardef`. This function is documented on page 95.)

### `\_stex_vardecl_notation_macro:`

```

4885 \cs_new_protected:Nn \stex_vardecl_notation_macro: {
4886     \tl_set_eq:cN {l_stex_notation_
4887         \l_stex_key_name_str _ 
4888         \l_stex_key_variant_str _cs
4889     }\l_stex_notation_macrocode_cs
4890     \cs_if_exist:cF {l_stex_notation_\l_stex_key_name_str __cs} {
4891         \tl_set_eq:cN{l_stex_notation_\l_stex_key_name_str __cs}
4892             \l_stex_notation_macrocode_cs
4893     }
4894     \tl_if_empty:NF \l_stex_key_op_tl {
4895         \tl_set_eq:cN {l_stex_notation_\l_stex_key_name_str _op_
4896             \l_stex_key_variant_str _cs}\l_stex_key_op_tl
4897         \cs_if_exist:cF {l_stex_notation_\l_stex_key_name_str _op__cs} {
4898             \cs_set_eq:cN{l_stex_notation_\l_stex_key_name_str _op__cs}
4899                 \l_stex_key_op_tl
4900         }
4901     }
4902 }
```

(End of definition for `\stex_vardecl_notation_macro`. This function is documented on page 125.)

```
\stex_get_symbol_or_var:n
  \stex_get_var:n 4903 \cs_new_protected:Nn \__stex_vars_set_vars:nnnnnnN {
  4904   \stex_debug:nn{symbols}{Variable~#1~found}
  4905   \cs_set:Npn \_stex_variable:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {}
  4906   \str_clear:N \l_stex_get_symbol_mod_str
  4907   \str_set:Nn \l_stex_get_symbol_name_str {#1}
  4908   \int_set:Nn \l_stex_get_symbol_arity_int {#2}
  4909   \tl_set:Nn \l_stex_get_symbol_args_tl {#3}
  4910   \tl_set:Nn \l_stex_get_symbol_def_tl {#4}
  4911   \tl_set:Nn \l_stex_get_symbol_type_tl {#5}
  4912   \tl_set:Nn \l_stex_get_symbol_return_tl {#6}
  4913   \tl_set:Nn \l_stex_get_symbol_invoke_cs {#7}
  4914 }
  4915
  4916 \cs_new_protected:Nn \__stex_vars_get_var:n {
  4917   \prop_map_inline:Nn \l_stex_variables_prop {
  4918     \__stex_vars_check_var:nnnnnnnnN {#1} ##2
  4919   }
  4920 }
  4921
  4922 \cs_new_protected:Nn \__stex_vars_check_var:nnnnnnnnN {
  4923   \str_if_eq:nnTF{#1}{#2}{%
  4924     \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}}
  4925   }{
  4926     \str_if_eq:nnT{#1}{#3}{%
  4927       \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}}
  4928     }
  4929   }
  4930 }
  4931
  4932 \cs_new_protected:Nn \stex_get_var:n {
  4933   \str_clear:N \l_stex_get_symbol_name_str
  4934   \__stex_vars_get_var:n{#1}
  4935   \str_if_empty:NT \l_stex_get_symbol_name_str {
  4936     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
  4937   }
  4938 }
  4939
  4940 \cs_new_protected:Nn \stex_get_symbol_or_var:n {
  4941   \str_clear:N \l_stex_get_symbol_name_str
  4942   \__stex_vars_get_var:n{#1}
  4943   \str_if_empty:NT \l_stex_get_symbol_name_str {
  4944     \stex_debug:nn{symbols}{No~variable~#1~found}
  4945     \stex_get_symbol:n{#1}
  4946   }
  4947 }
```

(End of definition for `\stex_get_symbol_or_var:n` and `\stex_get_var:n`. These functions are documented on page 125.)

```
\svar
  4948 \NewDocumentCommand \svar {O{} m}{}
```

```

4949 \group_begin:
4950   \tl_if_empty:nTF{#1}{
4951     \str_set:Nn \l_stex_current_symbol_str {#2}
4952   }{
4953     \str_set:Nn \l_stex_current_symbol_str {#1}
4954   }
4955   \bool_if:NTF \l_stex_allow_semantic_bool{
4956     \tl_clear:N \l_stex_current_term_tl
4957     \l_stex_term_omv:nnn{}{}{\l_varcomp{#2}}
4958   }{
4959     \msg_error:nnxx{stex}{error/notallowed}{Variable}{\l_stex_current_symbol_str}
4960   }
4961 \group_end:
4962 }
```

(End of definition for `\svar`. This function is documented on page 95.)

### 13.8.4 Sequences

```

4963 <@=stex_seqs>
\varseq
4964 \stex_new_stylable_cmd:nnnn {varseq}{m 0{} m m} {
4965   \stex_keys_set:nn{symdef}{#2}
4966   \str_set:Nx \l_stex_macroname_str { #1 }
4967   \str_if_empty:NT \l_stex_key_name_str {
4968     \str_set:Nx \l_stex_key_name_str { #1 }
4969   }
4970   \str_if_empty:NT \l_stex_key_args_str {
4971     \str_set:Nn \l_stex_key_args_str {1}
4972   }
4973   \stex_symdecl_do:
4974
4975   \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4976   \clist_set:Nn \l_stex_seqs_range_clist {#3}
4977   \tl_if_empty:NTF \l_stex_key_op_tl {
4978     \stex_notation_parse:n{#4}
4979     \tl_clear:N \l_stex_key_op_tl
4980   }{
4981     \stex_notation_parse:n{#4}
4982   }
4983   \stex_if_do_html:T \__stex_seqs_html:
4984   \stex_if_check_terms:T \__stex_seqs_check_terms:
4985   \__stex_seqs_add:
4986   \__stex_seqs_macro:
4987   \stex_if_check_terms:T \__stex_notation_check:
4988   \__stex_vardecl_notation_macro:
4989   \group_begin:
4990   \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4991   \tl_clear:N \thisstyle
4992   \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4993   \def\thisnotation{
4994     $ \let\l_stex_current_symbol_str\thisvarname
4995       \def\comp{\l_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}
4996       \__stex_seqs_make_args:
```

```

4997     }$  

4998 }  

4999 \stex_style_apply:  

5000 \group_end:\ignorespaces  

5001 }{}  

5002  

5003 \cs_new_protected:Nn \__stex_seqs_add: {  

5004   \exp_args:NNNo \exp_args:NNnx  

5005   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {  

5006     {\l_stex_mroname_str}  

5007     {\l_stex_key_name_str}  

5008     {\int_use:N \l_stex_get_symbol_arity_int}  

5009     {\l_stex_get_symbol_args_tl}  

5010     {\exp_args:No \exp_not:n \l_stex_key_def_tl}  

5011     {\exp_args:No \exp_not:n \l_stex_seqs_range_clist}  

5012     {\exp_args:No \exp_not:n \l_stex_key_return_tl}  

5013     \stex_invoke_sequence:  

5014   }  

5015 }  

5016  

5017 \cs_new_protected:Nn \__stex_seqs_macro: {  

5018   \tl_set:cx{\l_stex_mroname_str}{  

5019     \__stex_invoke_variable:nnnnnnN  

5020     {\l_stex_key_name_str}  

5021     {\int_use:N \l_stex_get_symbol_arity_int}  

5022     {\l_stex_get_symbol_args_tl}  

5023     {\exp_args:No \exp_not:n \l_stex_key_def_tl}  

5024     {\exp_args:No \exp_not:n \l_stex_seqs_range_clist}  

5025     {\exp_args:No \exp_not:n \l_stex_key_return_tl}  

5026     \stex_invoke_sequence:  

5027   }  

5028 }  

5029  

5030 \cs_new_protected:Nn \__stex_seqs_make_args: { \TODO }  

5031 \cs_new_protected:Nn \__stex_seqs_check_terms: { \TODO }  

5032  

5033 \cs_new_protected:Nn \__stex_seqs_html: {  

5034   \exp_args:Ne \stex_annotation_invisible:nn {  

5035     \shtml:vareq = {\l_stex_key_name_str},  

5036     \shtml:args = {\l_stex_key_args_str}  

5037     \str_if_empty:NF \l_stex_mroname_str {,  

5038       \shtml:macroname={\l_stex_mroname_str}  

5039     }  

5040     \str_if_empty:NF \l_stex_key_assoc_str {,  

5041       \shtml:assocotype={\l_stex_key_assoc_str}  

5042     }  

5043     \str_if_empty:NF \l_stex_key_role_str {,  

5044       \shtml:role={\l_stex_key_role_str}  

5045     }  

5046     \str_if_empty:NF \l_stex_key_reorder_str {,  

5047       \shtml:reorderargs={\l_stex_key_reorder_str}  

5048     }  

5049 }{\hbox\bgroup  

5050   \stex_annotation_force_break:n{
```

```

5051   \tl_if_empty:NF \l_stex_key_type_tl {
5052     \stex_annotation:nn{shml:type={}}{$\l_stex_key_type_tl$}
5053   }
5054   \tl_if_empty:NF \l_stex_key_def_tl {
5055     \stex_annotation:nn{shml:definiens={}}{$\l_stex_key_def_tl$}
5056   }
5057   \tl_if_empty:NF \l_stex_key_return_tl{
5058     \exp_args:Nno \use:n{
5059       \cs_generate_from_arg_count:NNnn \l_stex_seqs_CS
5060       \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
5061       \tl_set:Nx \l_stex_seqs_args_tl {\l_stex_map_args:N \l_stex_return_args:nn}
5062       \stex_annotation:nn{shml:returntype={}}{
5063         $\exp_after:wN \l_stex_seqs_CS \l_stex_seqs_args_tl!$}
5064     }
5065   \tl_if_empty:NF \l_stex_key_argtypes_clist {
5066     \stex_annotation:nn{shml:argtypes={}}{
5067       \l_stex_annotation_force_break:n{
5068         \clist_map_inline:Nn \l_stex_key_argtypes_clist {
5069           \stex_annotation:nn{shml:type={}}{$\#1$}
5070         }
5071       }
5072     }
5073   }
5074 }
5075 \egroup
5076 }

```

(End of definition for `\varseq`. This function is documented on page 95.)

`\stex_invoke_sequence:`

```

5077 \cs_new_protected:Nn \stex_invoke_sequence: {
5078   \peek_charcode_remove:NTF !
5079   \peek_charcode:NTF [ \l_stex_seqs_do_op:w { \l_stex_seqs_do_op:w [] } ]
5080   } \l_stex_seqs_do_first:
5081 }
5082
5083 \cs_new_protected:Npn \l_stex_seqs_do_op:w [#1] {
5084   \cs_if_exist:cTF {\l_stex_notation_\l_stex_current_symbol_str _op_#1_cs} {
5085     \l_stex_maybe_brackets:nn{\neginfpref}{}
5086     \l_stex_term_oms_or_omv:nnn{#1}{}{ }
5087     {\use:c{\l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}}
5088   }
5089   \group_end:
5090 } {
5091   \l_stex_seqs_get_index_notation:n{#1}
5092   \peek_charcode:NTF [ \l_stex_seqs_doop_range:w { \l_stex_seqs_doop_range:w[] } ]
5093 }
5094 }
5095
5096 \cs_new_protected:Npn \l_stex_seqs_doop_range:w [#1] {
5097   \bool_set_true:N \l_stex_allow_semantic_bool
5098   \clist_clear:N \l_stex_seqs_clist
5099   \clist_map_function:NN \l_stex_current_type_tl \l_stex_seqs_doop_arg:n
5100   \stex_annotation:nn{

```

```

5101     shtml:term=OMV,
5102     shtml:head={\l_stex_current_symbol_str},
5103     shtml:notationid={}
5104   }{
5105     \l__stex_seqs_clist
5106   }
5107   \group_end:
5108 }
5109
5110 \cs_new_protected:Nn \__stex_seqs_doop_arg:n {
5111   \tl_if_eq:nnTF{#1}{\ellipses}{
5112     \clist_put_right:Nn \l__stex_seqs_clist {
5113       \ellipses
5114     }
5115   }{
5116     \clist_put_right:Nn \l__stex_seqs_clist {
5117       \exp_args:No \str_if_eq:nnTF \l_stex_current_arity_str {1} {
5118         \group_begin:
5119           \l__stex_seqs_cs \group_end: {#1}
5120       }{
5121         \group_begin:
5122           \l__stex_seqs_cs \group_end: #1
5123       }
5124     }
5125   }
5126 }
5127 }
5128
5129 \cs_new_protected:Nn \__stex_seqs_get_index_notation:n {
5130   \cs_if_exist:cTF {l_stex_notation_\l_stex_current_symbol_str _#1_cs} {
5131     \cs_set_eq:Nc \l__stex_seqs_cs {l_stex_notation_\l_stex_current_symbol_str _#1_cs}
5132   }{
5133     \stex_do_default_notation:
5134     \cs_set_eq:NN \l__stex_seqs_cs \l_stex_default_notation
5135   }
5136 }
5137
5138
5139 \cs_new:Nn \__stex_seqs_do_first_arg:n {{\exp_not:n{## #1}}}
5140
5141 \cs_new_protected:Nn \__stex_seqs_do_first: {
5142   \exp_args:Nnx \use:nn{
5143     \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs \cs_set:Npn
5144     \l_stex_current_arity_str }{
5145       \tl_set:Nn \exp_not:N \l__stex_seqs_first_args_tl {
5146         \int_step_function:nN \l_stex_current_arity_str \__stex_seqs_do_first_arg:n
5147       }
5148       \exp_not:N \__stex_seqs_do_first_next:
5149     }}
5150   \l__stex_seqs_cs
5151 }
5152
5153 \cs_new_protected:Nn \__stex_seqs_do_first_next: {
5154   \peek_charcode_remove:NTF ! {

```

```

5155     \peekCharCode:NNTF [ \__stex_seqs_do_one:w {\__stex_seqs_do_one:w []}
5156   }{
5157     \peekCharCode:NNTF [ \__stex_seqs_do_all:w {\__stex_seqs_do_all:w []}
5158   }
5159 }
5160
5161 \cs_new_protected:Npn \__stex_seqs_do_one:w [#1] {
5162   \__stex_seqs_get_index_notation:n[#1]
5163   \stex_debug:nn{HERE~seq~one}{\meaning\l__stex_seqs_cs^~J\meaning\l__stex_seqs_first_args_t}
5164   \exp_args:Nno\use:nn{\l__stex_seqs_cs\group_end:}\l__stex_seqs_first_args_tl
5165 }
5166
5167 \cs_new_protected:Npn \__stex_seqs_do_all:w [#1] {
5168   \stex_debug:nn{HERE~seq~all}{\meaning\l__stex_seqs_first_args_t}
5169   \exp_args:Nno\use:nn{\l__stex_invoke_notation:w [#1]}\l__stex_seqs_first_args_tl
5170 }

```

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

### \seqmap

```

5171 \cs_new_protected:Npn \seqmap #1 #2 {
5172   \symuse{Metatheory?sequence-expression}{\seqmap[#1]{#2}}%\l_tmpa_tl {#2}
5173 }

```

(End of definition for `\seqmap`. This function is documented on page 96.)

## 13.8.5 Expressions

5174 `<@=stex_expr>`

Various variables:

```

5175 \bool_new:N \l_stex_allow_semantic_bool
5176 \bool_set_true:N \l_stex_allow_semantic_bool
5177 \tl_new:N \l_stex_current_term_tl
5178 \tl_set:Nn \l_stex_every_symbol_tl {
5179   \bool_set_false:N \l_stex_allow_semantic_bool
5180 }

```

`\_stex_next_symbol:n`

```

5181 \tl_new:N \l__stex_expr_reset_tl
5182 \cs_new_protected:Nn \_stex_next_symbol:n {
5183   \tl_set:Nx \l_stex_every_symbol_tl {
5184     \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5185       \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
5186         \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5187       }
5188       \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5189         \exp_args:No \exp_not:n \l__stex_expr_reset_tl
5190       }
5191     }
5192     \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
5193       \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5194     }
5195   \exp_not:n{ \aftergroup \l__stex_expr_reset_tl }
5196   \exp_not:N \l_stex_every_symbol_tl

```

```

5197     \exp_not:n{ #1 }
5198   }
5199 }
5200 \cs_generate_variant:Nn \stex_next_symbol:n {e}

```

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

### `\STEXinvisible`

```

5201 \cs_new_protected:Npn \STEXinvisible #1 {
5202   \stex_annotation_invisible:n { #1 }
5203 }

```

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

## Invoking Semantic Macros

### `\stex_invoke_symbol:nnnnnnnN`

```

5204 \cs_new_protected:Nn \stex_invoke_symbol:nnnnnnnN {
5205   \bool_if:NTF \l_stex_allow_semantic_bool{
5206     \stex_if_html_backend:T{ifvmode\indent\fi}
5207     \__stex_expr_setup:nnnnnf{\comp}{#1?#2}{#3}{#4}{#7}{#6}
5208     \cs_set_eq:NN \stex_term_oms_or_omv:nnn \stex_term_oms:nnn
5209     \tl_put_right:Nn \l_stex_current_redo_tl{
5210       \cs_set_eq:NN \stex_term_oms_or_omv:nnn \stex_term_oms:nnn
5211     }
5212     #8
5213   }{
5214     \msg_error:nnxx{stex}{error/notallowed}{#1?#2}{\l_stex_current_symbol_str}
5215   }
5216 }
5217 \cs_generate_variant:Nn \stex_invoke_symbol:nnnnnnnN {ooxooooN}
5218
5219 \cs_new_protected:Nn \__stex_expr_setup:nnnnnn {
5220   \group_begin:
5221   \tl_clear:N \l_stex_return_notation_tl
5222   \tl_set:Nn \l_stex_current_redo_tl {
5223     \let \this \stex_current_this:
5224     \def\comp{#1}
5225     \def\maincomp{\comp}
5226     \str_set:Nn \l_stex_current_symbol_str {#2}
5227     \str_set:Nn \l_stex_current_arity_str{ #3 }
5228     \tl_set:Nn \l_stex_current_args_tl{ #4 }
5229     \tl_set:Nn \l_stex_current_return_tl{ #5 }
5230     \tl_set:Nn \l_stex_current_type_tl{ #6 }
5231     \tl_clear:N \l_stex_current_term_tl
5232   }
5233   \tl_put_right:Nx \l_stex_current_redo_tl {
5234     \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5235   }
5236   \l_stex_current_redo_tl
5237 }

```

(End of definition for `\stex_invoke_symbol:nnnnnnnN`. This function is documented on page ??.)

```

\_stex_invoke_variable:nnnnnn
5238 \cs_new_protected:Nn \_stex_invoke_variable:nnnnnnN {
5239   \bool_if:NTF \l_stex_allow_semantic_bool{
5240     \stex_if_html_backend:T{\ifvmode\indent\fi}
5241     \__stex_expr_setup:nnnnnn{\_varcomp}{#1}{#2}{#3}{#6}{#5}
5242     \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
5243     \tl_put_right:Nn \l_stex_current_redo_tl {
5244       \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
5245     }
5246     #7
5247   }{
5248     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
5249   }
5250 }

```

(End of definition for `\_stex_invoke_variable:nnnnnn`. This function is documented on page ??.)

### \symuse

```

5251 \cs_new_protected:Npn \symuse #1 {
5252   \stex_get_symbol:n{#1}
5253   \exp_args:Nno \use:n {\_stex_invoke_symbol:ooooooN
5254   \l_stex_get_symbol_mod_str
5255   \l_stex_get_symbol_name_str
5256   {\int_use:N \l_stex_get_symbol_arity_int}
5257   \l_stex_get_symbol_args_tl
5258   \l_stex_get_symbol_def_tl
5259   \l_stex_get_symbol_type_tl
5260   \l_stex_get_symbol_return_tl}
5261   \l_stex_get_symbol_invoke_cs
5262 }

```

(End of definition for `\symuse`. This function is documented on page 89.)

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

```

5263 \cs_new_protected:Nn \stex_invoke_symbol: {
5264   \stex_debug:nn{expressions}{Invoking-\l_stex_current_symbol_str}
5265   \mode_if_math:TF \__stex_expr_invoke_math: \__stex_expr_invoke_text:
5266 }
5267
5268 \cs_new_protected:Nn \__stex_expr_invoke_text: {
5269   \stex_debug:nn{expressions}{text-mode}
5270   \peek_charcode_remove:NTF ! \__stex_expr_invoke_op_custom:n \__stex_expr_invoke_custom:n
5271 }
5272
5273 \cs_new_protected:Nn \__stex_expr_invoke_math: {
5274   \stex_debug:nn{expressions}{math-mode}
5275   \peek_charcode_remove:NTF !
5276     % operator
5277     \peek_charcode_remove:NTF * \__stex_expr_invoke_op_custom:n {
5278       % op notation
5279       \peek_charcode:NTF [ \__stex_expr_invoke_op_notation:w {
5280         \__stex_expr_invoke_op_notation:w []
5281       }

```

```

5282     }
5283   }{
5284     \peek_charcode_remove:NTF * \__stex_expr_invoke_custom:n {
5285       % normal
5286       \peek_charcode:NTF [ \_stex_invoke_notation:w {
5287         \_stex_invoke_notation:w []
5288       }
5289     }
5290   }
5291 }

```

Notations:

```

5292 \cs_new_protected:Npn \_stex_invoke_notation:w [#1] {
5293   \stex_debug:nn{expressions}{using~notation~#1~for~\l_stex_current_symbol_str}
5294   \cs_if_exist:cTF{\l_stex_notation_\l_stex_current_symbol_str _#1_cs} {
5295     \tl_if_empty:NTF \l_stex_current_return_tl {
5296       \stex_debug:nn{expressions}{return~empty}
5297       \use:c{\l_stex_notation_\l_stex_current_symbol_str _#1_cs}{\group_end:\_stex_eat_exclamation_point}
5298     }{
5299       \stex_debug:nn{expressions}{return?}
5300       \exp_after:wN\exp_after:wN\exp_after:wN
5301       \__stex_expr_invoke_return_maybe:n
5302       \exp_after:wN\exp_after:wN\exp_after:wN
5303       {\cs:w \l_stex_notation_\l_stex_current_symbol_str _#1_cs \cs_end: {}}
5304     }
5305   }{
5306     \stex_do_default_notation:
5307     \tl_if_empty:NTF \l_stex_current_return_tl {
5308       \l_stex_default_notation{\group_end:\_stex_eat_exclamation_point:}
5309     }{
5310       \exp_after:wN
5311       \__stex_expr_invoke_return_maybe:n
5312       \exp_after:wN
5313       {\l_stex_default_notation {}}
5314     }
5315   }
5316 }
5317
5318 \cs_new_protected:Npn \__stex_expr_invoke_op_notation:w [#1] {
5319   \stex_debug:nn{expressions}{op~notation~for~\l_stex_current_symbol_str}
5320   \cs_if_exist:cTF{\l_stex_notation_\l_stex_current_symbol_str _op_#1_cs} {
5321     \_stex_maybe_brackets:nn{\neginfpref}{%
5322       \_stex_term_oms_or_omv:nnn{#1}{}%
5323       {\use:c{\l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}}%
5324     }
5325     \group_end:
5326   }{
5327     \int_compare:nNnTF \l_stex_current_arity_str = 0 {
5328       \tl_clear:N \l_stex_current_return_tl
5329       \_stex_invoke_notation:w [#1]
5330     }{
5331       \stex_do_default_notation_op:
5332       \_stex_maybe_brackets:nn{\neginfpref}{%
5333         \_stex_term_oms_or_omv:nnn{#1}{}%

```

```

5334     {\l_stex_default_notation}
5335   }
5336   \group_end:
5337 }
5338 }
5339 }

Return:
5340 \cs_new_protected:Nn \__stex_expr_invoke_return_maybe:n {
5341   \tl_clear:N \l_stex_expr_return_args_tl
5342   \tl_set:Nn \l_stex_expr_return_this_tl {\#1}
5343   \exp_args:Nnx \use:n {
5344     \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
5345     \cs_set:Npn \l_stex_current_arity_str } {
5346       \int_step_function:nN \l_stex_current_arity_str \__stex_expr_return_arg:n
5347       \__stex_expr_invoke_return_next:
5348     }
5349   \__stex_expr_ret_cs
5350 }

5351 \cs_new:Nn \__stex_expr_return_arg:n {
5352   \tl_put_right:Nn \exp_not:N \l_stex_expr_return_args_tl {{#### #1}}
5353 }
5354 }

5355 \cs_new_protected:Nn \__stex_expr_invoke_return_next: {
5356   \peek_charcode_remove:NTF !
5357   \exp_after:wN \l_stex_expr_return_this_tl \l_stex_expr_return_args_tl \group_end:
5358 } \__stex_expr_invoke_return:
5359 }

5360 \cs_new_protected:Nn \__stex_expr_invoke_return: {
5361   \tl_set:Nx \l_stex_expr_return_this_tl {
5362     \__stex_expr_return_notation:n {
5363       \exp_after:wN \exp_after:wN \exp_after:wN
5364       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
5365         \exp_after:wN \l_stex_expr_return_this_tl \l_stex_expr_return_args_tl
5366       }
5367     }
5368   }
5369 }
5370 \stex_debug:nn{return}{Notation:~\meaning\l_stex_expr_return_this_tl}
5371 \tl_put_left:Nx \l_stex_expr_return_this_tl {
5372   \group_begin:\exp_args:No \exp_not:n \l_stex_current_redo_tl
5373 }
5374 \exp_args:Nnx \use:n {
5375 \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
5376   \cs_set:Npn \l_stex_current_arity_str } {
5377     \exp_args:No \exp_not:n \l_stex_current_return_tl
5378   }
5379 \stex_debug:nn{return}{
5380   \meaning\__stex_expr_ret_cs^J
5381   \meaning\l_stex_expr_return_this_tl^J
5382   \exp_args:No \exp_not:n \l_stex_expr_return_args_tl^J
5383 }
5384 \exp_args:Nnx \use:nn {

```

```

5386     \exp_after:wN \group_end: \_stex_expr_ret_cs
5387 }{
5388     \exp_args:No \exp_not:n \l__stex_expr_return_args_tl
5389 {
5390     \exp_args:No \exp_not:n \l__stex_expr_return_this_tl
5391     \group_end:
5392 }
5393 }
5394 }
5395
5396 \cs_new_protected:Nn \_stex_expr_return_notation:n {
5397     \tl_if_empty:NTF \l_stex_return_notation_tl { #1 }{
5398         \l_stex_return_notation_tl
5399     }
5400 }
5401 }
5402

```

Custom Notations:

```

5403 \cs_new_protected:Nn \_stex_expr_invoke_op_custom:n {
5404     \stex_debug:nn{expressions}{custom~op}
5405     \bool_set_true:N \l_stex_allow_semantic_bool
5406     \stex_term_oms_or_omv:nnn{}{}{\maincomp{#1}}
5407     \group_end:
5408 }
5409
5410 \int_new:N \l_stex_expr_arg_counter_int
5411 \cs_new_protected:Nn \_stex_expr_invoke_custom:n {
5412     \stex_debug:nn{custom}{custom~notation~for~\l_stex_current_symbol_str}
5413     \stex_pseudogroup:nn{
5414         \bool_set_true:N \l_stex_allow_semantic_bool
5415         \prop_gclear:N \l_stex_expr_customs_prop
5416         \seq_gclear:N \l_stex_expr_customs_seq
5417         \int_gzero:N \l_stex_expr_arg_counter_int
5418         \tl_if_empty:NTF \l_stex_current_args_tl {
5419             \exp_after:wN \_stex_expr_add_prop_arg:nnw \l_stex_current_args_tl \_stex_args_end:
5420             \cs_set_eq:NN \arg \_stex_expr_arg:n
5421         }
5422         \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
5423         \cs_set_eq:NN \_stex_expr_do_ab_next:nnn \stex_term_oma:nnn
5424         \stex_map_args:N \_stex_expr_check_b:nn
5425         \_stex_expr_do_ab_next:nnn{}{}{#1}
5426 }
5427     \prop_if_exist:NT \l_stex_expr_customs_prop {
5428         \prop_gset_from_keyval:Nn \exp_not:N \l_stex_expr_customs_prop {
5429             \prop_to_keyval:N \l_stex_expr_customs_prop
5430         }
5431     }
5432     \int_gset:Nn \l_stex_expr_arg_counter_int { \int_use:N \l_stex_expr_arg_counter_int}
5433     \seq_if_exist:NT \l_stex_expr_customs_seq {
5434         \seq_gset_split:Nnn \exp_not:N \l_stex_expr_customs_seq , {
5435             \seq_use:Nn \l_stex_expr_customs_seq ,
5436         }
5437 }

```

```

5438 }
5439 % TODO check that all arguments are present
5440 \group_end:
5441 }
5442
5443 \cs_new_protected:Npn \__stex_expr_add_prop_arg:nw #1 #2 #3\_stex_args_end: {
5444   \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {}
5445   \seq_gput_right:Nn \l__stex_expr_customs_seq {#2}
5446   \tl_if_empty:nF{#3}{\__stex_expr_add_prop_arg:nw #3 \_stex_args_end:}
5447 }
5448
5449 \cs_new:Nn \__stex_expr_check_b:nn {
5450   \str_case:nn #2 {
5451     b {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \stex_term_omb:nnn}
5452     B {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \stex_term_omb:nnn}
5453   }
5454 }
5455
5456 \NewDocumentCommand \__stex_expr_arg:n { s O{} m } {
5457   \IfBooleanTF #1 {
5458     \stex_annotation_invisible:n{
5459       \__stex_expr_arg_inner:nn{#2}{#3}
5460     }
5461   }{
5462     \__stex_expr_arg_inner:nn{#2}{#3}
5463   }
5464 }
5465
5466 \cs_new_protected:Nn \__stex_expr_arg_inner:nn {
5467   \tl_if_empty:nTF{#1} {
5468     \int_gincr:N \l__stex_expr_arg_counter_int
5469     \exp_args:Ne \__stex_expr_check:nTF{ \int_use:N \l__stex_expr_arg_counter_int }{
5470       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl
5471     }{
5472       \__stex_expr_arg_inner:nn{}
5473     }{ #2 }
5474   }{
5475     \__stex_expr_check:nTF {#1} {
5476       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5477     }{
5478       \exp_args:No \str_case:nnTF \l_tmpb_tl {
5479         {a} {
5480           \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {
5481             \l_tmpa_tl X
5482           }
5483           \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n { \tl_count:N \l_tmpa_tl + 1 } }
5484         }
5485         {B} {
5486           \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {
5487             \l_tmpa_tl X
5488           }
5489           \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n { \tl_count:N \l_tmpa_tl + 1 } }
5490         }
5491     }{

```

```

5492     \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5493   }{
5494     \msg_error:nxxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5495   }
5496 }
5497 }
5498 }
5499
5500 \prg_new_if:NNn \__stex_expr_check:n {TF} {
5501   \exp_args:NNe \prop_get:NnNTF \l_stex_expr_customs_prop {#1} \l_tmpa_tl {
5502     \tl_set:Nx \l_tmpb_tl {\seq_item:Nn \l_stex_expr_customs_seq {#1} }
5503     \tl_if_empty:NTF \l_tmpa_tl {
5504       \exp_args:NNe \prop_gput:Nnn \l_stex_expr_customs_prop
5505         { #1 }{X}
5506       \exp_args:No \str_case:nnF \l_tmpb_tl {
5507         {a} {
5508           \tl_set:Nx \l_tmpa_tl{ #1 1 }
5509         }
5510         {B} {
5511           \tl_set:Nx \l_tmpa_tl{ #1 1 }
5512         }
5513       }{
5514         \tl_set:Nx \l_tmpa_tl{ #1 }
5515       }
5516       \prg_return_true:
5517     }{
5518       \prg_return_false:
5519     }
5520   }{
5521     \msg_error:nxxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5522     \prg_return_false:
5523   }
5524 }
5525
5526 % #1 argnum #2 argmode #3 code
5527 \cs_new_protected:Nn \__stex_expr_arg_do:nnn {
5528   \stex_debug:nn{custom}{Doing~argument~#1~of~mode~#2:~\tl_to_str:n{#3}}
5529   \group_begin:
5530     \bool_set_true:N \l_stex_allow_semantic_bool
5531     \stex_term_arg:nnn {#2}{#1}{#3}
5532   \group_end:
5533 }
5534 \cs_generate_variant:Nn \__stex_expr_arg_do:nnn {oon}

```

(End of definition for `\stex_invoke_symbol`. This function is documented on page 131.)

## Argument Handling and Annotating

```

\stex_term_arg:nnnnn
\stex_term_arg:nnn
5535 % 1: argnum 2: argmode 3: precedence 4: argname 5: code
5536 \cs_new_protected:Nn \stex_term_arg:nnnnn {
5537   \group_begin:
5538     \str_clear:N \l_stex_current_symbol_str
5539     \tl_clear:N \l_stex_current_term_tl

```

```

5540   \int_set:Nn \l_stex_notation_downprec { #3 }
5541   \bool_set_true:N \l_stex_allow_semantic_bool
5542   \_stex_term_arg:nnn {#2}{#1}{
5543     \tl_if_empty:nTF{#4}{
5544       #5
5545     }{
5546       \stex_annotation:nn{mml:arg={#4}}{#5}
5547     }
5548   }
5549   \group_end:
5550 }
5551
5552 \cs_new_protected:Nn \_stex_term_arg:nnn {
5553   \stex_annotation:nn{ shtml:arg={#2}, shtml:argmode={#1}}{
5554     \_stex_annotation_force_break:n{ #3 }
5555   }
5556 }

```

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

### `\_stex_term_arg_aB:nnnn`

```

5557 \tl_set:Nn \__stex_expr_do_aB_clist: {
5558   \seq_use:Nn \l_stex_aB_args_seq {
5559     \mathpunct{\comp{,}}
5560   }
5561 }
5562 \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5563
5564 \int_new:N \l__stex_expr_count_int
5565 \cs_new_protected:Nn \_stex_term_arg_aB:nnnn {
5566   \tl_if_empty:nTF{#5} {
5567     \_stex_term_arg:nnnn{#1}{#2}{#3}{#4}{}
5568   }{
5569     \seq_clear:N \l_stex_aB_args_seq
5570     \int_zero:N \l__stex_expr_count_int
5571     \clist_map_inline:nn{#5} {
5572       \__stex_expr_aB_arg:nnnnn{##1}{#1}{#2}{#3}{#4}
5573     }
5574     \_stex_term_do_aB_clist:
5575   }
5576 }
5577
5578 % 1: code 2: argnum 3: argmode 4: precedence 5: argname
5579 \cs_new_protected:Npn \__stex_expr_aB_arg:nnnnn #1 {
5580   \int_incr:N \l__stex_expr_count_int
5581   \__stex_expr_is_varseq:nTF{#1} {
5582     \exp_after:wN \exp_after:wN \exp_after:wN
5583     \__stex_expr_assoc_seq:nnnnnnn
5584     \exp_after:wN
5585     \__stex_expr_gobble:nnnnnnnn #1 \__stex_expr_end:
5586   }{
5587     \__stex_expr_is_seqmap:nTF{#1} {
5588       \exp_args:NNe \use:nn \__stex_expr_do_seqmap:nnnnnn {\tl_tail:n{#1}}

```

```

5589 }
5590     \_stex_expr_aB_simple_arg:nnnnn{#1}
5591 }
5592 }
5593 }
5594
5595 \cs_new:Npn \_stex_expr_gobble:nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8 #9 \_stex_expr_end: {
5596     {#2} #3 {#6}
5597 }
5598
5599 \cs_new_protected:Nn \_stex_expr_aB_simple_arg:nnnnn{
5600     \seq_put_right:Nx \l_stex_aB_args_seq {
5601         \stex_term_arg:nnnnn{#2}\int_use:N\l_stex_expr_count_int}{#3}{#4}{#5}{#6}
5602         \exp_not:n{
5603             \tl_set_eq:NN \stex_term_do_aB_clist: \_stex_expr_do_aB_clist:
5604             #1
5605         }
5606     }
5607 }
5608 }
5609
5610 \cs_new_protected:Nn \stex_is_sequentialized:n {
5611     \group_begin: #1 \group_end:
5612 }

```

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

```

5613 \prg_new_conditional:Nnn \_stex_expr_is_varseq:n {TF} {
5614     \int_compare:nNnTF {\tl_count:n{#1}} = 1 {
5615         \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_head:n{#1}}
5616         \stex_invoke_variable:nnnnnN {
5617             \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No\tl_item:nn{#1}{8}}
5618                 \stex_invoke_sequence:
5619                 \prg_return_true:\prg_return_false:
5620             } \prg_return_false:
5621         } \prg_return_false:
5622     }
5623
5624 \prg_new_conditional:Nnn \_stex_expr_is_seqmap:n {TF} {
5625     \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
5626         \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\seqmap}
5627             \prg_return_true:\prg_return_false:
5628         } \prg_return_false:
5629     }

```

Sequence variable:

```

5630 % 1: name 2: arity 3: clist 4: argnum 5: argmode 6: precedence 7: argname
5631 \cs_new_protected:Nn \_stex_expr_assoc_seq:nnnnnnn {
5632     \group_begin:
5633         \seq_clear:N \l_stex_aB_args_seq
5634         \_stex_expr_assoc_make_seq:nnn{#1}{#3}{#2}
5635     \exp_args:NNe \use:nn \group_end: {
5636         \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5637             \stex_is_sequentialized:n{
5638                 \stex_term_arg:nnnnn{#4}\int_use:N\l_stex_expr_count_int}{#5}{#6}{#7}{#8}

```

```

5639   \bool_set_true:N \l_stex_allow_semantic_bool
5640   \str_set:Nn \exp_not:N \l_stex_current_symbol_str
5641     {\l_stex_current_symbol_str}
5642   \tl_if_empty:NF \l_stex_current_term_tl {
5643     \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5644       \exp_args:No \exp_not:n \l_stex_current_term_tl
5645     }
5646   }
5647   \stex_annotate:nn{
5648     shtml:term=OMV,
5649     shtml:head={#1},
5650     shtml:notationid={}
5651   }{
5652     \stex_annotate_force_break:n{
5653       \stex_term_do_aB_clist:
5654     }
5655   }
5656 }
5657 }
5658 }
5659 }
5660 }
5661 % #1: name, #2: clist, #3:arity
5662 \cs_new_protected:Nn \__stex_expr_assoc_make_seq:nnn {
5663   \cs_if_exist:cTF{\l_stex_notation_#1__cs} {
5664     \cs_set_eq:Nc \l_stex_expr_cs {\l_stex_notation_#1__cs}
5665   }{
5666     \stex_do_default_notation:
5667     \cs_set_eq:NN \l_stex_expr_cs \l_stex_default_notation
5668   }
5669 \clist_map_inline:nn{#2}{%
5670   \tl_if_eq:nnTF{##1}{\ellipses} {
5671     \seq_put_right:Nn \l_stex_aB_args_seq { ##1 }
5672   }{
5673     \int_compare:nNnTF {#3} = 1 {
5674       \tl_set:Nn \l_stex_expr_iarg_tl { ##1 }
5675     }{
5676       \tl_set:Nn \l_stex_expr_iarg_tl { ##1 }
5677     }
5678     \seq_put_right:Nx \l_stex_aB_args_seq {
5679       \group_begin:
5680         \exp_not:n {
5681           \tl_set_eq:NN \stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5682           \def\comp{\varcomp}
5683           \str_set:Nn \l_stex_current_symbol_str{#1}
5684         }
5685         \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
5686         \exp_after:wN \exp_after:wN \exp_after:wN {
5687           \exp_after:wN \l_stex_expr_cs \exp_after:wN \group_end: \l_stex_expr_iarg_tl
5688         }
5689       }
5690     }
5691   }
5692 }

```

(End of definition for \stex\_term\_arg\_aB:nnnnn. This function is documented on page ??.)

## Term HTML Annotations

```
\_stex_term_oms_or_omv:nnn
\stex_term_oms:nnn
\stex_term_omv:nnn
5744 \cs_new_protected:Nn \stex_eat_exclamation_point: {
5745   \peekCharCode_remove:NT ! {
5746     \stex_eat_exclamation_point:
5747   }
5748 }
5749
5750 \bool_new:N \stex_in_invisible_html_bool
5751 \bool_set_false:N \stex_in_invisible_html_bool
5752 \stex_if_html_backend:TF {
5753   % 1: variant 2: intent 3: code
5754   \cs_new_protected:Nn \stex_term_oms:nnn {
5755     \tl_if_empty:NTF \l_stex_current_term_tl {
5756       \stex_annotation:nn{
5757         shtml:term=OMID,
5758         shtml:head={\l_stex_current_symbol_str},
5759         shtml:notationid={#1},
5760       }{
5761         \stex_annotation_force_break:n{#3}
5762       }
5763     }{
5764       \__stex_expr_do_headterm:nn{#1}{#3}
5765     }
5766   }
5767 \cs_new_protected:Nn \stex_term_omv:nnn {
5768   \tl_if_empty:NTF \l_stex_current_term_tl {
5769     \stex_annotation:nn{
5770       shtml:term=OMV,
5771       shtml:head={\l_stex_current_symbol_str},
5772       shtml:notationid={#1}
5773     }{
5774       \stex_annotation_force_break:n{#3}
5775     }
5776   }{
5777     \__stex_expr_do_headterm:nn{#1}{#3}
5778   }
5779 }
5780 \cs_new_protected:Nn \__stex_expr_do_headterm:nn {
5781   \bool_if:NTF \stex_in_invisible_html_bool {
5782     {\bool_set_true:N \l_stex_allow_semantic_bool
5783       \ensuremath{\l_stex_current_term_tl}}
5784   }
5785 }{
5786   \stex_annotation:nn{
5787     shtml:term=complex,
5788     shtml:head={\l_stex_current_symbol_str},
5789     shtml:notationid={#1}
5790   }{
5791     \stex_annotation_force_break:n{
5792       \stex_annotation_invisible:nn{shtml:headterm={}}{
5793         {\bool_set_true:N \l_stex_allow_semantic_bool
5794           \ensuremath{\l_stex_current_term_tl}}
```

```

5795         }
5796     }
5797   }
5798 #2
5799   }
5800 }
5801 }
5802 }{
5803   \cs_new_protected:Nn \_stex_term_oms:nnn {#3}
5804   \cs_new_protected:Nn \_stex_term_omv:nnn {#3}
5805   \cs_new_protected:Nn \__stex_expr_do_headterm:nn { #2 }
5806 }
5807 \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn

```

(End of definition for `\_stex_term_oms_or_omv:nnn`, `\_stex_term_oms:nnn`, and `\_stex_term_omv:nnn`.  
These functions are documented on page ??.)

#### `\_stex_term_oma:nnn`

```

5808 \stex_if_html_backend:TF {
5809   \cs_new_protected:Nn \_stex_term_oma:nnn {
5810     \tl_if_empty:NTF \l_stex_current_term_tl {
5811       \stex_annotation:nn{
5812         shtml:term=OMA,
5813         shtml:head={\l_stex_current_symbol_str},
5814         shtml:notationid={#1}
5815     }{
5816       \_stex_annotation_force_break:n{#3}
5817     }
5818   }{
5819     \stex_annotation:nn{
5820       shtml:term=OMA,
5821       shtml:head={\l_stex_current_symbol_str},
5822       shtml:notationid={#1}
5823     }{
5824       \_stex_annotation_force_break:n{
5825         \stex_annotation_invisible:nn{shtml:headterm={} }{
5826           \bool_set_true:N \l_stex_allow_semantic_bool
5827           \l_stex_current_term_tl
5828         }
5829         #3
5830       }
5831     }
5832   }
5833 }
5834 }{
5835   \cs_new_protected:Nn \_stex_term_oma:nnn {#3}
5836 }

```

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

#### `\_stex_term_omb:nnn`

```

5837 \stex_if_html_backend:TF {
5838   \cs_new_protected:Nn \_stex_term_omb:nnn {
5839     \tl_if_empty:NTF \l_stex_current_term_tl {
5840       \stex_annotation:nn{

```

```

5841     shtml:term=OMBIND,
5842     shtml:head={\l_stex_current_symbol_str},
5843     shtml:notationid={#1}
5844   }{
5845     \_stex_annotation_force_break:n{#3}
5846   }
5847   }{
5848     \stex_annotation:nn{
5849       shtml:term=OMBIND,
5850       shtml:head={\l_stex_current_symbol_str},
5851       shtml:notationid={#1}
5852     }{
5853       \_stex_annotation_force_break:n{
5854         \stex_annotation_invisible:nn{shtml:headterm={}{{}
5855           \bool_set_true:N \l_stex_allow_semantic_bool
5856           \l_stex_current_term_tl
5857         }}}
5858       #3
5859     }
5860   }
5861 }
5862 }{
5863   \cs_new_protected:Nn \_stex_term_omb:nnn { #3 }
5864 }
5865

```

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

## Automated Bracketing

```

\infprec
\neginfprec
5866 \tl_const:Nx \infprec {\int_use:N \c_max_int}
5867 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}

```

(End of definition for `\infprec` and `\neginfprec`. These functions are documented on page 91.)

```

\dobrackets
5868 \int_new:N \l_stex_notation_downprec
5869 \int_set:Nn \l_stex_notation_downprec \infprec
5870 \tl_set:Nn \l_stex_expr_left_bracket_str (
5871 \tl_set:Nn \l_stex_expr_right_bracket_str )
5872 \bool_new:N \l_stex_brackets_dones_bool
5873
5874 \cs_new_protected:Nn \_stex_maybe_brackets:nn {
5875   \bool_if:NTF \l_stex_brackets_dones_bool {
5876     \bool_set_false:N \l_stex_brackets_dones_bool
5877     #2
5878   } {
5879     \stex_debug:nn{brackets}{#1}\int_eval:n \l_stex_notation_downprec?
5880     \int_compare:nNnTF { #1 } > \l_stex_notation_downprec {
5881       \%bool_if:NTF \l_stex_inpararray_bool { #2 }{
5882         \dobrackets {
5883           #2
5884         }
5885       }
5886     }
5887   }
5888 }

```

```

5885      %}
5886    }{
5887      #2
5888    }
5889  }
5890 }
5891 %\RequirePackage{scalerel}
5892 \cs_new_protected:Npn \dobrackets #1 {
5893   \%ThisStyle{\if D\m@switch
5894     % \exp_args:Nnx \use:nn
5895     % { \exp_after:wN \left\lvert l_stex_expr_left_bracket_str #1 }
5896     % { \exp_not:N\right\rvert l_stex_expr_right_bracket_str }
5897     % \else
5898     \group_begin:
5899     \%stex_pseudogroup_with:nn{\l_stex_brackets_dones_bool\l_stex_notation_downprec}{%
5900       \bool_set_true:N \l_stex_brackets_dones_bool
5901       \%int_set:Nn \l_stex_notation_downprec \infprec
5902       \mathopen{\cs_if_exist:NT\l_stex_current_symbol_str\comp
5903         \l_stex_expr_left_bracket_str
5904       }
5905     }
5906     #1
5907   \group_end:{}%
5908   \mathclose{\cs_if_exist:NT\l_stex_current_symbol_str\comp
5909     \l_stex_expr_right_bracket_str
5910   }
5911 \%fi}
5912 }

```

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

### \withbrackets

```

5913 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
5914   \%stex_pseudogroup_with:nn{\l_stex_expr_left_bracket_str\l_stex_expr_right_bracket_str}{%
5915     \tl_set:Nn \l_stex_expr_left_bracket_str { #1 }
5916     \tl_set:Nn \l_stex_expr_right_bracket_str { #2 }
5917     #3
5918   }
5919 }

```

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

### \dowithbrackets

```

5920 \cs_new_protected:Npn \dowithbrackets #1 #2 #3 {
5921   \withbrackets{#1}{#2}{\dobrackets{#3}}
5922 }

```

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

## Symname and Variants

```

\symname
  \sn 5923 \def\maincomp{\comp}
  \sns 5924
\Symname 5925 \stex_keys_define:nnnn{symname}{%
  \Sn
  \Sns
\symref
  \sr
\varref
\varname
\Varname

```

```

5926   \tl_clear:N \l_stex_key_pre_tl
5927   \tl_clear:N \l_stex_key_post_tl
5928   \%tl_clear:N \l_stex_key_proot_tl
5929 }{
5930   pre    .tl_set:N  = \l_stex_key_pre_tl ,
5931   post   .tl_set:N  = \l_stex_key_post_tl ,
5932   root   .code:n   = {}%.tl_set:N  = \l_stex_key_root_tl
5933 }{}
5934
5935 \NewDocumentCommand \symref { O{} m m} {
5936   \group_begin:
5937   \stex_keys_set:nn{symname}{#1}
5938   \stex_get_symbol:n{#2}
5939   \__stex_expr_do_ref:nNn{#3}\symrefemph@uri\stex_term_oms:nnn
5940 }
5941 \let\sr\symref
5942
5943 \NewDocumentCommand \symname { O{} m} {
5944   \group_begin:
5945   \stex_keys_set:nn{symname}{#1}
5946   \stex_get_symbol:n{#2}
5947   \__stex_expr_do_ref:nNn{
5948     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
5949   }\symrefemph@uri\stex_term_oms:nnn
5950 }
5951 \let\sn\symname
5952 \protected\def\sns{\symname[post=s]}
5953
5954 \NewDocumentCommand \Symname { O{} m} {
5955   \group_begin:
5956   \stex_keys_set:nn{symname}{#1}
5957   \stex_get_symbol: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   }\symrefemph@uri\stex_term_oms:nnn
5961 }
5962 \cs_new_protected:Nn \stex_capitalize:n {
5963   \uppercase{#1}
5964 }
5965 \let\Sn\Symname
5966 \protected\def\Sns{\Symname[post=s]}
5967
5968 \cs_new:Npn \stex_split_slash: #1/#2/#3\stex_args_end: {
5969   \tl_if_empty:nTF{#3}{
5970     #2
5971   }{
5972     \stex_split_slash: #2 / #3 \stex_args_end:
5973   }
5974 }
5975
5976 \NewDocumentCommand \varref { O{} m m} {
5977   \group_begin:
5978   \stex_keys_set:nn{symname}{#1}
5979   \stex_get_var:n{#2}

```

```

5980  \_stex_expr_do_ref:nNn{#3}\varemp@uri{
5981    \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5982    \def\comp{\_varcomp}
5983    \stex_term_omv:nnn
5984  }
5985 }
5986
5987 \NewDocumentCommand \varname { O{} m} {
5988   \group_begin:
5989   \stex_keys_set:nn{symname}{#1}
5990   \stex_get_var:n{#2}
5991   \_stex_expr_do_ref:nNn{
5992     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
5993   }\varemp@uri{
5994     \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5995     \def\comp{\_varcomp}
5996     \stex_term_omv:nnn
5997   }
5998 }
5999
6000 \NewDocumentCommand \Varname { O{} m} {
6001   \group_begin:
6002   \stex_keys_set:nn{symname}{#1}
6003   \stex_get_var:n{#2}
6004   \_stex_expr_do_ref:nNn{
6005     \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
6006   }\varemp@uri{
6007     \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
6008     \def\comp{\_varcomp}
6009     \stex_term_omv:nnn
6010   }
6011 }
6012
6013
6014 \cs_new_protected:Nn \_stex_expr_do_ref:nNn {
6015   \stex_if_html_backend:T{\ifvmode\indent\fi}
6016   \bool_if:NTF \l_stex_allow_semantic_bool{
6017     \str_set:Nx\l_stex_current_symbol_str
6018     {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6019     \str_if_in:NnT \l_stex_get_symbol_name_str / {
6020       \str_set:Nx \l_stex_get_symbol_name_str {
6021         \exp_after:wN \l_stex_split_slash: \l_stex_get_symbol_name_str
6022         /\_stex_args_end:
6023       }
6024     }
6025     \tl_clear:N \l_stex_current_term_tl
6026     \def\comp{\_comp}
6027     \let\compemph@uri#2
6028     #3{}{}\comp{#1}
6029   }{
6030     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
6031   }
6032   \group_end:
6033 }

```

(End of definition for \symname and others. These functions are documented on page 89.)

## Highlighting

```

6034 \comp
6035 \cs_new_protected:Nn \do_comp:nNn {
6036   \stex_pseudogroup_with:nn{\comp}{%
6037     \def\comp##1{##1}%
6038     \str_if_empty:NTF \l_stex_current_symbol_str {%
6039       #3%
6040     }{%
6041       \stex_if_html_backend:TF {%
6042         \stex_annotate:nn { shtml:#1 = \l_stex_current_symbol_str}{ #3 }%
6043       }{%
6044         \exp_args:Nno #2 { #3 } \l_stex_current_symbol_str%
6045       }%
6046     }%
6047   }%
6048 }
6049
6050 \cs_new_protected:Npn \comp {
6051   \do_comp:nNn {comp}\compemph@uri
6052 }
6053
6054 \cs_new_protected:Npn \varcomp {
6055   \do_comp:nNn {varcomp}\varempm@uri
6056 }
6057
6058 \cs_new_protected:Npn \defcomp {
6059   \do_comp:nNn {definiendum}\defemph@uri
6060 }
6061
6062 \cs_set_protected:Npn \comp {}
6063
6064 \cs_new_protected:Npn \compemph@uri #1 #2 {
6065   \compemph{ #1 }%
6066 }
6067
6068 \cs_new_protected:Npn \compemph #1 {
6069   #1%
6070 }
6071
6072 \cs_new_protected:Npn \defemph@uri #1 #2 {
6073   \defemph{#1}%
6074 }
6075
6076 \cs_new_protected:Npn \defemph #1 {
6077   \ifmmode\else\expandafter\textbf\fi{#1}%
6078 }
6079
6080 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
6081   \symrefemph{#1}%

```

```
6082 }
6083
6084 \cs_new_protected:Npn \symrefemph #1 {
6085     \emph{#1}
6086 }
6087
6088 \cs_new_protected:Npn \varemph@uri #1 #2 {
6089     \varemph{#1}
6090 }
6091
6092 \cs_new_protected:Npn \varemph #1 {
6093     #1
6094 }
```

(End of definition for `\comp` and others. These functions are documented on page 90.)

## 13.9 Mathematical Structures

6095 <@=stex\_structures>

\this

```
6096 \cs_new_protected:Npn \stex_current_this: {
6097   { \bool_set_true:N \l_stex_allow_semantic_bool
6098     \tl_if_empty:NTF \l_stex_current_this_tl {{}}{
6099       \str_set:Nx \l_stex_current_symbol_str {
6100         \l_stex_current_module_str ? \l__stex_structures_name_str
6101       }
6102       \maincomp{\l_stex_current_this_tl}
6103     }
6104   }
6105 }
6106 \let \this \stex_current_this:
```

(End of definition for \this. This function is documented on page 96.)

**mathstructure** (*env.*)

```
6107 \stex_new_stylable_env:nnnnnnnn {mathstructure}{m 0{}{}{}}{
6108   \__stex_structures_begin:nn{#1}{#2}
6109   \stex_smsmode_do:
6110 }{
6111   \stex_structural_feature_module_end:
6112   \__stex_structures_do_externals:
6113 }{}{}{}
6114
6115 \stex_keys_define:nnnn{mathstructure}{
6116   \tl_clear:N \l_stex_current_this_tl
6117   \str_clear:N \l__stex_structures_name_str
6118 }{
6119   this .tl_set:N = \l_stex_current_this_tl ,
6120   unknown .code:n = {
6121     \str_if_empty:NTF \l_keys_key_str {
6122       \str_set:Nx \l__stex_structures_name_str {\l_keys_key_tl}
6123     }{
6124       \str_set_eq:NN \l__stex_structures_name_str \l_keys_key_str
```

```

6125     }
6126   }
6127 }{}
6128
6129 \cs_new_protected:Nn \__stex_structures_begin:nn {
6130   \stex_keys_set:nn {mathstructure}{#2}
6131   \str_if_empty:NT \l__stex_structures_name_str {
6132     \str_set:Nn \l__stex_structures_name_str {#1}
6133   }
6134   \def\comp{\_comp}
6135
6136   \exp_args:Nne \use:nn { \stex_module_add_symbol:nnnnnnnnN }
6137   { {#1}{\l__stex_structures_name_str}{0}{}{defed}{
6138     \l_stex_current_module_str / \l__stex_structures_name_str-module
6139   }}
6140   {} \stex_invoke_structure:
6141   \str_set:Nx \l_stex_mroname_str {#1}
6142   \stex_execute_in_module:x{
6143     \seq_clear:c{\l_stex_structure_macros_\l_stex_current_module_str / \l__stex_structures_na
6144     \seq_put_right:cn{\l_stex_structure_macros_\l_stex_current_module_str / \l__stex_structur
6145   }
6146   \exp_args:No \stex_structural_feature_module:nn
6147   {\l__stex_structures_name_str}{structure}
6148 }
6149
6150 \stex_sms_allow_env:n{mathstructure}
6151 \stex_deactivate_macro:Nn \mathstructure {module-environments}
6152 \stex_every_module:n {\stex_reactivate_macro:N \mathstructure}
6153
6154 \cs_new_protected:Nn \__stex_structures_do_exernals: {
6155   \tl_set:Nn \l__stex_structures_replace_this_tl {####1}
6156   \exp_args:No \stex_iterate_symbols:nnf{\g_stex_last_feature_str} {
6157     \__stex_structures_external_decl:nnnn{##5}{##4}{##3}{##8}
6158   }
6159 }
6160
6161 \cs_new_protected:Nn \__stex_structures_external_decl:nnnn {
6162   \%stex_debug:nn{structure}%
6163   % Generating-external-declaration-\l__stex_structures_name_str/#3-in-
6164   % \l_stex_current_module_str^~J
6165   % \tl_to_str:n{#1}^~J\tl_to_str:n{#2}^~J\tl_to_str:n{#4}
6166   %
6167   \%tl_set:Nn \l_stex_get_symbol_args_tl {#1}
6168   \%exp_args:Nnx \use:nn { \stex_module_add_symbol:nnnnnnnnN} {
6169   % {}{\l__stex_structures_name_str/#3}{\int_eval:n{#2 + 1}}
6170   % {ii\tl_if_empty:nF{#1}{\stex_map_args:N \__stex_structures_shift_argls:nn}}
6171   % {defed}{typed}
6172   %}{#4}\stex_invoke_outer_field:
6173 }
6174
6175 \cs_new:Nn \__stex_structures_shift_argls:nn {
6176   \int_eval:n{#1+1}#2
6177 }

```

```

\stex_get_mathstructure:n
 6178 \cs_new_protected:Nn \stex_get_mathstructure:n {
 6179   \_stex_get_mathstructure:n{#1}
 6180   \str_if_empty:NT \l_stex_get_structure_module_str {
 6181     \msg_error:nnn{stex}{error/unknownstructure}{#1}
 6182   }
 6183 }
 6184 \cs_new_protected:Nn \_stex_get_mathstructure:n {
 6185   \str_clear:N \l_stex_get_structure_module_str
 6186   \_stex_get_symbol:n{#1}
 6187   \str_if_empty:NF \l_stex_get_symbol_name_str {
 6188     \exp_args:No \tl_if_eq:NNT \l_stex_get_symbol_invoke_cs \stex_invoke_structure: {
 6189       \str_set_eq:NN \l_stex_get_structure_module_str \l_stex_get_symbol_type_tl
 6190     }
 6191   }
 6192 }

```

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

`extstructure (env.)`

```

 6193 \stex_new_stylable_env:nnnnnnn {extstructure}{m 0{} m} {
 6194   \seq_clear:N \l__stex_structures_imports_seq
 6195   \clist_map_inline:nn{#3} {
 6196     \stex_get_mathstructure:n{##1}
 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     \stex_execute_in_module:x{
 6203       \seq_put_right:cn{\l_stex_structure_macros_\l_stex_get_structure_module_str _seq}{#1}
 6204     }
 6205   }
 6206   \__stex_structures_begin:nn{#1}{#2}
 6207   \seq_map_inline:Nn \l__stex_structures_imports_seq {
 6208     \stex_if_do_html:T {
 6209       \hbox{\stex_annotation_invisible:nn
 6210         {shtml:import={##1}} {}}
 6211     }
 6212     \stex_module_add_morphism:nonn
 6213     {}{##1}{import}{}
 6214     \stex_execute_in_module:x{
 6215       \stex_activate_module:n{##1}
 6216     }
 6217   }
 6218   \stex_smsmode_do:
 6219 }{
 6220   \stex_structural_feature_module_end:
 6221   \__stex_structures_do_externals:
 6222 }{}{}{}
 6223
 6224 \stex_sms_allow_env:n{extstructure}
 6225 \stex_deactivate_macro:Nn \extstructure {module-environments}
 6226 \stex_every_module:n {

```

```

6227   \stex_reactivate_macro:N \extstructure
6228 }
6229
6230 \cs_new:Nn \__stex_structures_extend_structure_i:NnnnnnnN {
6231   \exp_not:n{#1{#2}{#3}{#4}{#5}{defed}}{\l__stex_structures_extmod_str,\#7}\exp_not:n{{#8}{#9}}
6232 }
6233 \cs_new_protected:Nn \__stex_structures_extend_structure:nn {
6234   \stex_debug:nn{ext}{Extending~#1~by~#2}
6235   \str_set:Nn \l__stex_structures_extmod_str{#2}
6236   \tl_set:cx{#1}{
6237     \exp_after:wN \exp_after:wN \exp_after:wN
6238     \__stex_structures_extend_structure_i:NnnnnnnN \cs:w #1 \cs_end:
6239   }
6240 }
6241
6242 \stex_new_stylable_env:nnnnnnn {extstructure*}{m}){
6243   \__stex_structures_new_extstruct_name:
6244   \seq_clear:N \l__stex_structures_imports_seq
6245   \stex_get_mathstructure:n{#1}
6246
6247   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6248     \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6249       \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6250     }
6251   }
6252
6253 \stex_execute_in_module:x{
6254   \seq_map_inline:cn{\l_stex_structure_macros_\l_stex_get_structure_module_str _seq} {
6255     \exp_not:N \tl_if_exist:cT{####1} {
6256       \__stex_structures_extend_structure:nn{####1}{\l_stex_current_module_str/\l__stex_st
6257     }
6258   }
6259 }
6260
6261 \exp_args:No \__stex_structures_begin:nn\l__stex_structures_exstruct_name_str()
6262
6263 \seq_map_inline:Nn \l__stex_structures_imports_seq {
6264   \stex_if_do_html:T {
6265     \stex_annotation_invisible:nn
6266     {shtml:import= {##1}} {}
6267   }
6268   \stex_module_add_morphism:nonn
6269   {}{##1}{import}{}}
6270 \stex_execute_in_module:x{
6271   \stex_activate_module:n{##1}
6272 }
6273 }
6274
6275 \stex_smsmode_do:
6276 }{
6277 \prop_map_inline:cn{
6278   c_stex_module_ \l_stex_current_module_str _symbols_prop
6279 }{
6280   \__stex_structures_check_def:nnnnnnn ##2

```

```

6281   }
6282   \stex_structural_feature_module_end:
6283   \__stex_structures_do_externals:
6284 }{}{}{}

6285
6286 \stex_sms_allow_env:n{extstructure*}
6287 \exp_after:wN \stex_deactivate_macro:Nn
6288   \cs:w extstructure*\cs_end: {module-environments}
6289 \stex_every_module:n {
6290   \exp_after:wN \stex_reactivate_macro:N \cs:w extstructure*\cs_end:
6291 }
6292
6293 \cs_new_protected:Nn \__stex_structures_check_def:nnnnnnnn {
6294   \tl_if_empty:nT{#5} {
6295     \msg_error:nnnn{stex}{error/needsdefiniens}{#2}{extstructure*}
6296   }
6297 }
6298
6299 \stex_every_module:n{ \str_set:Nn \l__stex_structures_extname_count 0}
6300
6301 \cs_new_protected:Nn \__stex_structures_new_extstruct_name: {
6302   \stex_do_up_to_module:n {
6303     \str_set:Nx \l__stex_structures_extname_count {\int_eval:n{\l__stex_structures_extname_c
6304   }
6305   \str_set:Nx \l__stex_structures_exstruct_name_str {EXTSTRUCT_\l__stex_structures_extname_c
6306 }
6307

6308 Invoking structures:
6309 \cs_new_protected:Nn \stex_invoke_structure: {
6310   \tl_set:Nn \l__stex_structures_set_comp_tl {\__stex_structures_set_thiscomp:}
6311   \__stex_structures_invoke_top:n {}
6312
6313 \cs_new_protected:Nn \__stex_structures_invoke_top:n {
6314   \stex_debug:nn{structure} {
6315     invoking~structure~{\l_stex_current_type_tl}<\tl_to_str:n{#1}>
6316   }
6317   \peek_charcode:NTF [ {
6318     \__stex_structures_merge:nw{#1}
6319   }{
6320     \__stex_structures_invocation_type:n {#1}
6321     \tl_set:Nn \l__stex_structures_this_tl {}
6322     \peek_charcode_remove:NTF ! {
6323       \peek_charcode:NTF [ {
6324         \__stex_structures_maybe_notation:w
6325       }{
6326         \__stex_structures_maybe_notation:w []
6327       }
6328     }{
6329       \__stex_structures_invoke_this:n
6330     }
6331   }
6332 }

```

```

6333 \cs_new_protected:Npn __stex_structures_merge:nw #1 [ #2 ] {
6334   \exp_args:Ne \stex_if_starts_with:nNTF {\tl_to_str:n{#2}}{comp} {
6335     __stex_structures_set_customcomp: #2 __stex_structures_end:
6336     __stex_structures_invoke_top:n{#1}
6337   }{
6338     \exp_args:Ne \stex_if_starts_with:nNTF {\tl_to_str:n{#2}}{this} {
6339       __stex_structures_set_thisnotation: #2 __stex_structures_end:
6340       __stex_structures_invoke_top:n{#1}
6341     }{
6342       \tl_if_empty:nTF{#1} {
6343         __stex_structures_invoke_top:n{#2}
6344       }{
6345         \tl_if_empty:nTF{#2} {
6346           __stex_structures_invoke_top:n{#1}
6347         }{
6348           __stex_structures_invoke_top:n{#1, #2}
6349         }
6350       }
6351     }
6352   }
6353 }
6354 }
6355
6356 \cs_new_protected:Npn __stex_structures_set_thisnotation: this= #1 __stex_structures_end:
6357   \tl_set:Nn \l_stex_return_notation_tl { \comp{#1} }
6358   \tl_set:Nn \l_stex_structures_set_comp_tl {}
6359 }
6360
6361 \cs_new_protected:Npn __stex_structures_set_customcomp: comp= #1 __stex_structures_end: {
6362   \tl_set:Nn \l_stex_structures_set_comp_tl {
6363     __stex_structures_set_custom_comp:n{#1}
6364   }
6365   \tl_set:Nn \l_stex_return_notation_tl { \comp{} }
6366 }

```

The structure type:

```

6367 \cs_new_protected:Nn __stex_structures_invokation_type:n {
6368   __stex_structures_do_assign_list:n{#1}
6369   \clist_if_empty:NTF \l_stex_structures_fields_clist {
6370     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6371       = 1 {
6372         \tl_set:Nx \l_stex_structures_current_type_tl {
6373           \exp_args:No \exp_not:n \l_stex_current_redo_tl
6374           \stex_term_oms_or_omv:nnn{}{}{}
6375         }
6376       }{
6377         \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6378       }
6379     }{
6380       \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6381         = 1 {
6382           \__stex_structures_make_type:n {}
6383         }{
6384           \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6385         }

```

```

6386     }
6387 }
6388
6389 \cs_new_protected:Nn \__stex_structures_do_assign_list:n {
6390   \clist_clear:N \l__stex_structures_fields_clist
6391   \tl_if_empty:nF {#1} {
6392     \keyval_parse:NNn\TODO\__stex_structures_do_assign:nn{#1}
6393   }
6394 }
6395
6396 \cs_new_protected:Nn \__stex_structures_do_assign:nn {
6397   \clist_put_right:Nn \l__stex_structures_fields_clist {{#1}{#2}}
6398 }
6399
6400 \cs_new_protected:Nn \__stex_structures_make_type:n {
6401   \tl_if_empty:nTF{#1} {
6402     \seq_clear:N \l_tmpa_seq
6403   }{
6404     \seq_set_split:Nnn \l_tmpa_seq ,{#1}
6405     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
6406     \seq_reverse:N \l_tmpa_seq
6407   }
6408 \tl_set:Nx \l__stex_structures_current_type_tl {
6409   \symuse{Metatheory?module~type~merge}{%
6410     {
6411       \exp_args:No \exp_not:n \l_stex_current_redo_tl
6412       \stex_term_oms_or_omv:nnn{}{}{%
6413     }
6414     \seq_map_function:NN \l_tmpa_seq \__stex_structures_make_mod:n
6415     \clist_if_empty:NF \l__stex_structures_fields_clist {
6416       ,\symuse{Metatheory?anonymous~record}{%
6417         \exp_args:Ne \tl_tail:n{%
6418           \clist_map_function:NN \l__stex_structures_fields_clist \__stex_structures_make_
6419         }
6420       }
6421     }
6422   }
6423 }
6424 }
6425
6426 \cs_new:Nn \__stex_structures_make_mod:n {
6427   ,\symuse{Metatheory?module~type}{%
6428     \stex_annotation:nn{shtml:term=OMMOD,shtml:head={#1}}{}%
6429   }
6430 }
6431
6432 \cs_new:Nn \__stex_structures_make_oml:n {
6433   \__stex_structures_make_oml:nn #1
6434 }
6435 \cs_new:Nn \__stex_structures_make_oml:nn {
6436   ,\stex_annotation:nn{
6437     shtml:term=OML,
6438     shtml:head={#1}
6439 }{

```

```

6440     \_stex_annotation_force_break:n{
6441         \stex_annotation:nn{shtml:definiens={}}{\exp_not:n{#2!}}
6442     }
6443 }
6444 }
```

Insert the structure type as a term:

```

6445 \cs_new:Nn \__stex_structures_current_type: {
6446     \%exp_args:No \exp_not:n \l_stex_current_redo_tl
6447     \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6448         \exp_args:No\exp_not:n\l__stex_structures_current_type_tl
6449     }
6450     \_stex_term_oms_or_omv:nnn{}{}{}
6451 }
```

The structure type itself:

```

6452 \cs_new_protected:Npn \__stex_structures_maybe_notation:w [ #1 ] {
6453     \tl_set_eq:NN \l_stex_current_term_tl \l_stex_structures_current_type_tl
6454     \cs_if_exist:cTF{\l_stex_notation_\l_stex_current_symbol_str _#1_cs} {
6455         \use:c{\l_stex_notation_\l_stex_current_symbol_str _#1_cs}\group_end:
6456     }{
6457         \__stex_structures_make_prop:
6458         \__stex_structures_make_prop_assign:
6459         \__stex_structures_present_i:w [#1]
6460     }
6461 }
6462
6463 \cs_new_protected:Nn \__stex_structures_present: {
6464     \peek_charcode:NTF [ {
6465         \__stex_structures_present_i:w
6466     }{
6467         \__stex_structures_present:nn{}{}
6468     }
6469 }
6470
6471 \cs_new_protected:Npn \__stex_structures_present_i:w [#1] {
6472     \int_compare:nNnTF{\clist_count:n{#1}} = 1 {
6473         \__stex_structures_present:nn{}{#1}
6474     }{
6475         \peek_charcode:NTF [ {
6476             \__stex_structures_present_ii:nw{#1}
6477         }{
6478             \__stex_structures_present:nn{#1}{}
6479         }
6480     }
6481 }
6482
6483 %First: clist, second:notation-id
6484 \cs_new_protected:Npn \__stex_structures_present_ii:nw #1 [#2] {
6485     \__stex_structures_present:nn{#1}{#2}
6486 }
6487
6488 \cs_new_protected:Nn \__stex_structures_present:nn {
6489     \clist_clear:N \l__stex_structures_clist
6490     \tl_if_empty:nTF{#1}{
```

```

6491   \cs_set:Npn \l__stex_structures_cs ##1 ##2 ##3 {
6492     \tl_if_empty:nF{##2}{
6493       \l__stex_structures_present_entry:nn {##1}{##3}
6494     }
6495   }
6496 }{
6497   \cs_set:Npn \l__stex_structures_cs ##1 ##2 ##3 {
6498     \exp_args:Ne \clist_if_in:nnT{\tl_to_str:n{#1}}{##1} {
6499       \l__stex_structures_present_entry:nn {##1}{##3}
6500     }
6501   }
6502 }
6503 \prop_map_inline:Nn \l__stex_structures_prop {
6504   \l__stex_structures_cs {##1} ##2
6505 }
6506 \l__stex_term_oms_or_omv:nnn{}{}{
6507   \exp_args:Nno \use:n{
6508     \bool_set_true:N \l_stex_allow_semantic_bool
6509     \symuse{Metatheory?mathematical-structure}[#2]
6510   }{\l__stex_structures_clist}
6511 }\group_end:
6512 }
6513
6514 \cs_new_protected:Nn \l__stex_structures_present_entry:nn {
6515   \seq_if_in:NnTF \l__stex_structures_assigned_seq {#1} {
6516     \clist_put_right:Nn \l__stex_structures_clist {#2!}
6517   }
6518   \exp_args:NNe \clist_put_right:Nn \l__stex_structures_clist {
6519     \l__stex_next_symbol:n {
6520       \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
6521       \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
6522         \exp_args:No \exp_not:n \l__stex_structures_this_tl
6523       }
6524       \exp_not:n {
6525         \tl_set_eq:NN \this \l__stex_structures_this_tl
6526       }
6527       \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6528         \exp_args:No \exp_not:n \l_stex_return_notation_tl
6529       }
6530     }
6531     \exp_not:n{#2!}
6532   }
6533 }
6534 }
6535
6536 \cs_new_protected:Npn \_thiscomp #1 #2 {
6537   {\tl_set:cn{this}{{}}#1{#2}\c_math_subscript_token{
6538     \group_begin:
6539     \bool_set_true:N \l_stex_allow_semantic_bool
6540     \l__stex_structures_this_tl
6541     \group_end:
6542   }
6543 }
6544 }

```

```

6545 \cs_new_protected:Nn \__stex_structures_set_thiscomp: {
6546   \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_thiscomp {
6547     \edef\maincomp {\_thiscomp{\comp}}
6548   }
6549 }
6550 }
6551
6552 \cs_new_protected:Nn \__stex_structures_set_custom_comp:n {
6553   \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
6554     \cs_set_protected:Npx \_customthiscomp ##1 {
6555       \group_begin:
6556         \bool_set_true:N \l_stex_allow_semantic_bool
6557         \exp_not:n{
6558           \cs_set:Npn \l__stex_structures_comp_cs ##1 {
6559             #1
6560           }
6561           \def\maincomp
6562             }{\comp}
6563           \exp_not:N \l__stex_structures_comp_cs{\comp{##1}}
6564         \group_end:
6565       }
6566       \def\maincomp {\_customthiscomp}
6567     }
6568   }
6569 }

this (of type structure):
6570
6571 \cs_new_protected:Nn \__stex_structures_invoke_this:n {
6572   \peek_charcode_remove:NTF ! {
6573     \exp_args:Nne\use:nn{
6574       \group_end:\symuse{Metatheory?of-type}[invisible]{#1}
6575     }{
6576       {\__stex_structures_current_type:}
6577     }
6578   }{
6579     \__stex_structures_invoke_maybe_field:nn{#1}
6580   }
6581 }

6582 \cs_new_protected:Nn \__stex_structures_invoke_maybe_field:nn {
6583   \__stex_structures_make_prop:
6584   \__stex_structures_set_this:n{#1}
6585   \tl_if_empty:nTF{#2}{
6586     \__stex_structures_make_prop_assign:
6587     \__stex_structures_present:
6588   }{
6589     \__stex_structures_invoke_field:n{#2}
6590   }
6591 }
6592 }

6593 \cs_new_protected:Nn \__stex_structures_set_this:n {
6594   \tl_if_empty:nTF{#1}{
6595     \%tl_put_right:Nn \l_stex_current_redo_tl {
6596       \%tl_clear:N \l__stex_structures_this_tl

```

```

6598      %}
6599  }{
6600    \tl_set:Nx \l__stex_structures_this_tl {{
6601      \bool_set_true:N \l_stex_allow_semantic_bool
6602      \tl_set:Nn \exp_not:N \this {
6603        \exp_args:No \exp_not:n \this
6604      }
6605      \exp_not:n{#1}
6606    }}
6607    \tl_set_eq:NN \this \l__stex_structures_this_tl
6608    % \l_stex_return_notation_tl
6609  }
6610 }
6611
6612 \cs_new_protected:Nn \__stex_structures_get_field_name:n {
6613   \str_set:Nx \l__stex_structures_field_name_str {
6614     \exp_args:Nne \use:n {\exp_after:wN \use_i:nn \use:n}
6615     {\prop_item:Nn \l__stex_structures_prop {#1}}
6616   }
6617   \str_if_empty:NT \l__stex_structures_field_name_str {
6618     \str_set:Nn \l__stex_structures_field_name_str {#1}
6619   }
6620 }
6621
6622 \cs_new_protected:Nn \__stex_structures_invoke_field:n {
6623   \prop_if_in:NnTF \l__stex_structures_prop {#1} {
6624     \__stex_structures_get_field_name:n{#1}
6625     \tl_clear:N \l__stex_structures_more_nextsymbol_tl
6626     \%exp_args:NNe \seq_if_in:NnF \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}{
6627       \tl_set:Nx \l__stex_structures_more_nextsymbol_tl {
6628         \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
6629           \exp_args:No \exp_not:n \l__stex_structures_this_tl
6630         }
6631         \exp_not:n {
6632           \tl_set_eq:NN \this \l__stex_structures_this_tl
6633         }
6634         \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6635           \exp_args:No \exp_not:n \l_stex_return_notation_tl
6636         }
6637         \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
6638       }
6639     %}
6640   \exp_args:NNx \use:nn \group_end: {
6641     \stex_next_symbol:n {
6642       \exp_args:No \exp_not:n \l__stex_structures_redo_tl
6643       \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6644         \symuse{Metatheory?record~field}{%
6645           \symuse{Metatheory?of~type}{%
6646             \exp_args:No \exp_not:n \l__stex_structures_this_tl
6647           }{ \__stex_structures_current_type: }
6648         }{
6649           \stex_annotation:nn{shtml:term=OML,shtml:head={\l__stex_structures_field_name_str}}
6650         }
6651     }

```

```

6652     \exp_args:No \exp_not:n \l__stex_structures_more_nextsymbol_tl
6653   }
6654   \exp_not:N \use_ii:nn
6655   \prop_item:Nn \l__stex_structures_prop {#1}
6656 }
6657 }{
6658   \msg_error:nnn{stex}{error/unknownfield}{#1}
6659 }
6660 }
6661
6662 \cs_new_protected:Nn \__stex_structures_make_prop: {
6663   \prop_clear:N \l__stex_structures_prop
6664   \seq_clear:N \l__stex_structures_seq
6665   \seq_clear:N \l__stex_structures_assigned_seq
6666   \tl_clear:N \l__stex_structures_redo_tl
6667   \__stex_structures_prop_do_decls:
6668   \__stex_structures_prop_do_notations:
6669 }
6670
6671 \cs_new_protected:Nn \__stex_structures_make_prop_assign: {
6672   \clist_if_empty:NF \l__stex_structures_fields_clist {
6673     \clist_map_inline:Nn \l__stex_structures_fields_clist {
6674       \__stex_structures_make_prop_assign:nn ##1
6675     }
6676   }
6677 }
6678
6679 \cs_new_protected:Nn \__stex_structures_make_prop_assign:nn {
6680   \prop_if_in:NnTF \l__stex_structures_prop {#1} {
6681     \exp_args:NNN \seq_put_right:Nn \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}
6682     \exp_args:Nne \use:nn {\__stex_structures_make_prop_assign_replace:nnnn {#1}{#2}}
6683     {\prop_item:Nn \l__stex_structures_prop {#1}}
6684   }{
6685     \msg_error:nnn{stex}{error/unknownfieldass}{#1}
6686   }
6687 }
6688 \cs_new_protected:Nn \__stex_structures_make_prop_assign_replace:nnnn {
6689   \prop_put:Nnn \l__stex_structures_prop {#1}{##3}{#2}
6690   \tl_if_empty:nF{##3} {
6691     \tl_set:cn{#1}{#2}
6692     \tl_put_right:Nn \l__stex_structures_redo_tl {
6693       \tl_set:cn{#1}{#2}
6694     }
6695   }
6696 }
6697
6698 \cs_new_protected:Nn \__stex_structures_prop_do_decls: {
6699   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_type_tl {
6700     \tl_if_empty:nTF{##2} {
6701       \__stex_structures_do_decl_nomacro:nnnnnnnn{##3}
6702     }{
6703       \__stex_structures_do_decl:nnnnnnnn{##2}
6704     }
6705     {##1}{##3}{##4}{##5}{##6}{##7}{##8}{##9}

```

```

6706   }
6707 }
6708
6709 \cs_new_protected:Nn \__stex_structures_do_decl_nomacro:nnnnnnnnn {
6710   \prop_if_in:NnF \l__stex_structures_prop {#1} {
6711     \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
6712     \prop_put:Nnn \l__stex_structures_prop {#1} {
6713       {}{
6714         \stex_invoke_symbol:nnnnnnnnN
6715         {#2}
6716         {#3}
6717         {#4}{#5}{#6}{#7}{#8}#9
6718       }
6719     }
6720   }
6721 }
6722
6723 \cs_new_protected:Nn \__stex_structures_do_decl:nnnnnnnn {
6724   \prop_if_in:NnF \l__stex_structures_prop {#1} {
6725     \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
6726     \prop_put:Nnn \l__stex_structures_prop {#1} {
6727       {}{
6728         \stex_invoke_symbol:nnnnnnnnN
6729         {#2}
6730         {#3}
6731         {#4}{#5}{#6}{#7}{#8}#9
6732       }
6733     }
6734   }
6735 \% \tl_set:cn{#1} {
6736   \% \stex_invoke_symbol:nnnnnnnnN
6737   \% {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
6738   \%}
6739 \% \tl_put_right:Nn \l__stex_structures_redo_tl {
6740 \% \tl_set:cn{#1} {
6741 \% \stex_invoke_symbol:nnnnnnnnN
6742 \% {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
6743 \%}
6744 \%}
6745 }
6746
6747 \cs_new_protected:Nn \__stex_structures_prop_do_notations: {
6748   \exp_args:No \stex_iterate_notations:nn\l_stex_current_type_tl{
6749     \exp_args:NNe \seq_if_in:NnT \l__stex_structures_seq {\tl_to_str:n{##1}}{
6750       \tl_put_right:Nn \l__stex_structures_redo_tl {
6751         \cs_if_exist:cF{\l_stex_notation_##1 _##2_cs} {
6752           \tl_set:cn{\l_stex_notation_##1 _##2_cs}{##4}
6753         }
6754         \cs_if_exist:cF{\l_stex_notation_##1 __cs} {
6755           \tl_set:cn{\l_stex_notation_##1 __cs}{##4}
6756         }
6757       }
6758       \cs_if_exist:cF{\l_stex_notation_##1 _##2_cs} {
6759         \tl_set:cn{\l_stex_notation_##1 _##2_cs}{##4}

```

```

6760 }
6761 \cs_if_exist:cF{l_stex_notation_##1 __cs}{
6762     \tl_set:cn{l_stex_notation_##1 __cs}{##4}
6763 }
6764 \tl_if_empty:nF{##5}{
6765     \tl_put_right:Nn \l__stex_structures_redo_tl {
6766         \cs_if_exist:cF{l_stex_notation_##1 _op_##2_cs}{
6767             \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
6768         }
6769         \cs_if_exist:cF{l_stex_notation_##1 _op__cs}{
6770             \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
6771         }
6772     }
6773     \cs_if_exist:cF{l_stex_notation_##1 _op_##2_cs}{
6774         \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
6775     }
6776     \cs_if_exist:cF{l_stex_notation_##1 _op__cs}{
6777         \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
6778     }
6779 }
6780 }
6781 }
6782 }

```

### \usestructure

```

6783 \cs_new_protected:Npn \usestructure #1 {
6784     \stex_get_mathstructure:n{ #1 }
6785     \seq_clear:N \l__stex_structures_imports_seq
6786     \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6787         \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6788             \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6789         }
6790     }
6791     \seq_map_inline:Nn \l__stex_structures_imports_seq {
6792         \stex_if_do_html:T {
6793             \hbox{\stex_annotation_invisible:nn
6794                 {shtml:usemodule=##1} {}}
6795         }
6796         \stex_activate_module:n {##1}
6797     }
6798 }

```

(End of definition for \usestructure. This function is documented on page 96.)

## 13.10 Statements

```

6799 <@=stex_statements>
6800
6801 \stex_keys_define:nnnn{statement}{
6802     \str_clear:N \l_stex_key_name_str
6803     \str_clear:N \l_stex_key_macroname_str
6804     \clist_clear:N \l_stex_key_for_clist
6805     \str_clear:N \l_stex_key_args_str

```

```

6806   \tl_clear:N \l_stex_key_type_tl
6807   \tl_clear:N \l_stex_key_def_tl
6808   \tl_clear:N \l_stex_key_return_tl
6809   \clist_clear:N \l_stex_key_argtypes_clist
6810 }{
6811   name      .str_set:N = \l_stex_key_name_str ,
6812   for       .clist_set:N = \l_stex_key_for_clist ,
6813   macro     .str_set:N = \l_stex_key_mroname_str ,
6814   % start   .str_set:N = \l_stex_key_title_str , % TODO remove
6815   type     .tl_set:N = \l_stex_key_type_tl ,
6816   judgment  .code:n = {},
6817   from     .code:n= {}, % TODO remove
6818   to      .code:n={} % TODO remove
6819 }{id,title,style,symargs}

\stex_new_statement:nn
6820 \cs_new_protected:Npn \stex_do_for_list: {
6821   \seq_clear:N \l_stex_fors_seq
6822   \clist_map_inline:Nn \l_stex_key_for_clist {
6823     \exp_args:N\stex_get_symbol:n{\tl_to_str:n{##1}}
6824     \seq_put_right:Nx \l_stex_fors_seq
6825       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6826   }
6827 }
6828
6829 \cs_new_protected:Nn \__stex_statements_setup:nn {
6830   \str_if_empty:NF \l_stex_key_mroname_str {
6831     \str_if_empty:NT \l_stex_key_name_str {
6832       \str_set_eq:NN \l_stex_key_name_str \l_stex_key_mroname_str
6833     }
6834   }
6835   \stex_do_for_list:
6836   \str_if_empty:NF \l_stex_key_name_str {
6837     \__stex_statements_force_id:
6838     \seq_put_right:Nx \l_stex_fors_seq {
6839       \l_stex_current_module_str ? \l_stex_key_name_str
6840     }
6841     \str_set_eq:NN \l_stex_mroname_str \l_stex_key_mroname_str
6842     \str_set:Nn \l_stex_key_role_str {#2}
6843     \stex_symdecl_do:
6844     \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN} {
6845       {\l_stex_key_mroname_str}{\l_stex_key_name_str}
6846       {\int_use:N \l_stex_get_symbol_arity_int}
6847       {\l_stex_get_symbol_args_tl}
6848       {#1}{}{}\stex_invoke_symbol:
6849     }
6850     \stex_if_do_html:T \stex_symdecl_html:
6851   }
6852   \str_clear:N \l__stex_statements_uri_str
6853   \str_if_empty:NTF \l_stex_key_name_str {
6854     \stex_debug:nn{statement}{no~name}
6855     \int_compare:nNnTF {\seq_count:N \l_stex_fors_seq} = 1 {
6856       \str_set:Nx \l__stex_statements_uri_str {\seq_item:Nn \l_stex_fors_seq 1}
6857       \stex_debug:nn{statement}{for:~\l__stex_statements_uri_str}

```

```

6858     }{
6859         \stex_debug:nn{statement}{no~for}
6860     }
6861 }{
6862     \str_set:Nx \l__stex_statements_uri_str {\l_stex_current_module_str ? \l_stex_key_name_s
6863     \stex_debug:nn{statement}{name:~\l__stex_statements_uri_str}
6864 }
6865 }
6866
6867 \cs_new:Nn \__stex_statements_html_keyvals:nn {
6868     shtml:#1={},
6869     shtml:inline={#2},
6870     \seq_if_empty:NF \l_stex_fors_seq {
6871         shtml:fors=\seq_use:Nn \l_stex_fors_seq ,}
6872     }
6873     \str_if_empty:NF \l_stex_key_id_str {
6874         shtml:id=\stex_uri_use:N \l_stex_current_doc_uri ? \l_stex_key_id_str}
6875     }
6876     \clist_if_empty:NF \l_stex_key_style_clist {
6877         shtml:styles=\l_stex_key_style_clist}
6878     }
6879 }
6880
6881 \cs_new_protected:Nn \stex_new_statement:nnn {
6882     \stex_new_stylable_env:nnnnnnn {#1}{0{} }{
6883         \stex_keys_set:nn{statement}{##1}
6884         #3
6885
6886         \stex_if_smsmode:F {
6887             \exp_args:Nne \begin{stex_annotation_env} {
6888                 \__stex_statements_html_keyvals:nn{#1}{false}
6889             }
6890             \tl_set_eq:NN \thistitle \l_stex_key_title_tl
6891             \str_set_eq:NN \thisname \l_stex_key_name_str
6892             \clist_set_eq:NN \thisfor \l_stex_key_for_str
6893             \stex_if_html_backend:TF {
6894                 \noindent
6895                 \stex_annotation:nn{shtml:statementtitle={} }{\stex_annotation_force_break:n\l_stex_key_
6896             }
6897             \stex_style_apply:
6898         }
6899         \stex_do_id:
6900         \stex_smsmode_do:
6901     }{
6902         \stex_if_smsmode:F {
6903             \stex_if_html_backend:F \stex_style_apply:
6904             \end{stex_annotation_env}
6905         }
6906     }{}{}{s}
6907     \stex_sms_allow_env:n{s#1}
6908
6909     \tl_if_empty:nF{#2} {
6910         \exp_after:wN \NewDocumentCommand \cs:w inline#2\cs_end: { 0{} m}{%
6911             \group_begin:

```

```

6912      \stex_keys_set:nn{statement}{##1}
6913      #3
6914      \_stex_do_id:
6915      \stex_if_smsmode:F{
6916          \exp_args:N \stex_annotation:nn{\_stex_statements_html_keyvals:nn{#1}{true}}{%
6917              \stex_annotation_force_break:n{##2}
6918          }
6919      }
6920      \group_end:
6921      \stex_smsmode_do:
6922  }
6923  \exp_after:wN \stex_sms_allow_escape:N\cs:w inline#2\cs_end:
6924 }
6925 }

6926 \cs_new_protected:Nn \_stex_statements_setup_def: {
6927     \stex_if_smsmode:F{
6928         \seq_map_inline:Nn \l_stex_fors_seq {
6929             \stex_ref_new_sym_target:n{##1}
6930         }
6931     }
6932     \stex_reactivate_macro:N \definiendum
6933     \stex_reactivate_macro:N \defnotation
6934     \stex_reactivate_macro:N \defname
6935     \stex_reactivate_macro:N \Defname
6936     \stex_reactivate_macro:N \varbind
6937 }
6938 }

6939 \cs_new_protected:Nn \_stex_statements_force_id: {
6940     \str_if_empty:NT \l_stex_key_id_str {
6941         \_stex_ref_new_id:n{}
6942         \str_set_eq:NN \l_stex_key_id_str \l_stex_refs_str
6943     }
6944 }
6945 }

6946 \stex_new_statement:nnn{definition}{def} {
6947     \_stex_statements_force_id:
6948     \_stex_statements_setup:nn{}{}
6949     \_stex_statements_setup_def:
6950     \stex_reactivate_macro:N \definiens
6951 }
6952 }

6953 \stex_new_statement:nnn{assertion}{ass} {
6954     \_stex_statements_setup:nnf{}{assertion}
6955     \stex_if_smsmode:F{
6956         \seq_map_inline:Nn \l_stex_fors_seq {
6957             \stex_ref_new_sym_target:n{##1}
6958         }
6959     }
6960     \stex_reactivate_macro:N \varbind
6961     \stex_reactivate_macro:N \conclusion
6962     \stex_reactivate_macro:N \premise
6963     \stex_reactivate_macro:N \definiendum
6964     \stex_reactivate_macro:N \defnotation
6965     \stex_reactivate_macro:N \defname

```

```

6966   \stex_reactivate_macro:N \Definename
6967 }
6968 \stex_new_statement:nnn@example}{ex}{{\_\_stex_statements_setup:nn{}{example}}
6969 \stex_new_statement:nnn{paragraph}{}{
6970   \clist_if_in:NnTF \l_stex_key_style_clist {symdoc} {
6971     \_\_stex_statements_force_id:
6972     \_\_stex_statements_setup:nn{}{}
6973     \_\_stex_statements_setup_def:
6974   }{
6975     \_\_stex_statements_setup:nn{}{}
6976   }
6977 }

```

(End of definition for \stex\_new\_statement:nn. This function is documented on page ??.)

#### definiendum

```

6978 \cs_new_protected:Nn \_\_stex_statements_do_deref:nn {
6979   \stex_if_html_backend:T{\ifvmode\indent\fi}
6980   \group_begin:
6981   \stex_get_symbol:n{#1}
6982   \bool_if:NTF \l_stex_allow_semantic_bool{
6983     \str_set:Nx \l_stex_current_symbol_str
6984       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6985     \str_if_in:NnT \l_stex_get_symbol_name_str / {
6986       \str_set:Nx \l_stex_get_symbol_name_str {
6987         \exp_after:wN \l_stex_split_slash: \l_stex_get_symbol_name_str
6988         /\l_stex_args_end:
6989       }
6990     }
6991     \exp_args:No \stex_ref_new_sym_target:n \l_stex_current_symbol_str
6992     \def\comp{\_defcomp}
6993     \stex_annotation:nn{shtml:definiendum=\l_stex_current_symbol_str}{\comp{#2}}
6994   }{
6995     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
6996   }
6997   \group_end:
6998 }
6999
7000 \NewDocumentCommand \defnotation{ m } {
7001   \l_stex_next_symbol:n { \def\comp{\_defcomp}}#1
7002 }
7003 \stex_deactivate_macro:Nn \defnotation {definition~environments}
7004
7005 \NewDocumentCommand \definiendum { O{} m m } {
7006   \stex_keys_set:nn{symname}{ #1 }
7007   \_\_stex_statements_do_deref:nn{#2}{#3}
7008 }
7009 \stex_deactivate_macro:Nn \definiendum {definition~environments}
7010
7011 \NewDocumentCommand \definename { O{} m } {
7012   \stex_keys_set:nn{symname}{#1}
7013   \_\_stex_statements_do_deref:nn{#2}(
7014     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
7015   )

```

```

7016 }
7017 \stex_deactivate_macro:Nn \definame {definition-environments}
7018
7019 \NewDocumentCommand \Definame { O{} m } {
7020   \stex_keys_set:nn{symname}{#1}
7021   \__stex_statements_do_deref:nn{#2}{
7022     \l_stex_key_pre_tl\exp_after:wN\stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
7023   }
7024 }
7025 \stex_deactivate_macro:Nn \Definame {definition-environments}
7026
7027
7028 \NewDocumentCommand \definiens { O{} m }{
7029   \group_begin:
7030   \str_clear:N \l_stex_get_symbol_name_str
7031   \tl_if_empty:nF {#1} {
7032     \stex_get_symbol:n { #1 }
7033     \str_set:Nx \l__stex_statements_uri_str
7034       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7035   }
7036   \str_if_empty:NT \l__stex_statements_uri_str {
7037     \msg_error:nn{stex}{error/definiensfor}
7038   }
7039   \stex_debug:nn{definiens}{Checking~\l__stex_statements_uri_str}
7040
7041   \exp_args:No \_stex_add_definiens:nn \l__stex_statements_uri_str{#2}
7042
7043   \group_end:
7044   \stex_smsmode_do:
7045 }
7046 \stex_deactivate_macro:Nn \definiens {definition-environments}
7047 \stex_sms_allow_escape:N \definiens
7048
7049 \cs_new_protected:Nn \stex_add_definiens:nn {
7050   \exp_args:Nno \stex_str_if_starts_with:nnT{#1} \l_stex_current_module_str {
7051     \prop_map_inline:cN{c_stex_module_\l_stex_current_module_str _symbols_prop} {
7052       \stex_debug:nn{definiens}{#1 == \l_stex_current_module_str?##1}
7053       \str_if_eq:noT {#1} {\l_stex_current_module_str?##1} {
7054         \prop_map_break:n{\_stex_add_definiens_inner:nnnnnnnn ##2}
7055       }
7056     }
7057   }
7058   \stex_if_smsmode:F{
7059     \stex_annotation:nn{ shtml:definiens={#1}}{
7060       #2 \%_stex_annotation_force_break:n{ #2 }
7061     }
7062   }
7063 }
7064
7065 \cs_new_protected:Nn \stex_add_definiens_inner:nnnnnnnn {
7066   \stex_debug:nn{definiens}{Adding~definiens~to~\l_stex_current_module_str?#2}
7067   \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
7068     {#2}{##1}{#2}{#3}{#4}{defed}{#6}{#7}{#8}}
7069 }

```

```

7070 \NewDocumentCommand \varbind {m} {
7071   \clist_map_inline:nn {#1} {
7072     \stex_get_var:n {##1}
7073     \stex_if_do_html:T {
7074       \stex_annotation_invisible:nn {shtml:bind=\l_stex_get_symbol_name_str}{}
7075     }
7076   }
7077 }
7078 }
7079 \stex_deactivate_macro:Nn \varbind {definition-or-assertion-environments}
7080
7081 \NewDocumentCommand \conclusion { O{} m} {
7082   \group_begin:
7083   \str_clear:N \l_stex_get_symbol_name_str
7084   \tl_if_empty:nF {#1} {
7085     \stex_get_symbol:n { #1 }
7086     \str_set:Nx \l__stex_statements_uri_str
7087       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7088   }
7089   \str_if_empty:NT \l__stex_statements_uri_str {
7090     \msg_error:nn{stex}{error/conclusionfor}
7091   }
7092   \stex_annotation:nn{ shtml:conclusion=\l__stex_statements_uri_str}{
7093     #2 \%_stex_annotation_force_break:n{ #2 }
7094   }
7095   \group_end:
7096 }
7097 \stex_deactivate_macro:Nn \conclusion {assertion-environments}
7098
7099 \NewDocumentCommand \premise {O{} m} {
7100   \tl_if_empty:nF {#1} {
7101     \stex_debug:nn{Here:}{Variable~#1}
7102     \exp_args:Nne\use:nn{\vardef}{\v#1}[name=#1]{#1}
7103   }
7104   \stex_annotation:nn{shtml:premise={#1}}{#2}
7105 }
7106 \stex_deactivate_macro:Nn \premise {assertion-environments}

```

(End of definition for `definiendum`. This function is documented on page 100.)

## 13.11 Proofs

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

```

7107 <@@=stex_proof>
7108 \stex_keys_define:nnnn{ spf }{
7109   \tl_clear:N \l_stex_key_for_clist
7110   \tl_clear:N \l_stex_key_from_tl
7111   \tl_set_eq:NN \l_stex_key_proofend_tl \__stex_proof_proof_box_tl
7112   \tl_clear:N \l_stex_key_continues_tl
7113   \tl_clear:N \l_stex_key_term_tl
7114   \tl_clear:N \l_stex_key_functions_tl
7115   \tl_clear:N \l_stex_key_method_tl
7116   \bool_set_false:N \l_stex_key_hide_bool

```

```

7117 }{
7118   for      .clist_set:N = \l_stex_key_for_clist ,
7119   from     .tl_set:N   = \l_stex_key_from_tl ,
7120   proofend .tl_set:N   = \l_stex_key_proofend_tl,
7121   continues .tl_set:N   = \l_stex_key_continues_tl,
7122   functions .tl_set:N   = \l_stex_key_functions_tl,
7123   term     .tl_set:N   = \l_stex_key_term_tl,
7124   method    .tl_set:N   = \l_stex_key_method_tl,
7125   hide      .bool_set:N = \l_stex_key_hide_bool
7126 }{id,style,title}
7127
7128 \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 accommodate semantic information.

```

7129 \intarray_new:Nn\l__stex_proof_counter_intarray{50}
7130 \cs_new_protected:Npn \__stex_proof_insert_number: {
7131   \int_set:Nn \l_tmpa_int {1}
7132   \bool_while_do:nn {
7133     \int_compare_p:nNn {
7134       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7135     } > 0
7136   }{
7137     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .
7138     \int_incr:N \l_tmpa_int
7139   }
7140 }
7141 \cs_new_protected:Nn \__stex_proof_number_as_string:N {
7142   \str_clear:N #1
7143   \int_set:Nn \l_tmpa_int {1}
7144   \bool_while_do:nn {
7145     \int_compare_p:nNn {
7146       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7147     } > 0
7148   }{
7149     \str_put_right:Nx #1 {\intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .}
7150     \int_incr:N \l_tmpa_int
7151   }
7152 }
7153
7154 \cs_new_protected:Npn \__stex_proof_inc_counter: {
7155   \int_set:Nn \l_tmpa_int {1}
7156   \bool_while_do:nn {
7157     \int_compare_p:nNn {
7158       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7159     } > 0
7160   }{
7161     \int_incr:N \l_tmpa_int

```

```

7162 }
7163 \int_compare:nNnF \l_tmpa_int = 1 {
7164   \int_decr:N \l_tmpa_int
7165 }
7166 \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int {
7167   \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int + 1
7168 }
7169 }
7170
7171 \cs_new_protected:Npn \__stex_proof_add_counter: {
7172   \int_set:Nn \l_tmpa_int {1}
7173   \bool_while_do:nn {
7174     \int_compare_p:nNn {
7175       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7176     } > 0
7177   }{
7178     \int_incr:N \l_tmpa_int
7179   }
7180   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 1 }
7181 }
7182
7183 \cs_new_protected:Npn \__stex_proof_remove_counter: {
7184   \int_set:Nn \l_tmpa_int {1}
7185   \bool_while_do:nn {
7186     \int_compare_p:nNn {
7187       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7188     } > 0
7189   }{
7190     \int_incr:N \l_tmpa_int
7191   }
7192   \int_decr:N \l_tmpa_int
7193   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 0 }
7194 }

```

### spfesketch

```

7195 \newenvironment{spfesketchenv}{}{}
7196 \stex_new_stylable_cmd:nnnn{spfesketch}{0{} m}{\par
7197   \begin{spfesketchenv}
7198   \stex_keys_set:nn{spf}{#1}
7199   \stex_do_for_list:
7200   \stex_do_id:
7201   \exp_args:Ne \stex_annotation:nn{
7202     shtml:proofsketch={
7203       \seq_if_empty:NF \l_stex_fors_seq {
7204         \seq_use:Nn \l_stex_fors_seq ,
7205       }
7206     }
7207   }{
7208     \stex_style_apply:
7209     #2
7210   }
7211   \end{spfesketchenv}
7212 }{
7213   \noindent\emph{\spfesketchenvautorefname :}~

```

```
7214 }
```

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

- `\sproofend` This macro places a little box at the end of the line if there is space, or at the end of the next line if there isn't

```
7215 \tl_set:Nn \__stex_proof_box_tl {
7216   \ltx@ifpackageloaded{amssymb}{$\square$} {
7217     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
7218   }
7219 }
7220
7221 \tl_set:Nn \sproofend {
7222   \tl_if_empty:NF \l_stex_key_proofend_tl {
7223     \hfil\null\nobreak\hfill\l_stex_key_proofend_tl\par\smallskip
7224   }
7225 }
```

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

#### `\stexcommentfont`

```
7226 \cs_new_protected:Npn \stexcommentfont {
7227   \small\itshape
7228 }
```

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

#### `sproof (env.)`

```
7229 \cs_new_protected:Nn \__stex_proof_start_list:n {
7230   \begin{list}{}{
7231     \setlength\topsep{0pt}
7232     \setlength\parsep{0pt}
7233     \setlength\rightmargin{0pt}
7234   }\item[#1]
7235 }
7236 \cs_new_protected:Nn \__stex_proof_end_list: {
7237   \end{list}
7238 }
7239
7240 \cs_new_protected:Nn \__stex_proof_html: {
7241   \stex_annotation_invisible:n{\hbox{
7242     \tl_if_empty:NF \l_stex_key_term_tl {
7243       $\l_stex_annotation:nn{shtml:proofterm={}}{\l_stex_key_term_tl}$}
7244     }
7245     \tl_if_empty:NF \l_stex_key_method_tl {
7246       \stex_annotation:nn{shtml:proofmethod={}}{\l_stex_key_method_tl}
7247     }
7248   }}
7249 }
7250
7251 \cs_new_protected:Nn \__stex_proof_html_env:n {
7252   \exp_args:Nne \begin{stex_annotation_env}{
7253     shtml:#1={
7254       \seq_if_empty:NF \l_stex_fors_seq {
7255         \seq_use:Nn \l_stex_fors_seq ,
```

```

7256     }
7257   }
7258   \bool_if:NT \l_stex_key_hide_bool {
7259     shtml:proofhide=true
7260   }
7261 }
7262 \__stex_proof_html:
7263 }
7264
7265 \bool_set_false:N \l__stex_proof_in_spfblock_bool
7266 \cs_new_protected:Nn \__stex_proof_begin_proof:nn {\par
7267   \intarray_gzero:N \l_stex_proof_counter_intarray
7268   \intarray_gset:Nnn \l_stex_proof_counter_intarray 1 1
7269   \stex_keys_set:nn{spfsteps}{#1}
7270   \stex_do_for_list:
7271   \stex_if_do_html:T {
7272     \__stex_proof_html_env:n{proof}
7273   }
7274   \seq_map_inline:Nn \l_stex_fors_seq {
7275     \stex_debug:nn{definiens}{Adding-definiens-to-##1}
7276     \stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7277   }
7278   \stex_style_apply:
7279   \stex_do_id:
7280   \stex_reactivate_macro:N \subproof
7281   \stex_reactivate_macro:N \spfstep
7282   \stex_reactivate_macro:N \conclude
7283   \stex_reactivate_macro:N \assumption
7284   \stex_reactivate_macro:N \eqstep
7285   \stex_reactivate_macro:N \yield
7286   \stex_reactivate_macro:N \spfblock
7287   \stex_reactivate_macro:N \spfjust
7288   \stex_annotation:nn{shtml:prooftitle={}}{#2}
7289   \stex_if_do_html:T{
7290     \begin{stex_annotation_env}{shtml:proofbody={}}
7291   }
7292 }
7293 \stex_new_stylable_env:nnnnnnn{proof}{0{} m}{
7294   \__stex_proof_begin_proof:nn{#1}{#2}
7295   \bool_set_true:N\l_stex_proof_in_spfblock_bool\__stex_proof_start_list:n{}
7296   \group_begin:\stexcommentfont
7297 }{
7298   \stex_style_apply:
7299   \stex_if_do_html:T{\end{stex_annotation_env}\end{stex_annotation_env}}
7300 }{
7301   \emph{\sproofautorefname :}~
7302 }{
7303   \sproofend
7304 }{s}
7305 \AddToHook{env/sproof/end}{
7306   \bool_if:NT\l__stex_proof_in_spfblock_bool {
7307     \group_end:\__stex_proof_end_list:
7308   }
7309 }

```

```

7310 \stex_new_stylable_env:nnnnnnn{proof*}{0{}}
7311   \__stex_proof_begin_proof:nn{#1}{}
7312   \bool_set_false:N\l__stex_proof_in_spfblock_bool
7313 }{
7314   \stex_style_apply:
7315   \stex_if_do_html:T{\end{stex_annotation_env}\end{stex_annotation_env}}
7316 }{
7317   \emph{Proof:}~
7318 }{
7319   \sproofend
7320 }{s}

subproof (env.)
7321 \str_set_eq:NN \subproofautorefname \spfstepautorefname
7322 \stex_new_stylable_env:nnnnnnn{subproof}{s 0{} m}{\par
7323   \stex_keys_set:nn{spf}{#2}
7324   \stex_do_for_list:
7325   \stex_if_do_html:T {
7326     \__stex_proof_html_env:n{subproof}
7327   }
7328   \seq_map_inline:Nn \l_stex_fors_seq {
7329     \stex_debug:nn{definiens}{Adding-definiens-to~##1}
7330     \stex_add_definiens:nn {##1}{\STEXinvisibl{proven}}
7331   }
7332 }

7333 \IfBooleanTF #1 {
7334   \stex_style_apply:
7335   \str_if_empty:NF \l_stex_key_id_str {
7336     \__stex_proof_number_as_string:N \@currentlabel
7337     \str_set:Nx \@currentHref{subproof.\@currentlabel}
7338     \stex_do_id:
7339   }
7340   \bool_set_false:N \l__stex_proof_in_spfblock_bool
7341   \stex_annotation:nn{shtml:prooftitle={}}{#3}
7342 }{
7343   \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7344     \str_if_empty:NF \l_stex_key_id_str {
7345       \__stex_proof_number_as_string:N \@currentlabel
7346       \str_set:Nx \@currentHref{subproof.\@currentlabel}
7347       \stex_do_id:
7348     }
7349     \__stex_proof_start_list:n\__stex_proof_insert_number:
7350       \stex_annotation:nn{shtml:prooftitle={}}{#3}
7351       \__stex_proof_add_counter:
7352       \stex_style_apply:
7353     }{
7354       \stex_annotation:nn{shtml:prooftitle={}}{#3}
7355       \stex_style_apply:
7356       \stex_do_id:
7357     }
7358   }
7359 }
7360 \stex_if_do_html:T{
7361   \begin{stex_annotation_env}{shtml:proofbody={}}

```

```

7362   }
7363   \bool_if:NT \l__stex_proof_in_spfblock_bool {\group_begin:\stexcommentfont}
7364 }{
7365   \stex_style_apply:
7366   \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_inc_counter:
7367   \stex_if_do_html:T{\end{stex_annotation_env}}
7368   \bool_if:NT\l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
7369   \stex_if_do_html:T{\end{stex_annotation_env}}
7370   \aftergroup \__stex_proof_inblock_restore:
7371 }{}{}{}
7372 \AddToHook{env/subproof/before}{
7373   \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7374 }
7375 \AddToHook{env/subproof/end}{
7376   \bool_if:NT\l__stex_proof_in_spfblock_bool {
7377     \group_end:\__stex_proof_remove_counter:
7378     \%__stex_proof_end_list:
7379   }
7380 }
7381 \stex_deactivate_macro:Nn \subproof {sproof~environments}
7382
7383 \cs_new_protected:Nn \__stex_proof_inblock_restore: {
7384   \bool_if:NT\l__stex_proof_in_spfblock_bool {
7385     \group_begin:\stexcommentfont
7386   }
7387 }

\spfstep
\conclude
7388
\assumption
7389 \stex_keys_define:nnnn { spfsteps } {
7390   \clist_clear:N \l_stex_key_for_clist
7391   \str_clear:N \l_stex_key_name_str
7392   \tl_clear:N \l_stex_key_method_tl
7393   \tl_clear:N \l_stex_key_term_tl
7394 }{
7395   for .clist_set:N = \l_stex_key_for_clist ,
7396   method .tl_set:N = \l_stex_key_method_tl,
7397   term .tl_set:N = \l_stex_key_term_tl,
7398   name .str_set_x:N = \l_stex_key_name_str
7399   % todo: style=inline
7400 }{id,style,title}
7401
7402 \newenvironment{spfstepenv}{
7403   \str_set_eq:NN \spfstepenvautorefname \spfstepautorefname
7404 }{}{7405}
7406 \cs_new_protected:Nn \__stex_proof_step_html:nn {
7407   \stex_if_do_html:TF{
7408     \exp_args:Ne \stex_annotation:n{7409
7410       shtml:spf#1={7411
7412         \seq_if_empty:NF \l_stex_fors_seq {
7413           \seq_use:Nn \l_stex_fors_seq ,7414
7415         }7416
7417     }7418
7419   }
7420 }
```

```

7414     \str_if_empty:NF \l_stex_key_name_str {
7415         shtml:stepname={\l_stex_key_name_str}
7416     }
7417     }{
7418         \__stex_proof_html:
7419         #2
7420     }
7421     }{ #2 }
7422 }
7423
7424 \cs_new_protected:Nn \__stex_proof_make_step_macro:Nnnnn {
7425     \NewDocumentCommand #1 {s O{} +m} {
7426         \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7427         \stex_keys_set:nn{spfsteps}{##2}
7428         \str_if_empty:NF \l_stex_key_name_str {
7429             \stex_debug:nn{Here:}{Variable~\l_stex_key_name_str}
7430             \exp_args:Nne\use:nn{\vardef}{{v}\l_stex_key_name_str}[name=\l_stex_key_name_str]{\l_st
7431         }
7432
7433         \begin{spfstepenv}
7434             \str_if_empty:NF \l_stex_key_id_str {
7435                 \__stex_proof_number_as_string:N \@currentlabel
7436                 \str_set:Nx \@currentHref{spfstep.\@currentlabel}
7437                 \__stex_do_id:
7438             }
7439
7440             \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7441                 \IfBooleanTF ##1 {
7442                     \__stex_proof_step_html:nn{#2}{##3}
7443                 }{
7444                     \__stex_proof_step_html:nn{#2}{\__stex_proof_start_list:n{#3} ##3 \__stex_proof_end_#
7445                     #5
7446                 }
7447                 \end{spfstepenv}
7448                 \group_begin:\stexcommentfont
7449             }{
7450                 \__stex_proof_step_html:nn{#2}{##3}
7451                 \end{spfstepenv}
7452             }
7453         }
7454         \stex_deactivate_macro:Nn #1 {sproof-environments}
7455     }
7456
7457     \__stex_proof_make_step_macro:Nnnnn \assumption {assumption} \__stex_proof_insert_number: {}
7458     \__stex_proof_make_step_macro:Nnnnn \conclude {conclusion} {$\Rightarrow$} {} {}
7459     \__stex_proof_make_step_macro:Nnnnn \spfstep {step} \__stex_proof_insert_number: {} \__stex_
7460
7461     \NewDocumentCommand \eqstep {s m} {
7462         \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7463             \group_end:
7464             \IfBooleanTF #1 {
7465                 \__stex_proof_step_html:nn{eqstep}{$= #2$}
7466             }{
7467                 \__stex_proof_step_html:nn{eqstep}{\__stex_proof_start_list:n{$=$} $#2$ \__stex_proof_

```

```

7468     }
7469     \group_begin:\stexcommentfont
7470   }{
7471     \__stex_proof_step_html:nn{eqstep}{$= #2$}
7472   }
7473 }
7474 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
7475
7476 \NewDocumentCommand \yield {+m} {
7477   \stex_annotate:nn{shtml:proofterm={}}{#1}
7478 }
7479 \stex_deactivate_macro:Nn \yield {sproof~environments}
7480
7481 \NewDocumentEnvironment{spfblock}{}{
7482   \bool_set_false:N \l__stex_proof_in_spfblock_bool
7483 }
7484   \aftergroup\__stex_proof_inblock_restore:
7485 }
7486 \stex_deactivate_macro:Nn \spfblock {sproof~environments}
7487 \AddToHook{env/spfblock/before} {
7488   \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7489 }
7490
7491 \newcommand\spfjust[1]{
7492   \stex_annotate:nn{spfjust={}}{#1}
7493 }
7495 \stex_deactivate_macro:Nn \spfjust {sproof~environments}

```

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

## 13.12 Metatheory

```

7496 <@=stex_meta>
7497 \group_begin:
7498   \cs_set:Npn \__stex_modules_persist_module: {}
7499   \cs_set:Npn \stex_check_term:n #1 {}
7500   \cs_set:Npn \__stex_sref_do_aux:n #1 { #1 }
7501   \bool_set_false:N \stex_html_do_output_bool
7502   \bool_set_false:N \c_stex_check_terms_bool
7503   \stex_uri_resolve:Nn \l_stex_current_ns_uri {http://mathhub.info/sTeX/meta}
7504   \stex_module_setup:n{Metatheory}
7505
7506   \symdef{of~type}[args=ii,invisible]{#1}
7507   \notation{of~type}[colon]{#1 \mathbin{\comp{}} #2}
7508
7509   \symdef{apply}[args=ia,prec=0;\infprec x\infprec]{#1\mathopen{\comp{}} #2 \mathclose{\comp{}}}
7510   \notation{apply}[\lambda]{#1\mathbin{\lambda}\mathbin{\argsep}{#2}\mathbin{;}}
7511   \notation{apply}[infixop]{\argsep{#2}\mathbin{\mathrel{#1}}}
7512   \notation{apply}[infixrel]{\argsep{#2}\mathbin{\mathrel{#1}}}
7513
7514 % structures
7515 \symdef{module~type}[args=i,op=\mathtt{MOD}]{\mathopen{\comp{\mathit{MOD}}}\mathclose{\comp{}}}
7516

```

```

7517 \symdef{module~type~merge}[args=a,op=\oplus]
7518   {\argsep{\#1}{\mathbin{\comp{\oplus}}}}
7519 \symdef{anonymous~record}[args=a]
7520   {\mathopen{\comp{[]}\#1\mathclose{\comp{}}}}
7521 \symdef{record~field}[args=2]{\#1\comp{.}\#2}
7522 \symdecl*{record-type}

7523
7524 \symdecl{mathstruct}[name=mathematical~structure,args=a] % TODO
7525 \notation{mathstruct}[angle,prec=nobrackets]
7526   {\mathopen{\comp{\langle}}\#1\mathclose{\comp{\rangle}}}
7527 \notation{mathstruct}[parens,prec=nobrackets]
7528   {\mathopen{\comp{()}\#1\mathclose{\comp{}}}}

7529
7530 % sequences
7531 \symdef{ellipses}[ldots]{\ldots}
7532 \symdef{sequence-expression}[comma,args=a]{\#1}
7533 \symdef{sequence-type}[args=1]{\#1^{\comp{\ast}}}
7534 \symdef{sequence-map}[args=ia]{
7535   \comp{\mathrm{map}}\mathopen{\comp{()}\#1\mathpunct{\comp{,}}}
7536   \#2\mathclose{\comp{()}}}
7537 }
7538 \iffalse
7539 % binder (\forall, \Pi, \lambda etc.)
7540 \symdef{pibind}[name=dependent function type,prec=nobrackets,
7541   op=(\cdot)\; ;\cdot\cdot, args=Bi,assoc=pre]
7542   {\argmap{\#1}{%
7543     \mathopen{\comp{()}\#1\mathclose{\comp{()}}}
7544     }{\mathbin{\comp{\rightarrow}}}\mathbin{\comp{\rightarrow}}\#2}
7545 \notation{pibind}[\forall]{\comp{\forall}\#1\mathpunct{\comp{.}}\#2}
7546 \notation{pibind}[\Pi]{\mathop{\comp{\prod}}\c_math_subscript_token{\#1}\#2}

7547
7548 \symdef{mapbind}[name=lambda,mapsto,prec=nobrackets,op=\mapsto,args=Bi,assoc=pre]
7549   {\#1\mathrel{\comp{\mapsto}}\#2}
7550 \notation{mapbind}[\lambda,prec=nobrackets,op=\lambda]
7551   {\comp{\lambda}\#1\mathpunct{\comp{.}}\#2}
7552 \fi
7553 \symdecl{bind}[args=Bi,assoc=pre]
7554 \notation{bind}[defun,prec=nobrackets,op=(\cdot)\; ;\cdot\cdot]
7555   {\mathopen{\comp{()}\#1\mathclose{\comp{()}}}\mathbin{\comp{\rightarrow}}\#2}
7556 \notation{bind}[\forall]{\comp{\forall}\#1.\#2}
7557 \notation{bind}[\Pi]{\mathop{\comp{\prod}}\c_math_subscript_token{\#1}\#2}

7558
7559 \symdef{implicit-bind}[args=Bi,assoc=pre]{\mathopen{\comp{\{}}\#1\mathclose{\comp{\}}}\c_math_
7560
7561 \symdecl*{integer-literal}
7562 \notation{integer-literal}{\mathbb{Z}}
7563
7564 \symdecl*{ordinal}
7565 \notation{ordinal}{\mathtt{Ord}}
7566
7567 % propositions
7568 \symdef{prop}[name=proposition]{\mathtt{Prop}}
7569 \symdef{judgment-holds}[args=i,role=judgment]{\comp{\vdash}\#1}
7570

```

```

7571 % any object
7572 \symdef{object}{\mathhtt{Obj}}
7573
7574 % TODO DELETE
7575 \symdef{aseqdots}[args=a,prec=nobrackets]
7576   {#1\comp{,\ldots}}%{##1\comp,##2}
7577 \symdef{aseqfromto}[args=ai,prec=nobrackets]
7578   {#1\comp{,\ldots},#2}%{##1\comp,##2}
7579 \symdef{aseqfromtovia}[args=a ii,prec=nobrackets]
7580   {#1\comp{,\ldots},#2\comp{,\ldots},#3}%{##1\comp,##2}
7581
7582
7583 \stex_close_module:
7584 \stex_uri_add_module:NNn \l_stex_metatheory_uri \l_stex_current_ns_uri {Metatheory}
7585 \global \let \l_stex_metatheory_uri \l_stex_metatheory_uri
7586 \global \let \c_stex_default_metatheory \l_stex_metatheory_uri
7587 \group_end:

```

### 13.13 MMT Interfaces

```

7588 <@=todo>
7589 \cs_new_protected:Npn \MSC #1 {}

```

\MMTinclude

```

7590 \stex_new_stylable_cmd:nnnn{MMTinclude}{m}{}
7591   \stex_annotation_invisible:nn{shtml:import={#1}}{}
7592 }{}}
7593 \stex_deactivate_macro:Nn \MMTinclude {module~environments}
7594 \stex_every_module:n {\stex_reactivate_macro:N \MMTinclude}

```

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

\MMTrule

```

7595 \NewDocumentCommand \MMTrule {m m} {
7596   \tl_if_empty:nTF{#2}{\seq_clear:N \l_tmpa_seq}{%
7597     \seq_set_split:Nnn \l_tmpa_seq , {#2}}
7598   }
7599   \int_zero:N \l_tmpa_int
7600   \stex_annotation_invisible:n{
7601     $
7602     \stex_annotation:nn{shtml:rule={scala://#1}}{%
7603       \stex_annotation_force_break:n{%
7604         \seq_if_empty:NF \l_tmpa_seq {%
7605           \seq_map_inline:Nn \l_tmpa_seq {%
7606             \int_incr:N \l_tmpa_int
7607             \stex_annotation:nn{%
7608               shtml:argmode=i,
7609               shtml:arg={\int_use:N \l_tmpa_int}
7610             }{ ##1 }
7611           }
7612         }
7613       }
7614     }$}
7615   }
7616 }

```

```

7617 \stex_deactivate_macro:Nn \MMTrule {module~environments}
7618 \stex_every_module:n{\stex_reactivate_macro:N \MMTrule}

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

mmtinterface (env.)
7619 \NewDocumentEnvironment { mmtinterface } { O{} m m } {
7620   \stex_module_setup_top_nosig:n { #3 }
7621   \str_set_eq:NN \l__todo_mmt_module_str \l_stex_current_module_str
7622   \str_clear:N \l_stex_current_module_str
7623   \stex_keys_set:nn { smodule }{ #1 }
7624   \stex_module_setup:n{ #2 }
7625   \str_set_eq:NN \l__todo_stex_module_str \l_stex_current_module_str
7626   \stex_debug:nn{mmt}{Interface~\l__todo_stex_module_str^~Jfor~\l__todo_mmt_module_str}
7627
7628 \stex_if_do_html:T {
7629   \exp_args:Nne \begin{stex_annotation_env} {
7630     shtml:theory={\l_stex_current_module_str},
7631     shtml:language={ \l_stex_current_language_str },
7632     shtml:signature={}
7633     \tl_if_empty:NF \l_stex_metatheory_uri {},
7634     shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
7635   }
7636 }
7637 \stex_annotation_invisible:n{}
7638 \stex_annotation_invisible:nn
7639   {shtml:import=\l__todo_mmt_module_str} {}
7640 }
7641 \stex_module_add_code:x{
7642   \stex_activate_module:n{ \l__todo_mmt_module_str }
7643 }
7644 \stex_module_add_morphism:nonn
7645   {}{\l__todo_mmt_module_str}{import}{}
7646 \stex_reactivate_macro:N \mmtdef
7647 \stex_smsmode_do:
7648 }{
7649   \str_set_eq:NN \l_stex_current_module_str \l__todo_mmt_module_str
7650   \stex_close_module:
7651   \str_set_eq:NN \l_stex_current_module_str \l__todo_stex_module_str
7652   \stex_close_module:
7653   \stex_if_do_html:T { \end{stex_annotation_env} }
7654 }
7655 \stex_sms_allow_env:n{mmtinterface}

\mmtdef
7656 \NewDocumentCommand \mmtdef {m O{} m} {
7657   \stex_keys_set:nn{symdef}{#2}
7658   \str_set:Nx \l_stex_macroname_str { #1 }
7659   \str_if_empty:NT \l_stex_key_name_str {
7660     \str_set:Nx \l_stex_key_name_str { #1 }
7661   }
7662   \stex_symdecl_do:
7663
7664   \str_set_eq:NN \l_stex_current_module_str \l__todo_mmt_module_str

```

```

7665 \cs_set_eq:NN \l__todo_old_metagroup_cd \stex_metagroup_do_in:nn
7666 \cs_set_protected:Npn \stex_metagroup_do_in:nn ##1 ##2 {##2}
7667 \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN} {
7668   {\l_stex_mroname_str}
7669   {\l_stex_key_name_str}
7670   {\int_use:N \l_stex_get_symbol_arity_int}
7671   {\l_stex_get_symbol_args_tl}
7672   {}
7673   {}
7674   {}
7675   \stex_invoke_symbol:
7676 }
7677 \cs_set_eq:NN \stex_metagroup_do_in:nn \l__todo_old_metagroup_cd
7678 \str_set_eq:NN \l_stex_current_module_str \l__todo_stex_module_str
7679
7680 \str_set_eq:NN \l_stex_get_symbol_mod_str \l__todo_mmt_module_str
7681 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
7682 \stex_notation_parse:n{#3}
7683 \_stex_notation_check:
7684 \_stex_notation_add:
7685 \stex_if_do_html:T{
7686   \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7687 }
7688 \stex_smsmode_do:
7689 }
7690 \stex_deactivate_macro:Nn \mmtdef {mmtinterface-environments}
7691 \stex_sms_allow_escape:N \mmtdef

```

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

#### VoLL-KI Annotations

```

7692 \newcommand\precondition[2]{
7693   \str_clear:N \l_stex_get_symbol_name_str
7694   \stex_get_symbol:n{#2}
7695   \str_if_empty:NTF \l_stex_get_symbol_name_str{
7696     \errmessage{Unknown-symbol~#2}
7697   }{
7698     \str_case:nnTF {#1}{
7699       {remember}{}
7700       {understand}{}
7701       {analyze}{}
7702       {evaluate}{}
7703       {apply}{}
7704       {create}{}
7705     }{
7706       \stex_annotation_invisible:nn{
7707         shtml:preconditionsymbol={\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str},
7708         shtml:preconditiondimension={#1}
7709       }{}
7710     }{\errmessage{Unknown-cognitive-dimension~#1}}
7711   }
7712 }
7713 \newcommand\objective[2]{
7714   \str_clear:N \l_stex_get_symbol_name_str
7715   \stex_get_symbol:n{#2}

```

```

7716   \str_if_empty:NTF \l_stex_get_symbol_name_str{
7717     \errmessage{Unknown~symbol~#2}
7718   }{
7719   \str_case:nnTF {#1}{
7720     {remember}){}
7721     {understand}){}
7722     {analyze}){}
7723     {evaluate}){}
7724     {apply}){}
7725     {create}){}
7726   }{
7727     \stex_annotation_invisible:nn{
7728       shtml:objectivesymbol={\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str},
7729       shtml:objectivedimension={#1}
7730     }{}
7731   }{\errmessage{Unknown~cognitive~dimension~#1}}
7732 }
7733 }

7734 \seq_if_empty:NT \g_stex_current_file {
7735   \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
7736 }
7737 \_stex_persist_read_now:
7738 \_stex_every_file:
7739 \cs_new_protected:Nn \__todo_newlabel:n {
7740   \exp_args:Ne \__todo_old_newlabel:{\tl_to_str:n{#1}}
7741 }
7742 \AtBeginDocument{
7743   \iow_now:Nn \@auxout {
7744     \ExplSyntaxOn
7745     \let\__todo_old_newlabel:\newlabel
7746     \let\newlabel\__todo_newlabel:n
7747     \ExplSyntaxOff
7748   }
7749 }
7750 </package>

```

# Chapter 14

## Additional Packages

### 14.1 Implementation: The `notesslides` Package

#### 14.1.1 Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` package. We pass the `nontheorem` option to the `statements` package when we are not in notes mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
7751 <*cls>
7752 @@=notesslides
7753 \ProvidesExplClass{notesslides}{2023/10/13}{3.4.0}{notesslides Class}
7754 \RequirePackage{13keys2e}
7755
7756 \str_const:Nn \c__notesslides_class_str {article}
7757
7758 \keys_define:nn{notesslides / cls}{
7759   class .str_set_x:N = \c__notesslides_class_str,
7760   notes .bool_set:N = \c__notesslides_notes_bool ,
7761   slides .code:n    = { \bool_set_false:N \c__notesslides_notes_bool },
7762   %docopt .str_set_x:N = \c__notesslides_docopt_str,
7763   unknown .code:n   = {
7764     \PassOptionsToClass{\CurrentOption}{beamer}
7765     \PassOptionsToClass{\CurrentOption}{\c__notesslides_class_str}
7766     \PassOptionsToPackage{\CurrentOption}{notesslides}
7767     \PassOptionsToPackage{\CurrentOption}{stex}
7768   }
7769 }
7770 \ProcessKeysOptions{ notesslides / cls }
7771
7772 \RequirePackage{stex}
7773 \stex_if_html_backend:T {
7774   \bool_set_true:N \c__notesslides_notes_bool
7775 }
7776
7777 \bool_if:NTF \c__notesslides_notes_bool {
7778   \PassOptionsToPackage{notes=true}{notesslides}
7779   \message{notesslides.cls:-Formatting-document-in-notes-mode}
```

```

7780 }{
7781   \PassOptionsToPackage{notes=false}{notesslides}
7782   \message{notesslides.cls:~Formatting~document~in~slides~mode}
7783 }
7784
7785 \bool_if:NTF \c_notesslides_notes_bool {
7786   \LoadClass{\c__notesslides_class_str}
7787 }{
7788   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7789   \%newcounter{Item}
7790   \%newcounter{paragraph}
7791   \%newcounter{ subparagraph}
7792   \%newcounter{Hfootnote}
7793 }
7794 \RequirePackage{notesslides}
7795 
```

now we do the same for the `notesslides` package.

```

7796 <*package>
7797 \ProvidesExplPackage{notesslides}{2023/10/13}{3.4.0}{notesslides Package}
7798 \RequirePackage{l3keys2e}
7799
7800 \keys_define:nn{notesslides / pkg}{
7801   notes          .bool_set:N  = \c_notesslides_notes_bool ,
7802   slides         .code:n    = { \bool_set_false:N \c_notesslides_notes_bool },
7803   sectocframes  .bool_set:N  = \c_notesslides_sectocframes_bool ,
7804   topsect        .str_set_x:N = \c_notesslides_topsect_str,
7805   unknown        .code:n    = {
7806     \PassOptionsToPackage{\CurrentOption}{stex}
7807     \PassOptionsToPackage{\CurrentOption}{tikzinput}
7808   }
7809 }
7810 \ProcessKeysOptions{ notesslides / pkg }
7811
7812 \RequirePackage{stex}
7813 \stex_if_html_backend:T {
7814   \bool_set_true:N\c_notesslides_notes_bool
7815 }
7816
7817 \cs_set:Npn \sectiontitleemph #1 {
7818   \textbf{\Large #1}
7819 }
7820
7821 \newif\ifnotes
7822 \bool_if:NTF \c_notesslides_notes_bool {
7823   \notestrue
7824   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7825   \RequirePackage[noamsthm,hyperref]{beamerarticle}
7826   \RequirePackage{mdframed}
7827   \str_if_empty:NTF \c_notesslides_topsect_str{
7828     \%setsectionlevel{section}
7829   }{
7830     \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7831   }

```

```

7832 }{
7833   \notesfalse
7834
7835   \cs_new_protected:Nn \__notesslides_do_sectocframes: {
7836     \cs_set_protected:Nn \__notesslides_do_label:n {
7837       \str_case:nnF{##1} {
7838         {part} {
7839           \tl_set:Nx\l__notesslides_num{\the part}
7840           \tl_set:cx{@ @ label}{%
7841             \cs_if_exist:NTF\parttitlename{\exp_not:N\parttitlename}{\exp_not:N\partname}{-}}
7842         }
7843         {chapter} {
7844           \tl_set:Nx\l__notesslides_num{\the chapter}
7845           \tl_set:cx{@ @ label}{%
7846             \cs_if_exist:NTF\chapertitlename{\exp_not:N\chapertitlename}{\exp_not:N\chaptername}{-}}
7847         }
7848         {section} {
7849           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesection.}
7850           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7851         }
7852         {subsection} {
7853           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsection.}
7854           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7855         }
7856         {subsubsection} {
7857           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsubsection.}
7858           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7859         }
7860         {paragraph} {
7861           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubparagraph.}
7862           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7863         }
7864       }{
7865         \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsubsubsection.}
7866         \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7867       }
7868     }
7869     \cs_set_protected:Nn \_sfragment_do_level:nn {
7870       \tl_if_exist:cT{c@##1}{\stepcounter{##1}}
7871       \addcontentsline{toc}{##1}{\protect\numberline{\use:c{the##1}}##2}
7872       \__notesslides_do_label:n{##1}
7873       \pdfbookmark[\int_use:N \l_stex_docheader_sect]{\l__notesslides_num##2}{##1.\l__note}
7874       \begin{frame}[noframenumbering]
7875         \vfill\centering
7876         \sectiontitleemph{%
7877           \use:c{@ @ label} ##2
7878         }
7879         \end{frame}
7880         \int_incr:N \l_stex_docheader_sect
7881         \tl_set:Nn \stex_current_section_level{##1}
7882     }
7883   }
7884
7885 \AtBeginDocument{

```

```

7886 \str_if_empty:NTF \c_notesslides_topsect_str {
7887   \setsectionlevel{section}
7888 } {
7889   \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7890   \exp_args:No \str_if_eq:nnTF \c_notesslides_topsect_str {chapter} {
7891     \__notesslides_define_chapter:
7892   }{
7893     \exp_args:No \str_if_eq:nnT \c_notesslides_topsect_str {part} {
7894       \__notesslides_define_chapter:
7895       \__notesslides_define_part:
7896     }
7897   }
7898 }
7899 }
7900
7901 \bool_if:NT \c_notesslides_sectocframes_bool {
7902   \__notesslides_do_sectocframes:
7903 }
7904 }
7905
7906 \cs_new_protected:Nn \__notesslides_define_chapter: {
7907   \cs_if_exist:NF \chaptername {
7908     \cs_set_protected:Npn \chaptername {Chapter}
7909   }
7910   \cs_if_exist:NF \chapter {
7911     \cs_set_protected:Npn \chapter {INVALID}
7912   }
7913   \cs_if_exist:NF \c@chapter {
7914     \newcounter{chapter}\counterwithin*{section}{chapter}
7915   }
7916 }
7917
7918 \cs_new_protected:Nn \__notesslides_define_part: {
7919   \cs_if_exist:NF \partname {
7920     \cs_set_protected:Npn \partname {Part}
7921   }
7922   \cs_if_exist:NF \part {
7923     \cs_set_protected:Npn \part {INVALID}
7924   }
7925   \cs_if_exist:NF \c@part {
7926     \newcounter{part}\counterwithin*{chapter}{part}
7927   }
7928 }

```

### \prematurestop

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

```

7929 \def \c_notesslides_document_str{document}
7930 \newcommand\afterprematurestop{}
7931 \def\prematurestop@endsfragment{
7932   \unless\ifx\@currenvir\c_notesslides_document_str
7933     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7934     \expandafter\prematurestop@endsfragment
7935   \fi
7936 }

```

```

7937 \providecommand\prematurestop{
7938   \stex_if_html_backend:F{
7939     \message{Stopping-sTeX-processing-prematurely}
7940     \prematurestop@endsfragment
7941   \afterprematurestop
7942   \end{document}
7943 }
7944 }
```

(End of definition for `\prematurestop`. This function is documented on page 108.)

### 14.1.2 Notes and Slides

For the notes case, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class.

```

7945 \bool_if:NT \c_notesslides_notes_bool {
7946   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamertheme#2}}
7947 }
7948 \NewDocumentCommand \libusetheme {O{} m} {
7949   \libusepackage[#1]{beamertheme#2}
7950 }
```

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

```

7951 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7952 \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.

```

7953 \ifnotes
7954
7955 \newlength{\slideframewidth}
7956 \setlength{\slideframewidth}{1.5pt}
```

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

```

7957 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7958   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7959     \bool_set_true:N #1
7960   }{
7961     \bool_set_false:N #1
7962   }
7963 }
7964
7965 \stex_keys_define:nnnn{notesslides / frame}{
7966   \str_clear:N \l__notesslides_frame_label_str
7967   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7968   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7969   \bool_set_true:N \l__notesslides_frame_fragile_bool
7970   \bool_set_true:N \l__notesslides_frame_shrink_bool
7971   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7972   \bool_set_true:N \l__notesslides_frame_t_bool
7973 }{
7974   label .str_set_x:N = \l__notesslides_frame_label_str,
7975   allowframebreaks .code:n = {
```

```

7976     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7977 },
7978 allowdisplaybreaks .code:n      = {
7979     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7980 },
7981 fragile          .code:n      = {
7982     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7983 },
7984 shrink           .code:n      = {
7985     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7986 },
7987 squeeze          .code:n      = {
7988     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7989 },
7990 t                .code:n      = {
7991     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7992 },
7993 unknown         .code:n      = {}
7994 }{}
```

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

```

7995 \cs_new_protected:Nn \__notesslides_setup_itemize: {
7996     \def\itemize@level{outer}
7997     \def\itemize@outer{outer}
7998     \def\itemize@inner{inner}
7999     \%newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
8000     \renewenvironment{itemize}{
8001         \ifx\itemize@level\itemize@outer
8002             \def\itemize@label{$\rhd$}
8003         \fi
8004         \ifx\itemize@level\itemize@inner
8005             \def\itemize@label{$\scriptstyle\rhd$}
8006         \fi
8007         \begin{list}
8008             {\itemize@label}
8009             {\setlength{\labelsep}{.3em}
8010                 \setlength{\labelwidth}{.5em}
8011                 \setlength{\leftmargin}{1.5em}
8012             }
8013             \edef\itemize@level{\itemize@inner}
8014         }{
8015             \end{list}
8016         }
8017     }
```

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

```

8018 \stex_if_html_backend:TF {
8019     \cs_new_protected:Nn \__notesslides_frame_box_begin: {
8020         \vbox\bgroup
8021         \begin{stex_annotation_env}[shtml:frame={}]
8022             \mdf@patchamsthm\notesslidesfont
8023         }
8024     \cs_new_protected:Nn \__notesslides_frame_box_end: {
8025         %^A \notesslides@slidelabel
```

```

8026     \medskip\par\noindent\tiny\notesslidesfooter
8027     \end{stex_annotation_env}\egroup
8028 }
8029 }{
8030     \cs_new_protected:Nn \__notesslides_frame_box_begin: {
8031         \begin{mdframed}[
8032             linewidth=\slideframewidth,
8033             skipabove=1ex,
8034             skipbelow=1ex,
8035             userdefinedwidth=\slidewidth,
8036             align=center
8037         ]\notesslidesfont
8038     }
8039     \cs_new_protected:Nn \__notesslides_frame_box_end: {
8040         \medskip\par\noindent\tiny\notesslidesfooter%^^A\notesslides@slidelabel
8041         \end{mdframed}
8042     }
8043 }

```

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

```

8044 \renewenvironment{frame}[1][]{
8045     \stex_keys_set:nn{notesslides / frame}{#1}
8046     \stepcounter{framenumber}
8047     \renewcommand{\newpage}{\addtocounter{framenumber}{1}}
8048     \def\@currentlabel{\theframenumber}
8049     \str_if_empty:NF \l__notesslides_frame_label_str {
8050         \label{\l__notesslides_frame_label_str}
8051     }
8052     \__notesslides_setup_itemize:
8053     \__notesslides_frame_box_begin:
8054 }{
8055     \__notesslides_frame_box_end:
8056 }

```

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

```
\frametitle
8057 \renewcommand{\frametitle}[1]{
8058     \stexdoctitle { #1 }
8059     \notesslidestitlenameph{#1}\medskip
8060 }
```

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

```
\pause
8061 \newcommand{\pause}{}
```

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

We redefine the `columns` and `column` environments:

```

8062 \renewenvironment{columns}[1][]{
8063     \par\noindent
8064     \begin{minipage}
8065         \slidewidth\centering\leavevmode
8066 % \stex_if_html_backend:T{
```

```

8067 %     \cs_if_exist:NT \rustex_if:T {
8068 %         \rustex_if:T {\par
8069 %             \rustex_direct_HTML:n{<table><tr><td>}
8070 %         }
8071 %     }
8072 % }
8073 }{
8074 % \stex_if_html_backend:T{
8075 %     \cs_if_exist:NT \rustex_if:T {
8076 %         \rustex_if:T {\par
8077 %             \rustex_direct_HTML:n{</td></tr></table>}
8078 %         }
8079 %     }
8080 % }
8081     \end{minipage}\par\noindent
8082 }
8083 \newsavebox\columnbox
8084 \renewenvironment<>{column}[2] []{
8085     \begin{lrbox}{\columnbox}
8086 % \stex_if_html_backend:T{
8087 %     \cs_if_exist:NT \rustex_if:T {
8088 %         \rustex_if:T {\par
8089 %             \rustex_direct_HTML:n{</td><td>}
8090 %         }
8091 %     }
8092 % }
8093     \begin{minipage}{#2}
8094 }{
8095     \end{minipage}
8096 % \stex_if_html_backend:T{
8097 %     \cs_if_exist:NT \rustex_if:T {
8098 %         \rustex_if:T {\par
8099 %             \rustex_direct_HTML:n{</td><td>}
8100 %         }
8101 %     }
8102 % }
8103     \end{lrbox}\usebox\columnbox
8104 }
8105 \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.

```

8106 \bool_if:NTF \c_notesslides_notes_bool {
8107     \renewenvironment{note}{\ignorespaces}{}
8108 }{
8109     \renewenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup\egroup}
8110 }

```

For other environments we introduce variants prefixed with `n`, which are excluded in `slides` mode.

```

8111 \cs_new_protected:Nn \__notesslides_notes_env:n {
8112   \bool_if:NTF \c_notesslides_notes_bool {
8113     \newenvironment{#1}{#2}{#3}{#4}
8114   }{
8115     \newenvironment{#1}{#2{
8116       \cs_set:Npn \__notesslides_eat: #####1 \end #####2 {
8117         \str_if_eq:nnTF{#1}{#####2} {
8118           \end{#1}
8119         }{
8120           \__notesslides_eat:
8121         }
8122       }
8123       \__notesslides_eat:
8124       \%setbox\l_tmpa_box\vbox\bgroup#3
8125     }{
8126       %#4\egroup
8127     }
8128   }
8129 }
8130
8131 \__notesslides_notes_env:nnnn{nparagraph}{[1]}{\begin{sparagraph}{#1}}{\end{sparagraph}}
8132 \__notesslides_notes_env:nnnn{nfragment}{[2]}{\begin{sfragment}{#1}{#2}}{\end{sfragment}}
8133 \__notesslides_notes_env:nnnn{ndefinition}{[1]}{\begin{sdefinition}{#1}}{\end{sdefinition}}
8134 \__notesslides_notes_env:nnnn{nassertion}{[1]}{\begin{sassertion}{#1}}{\end{sassertion}}
8135 \__notesslides_notes_env:nnnn{nproof}{[2]}{\begin{sproof}{#1}{#2}}{\end{sproof}}
8136 \__notesslides_notes_env:nnnn{nexample}{[1]}{\begin{sexample}{#1}}{\end{sexample}}
8137
8138 \RequirePackage{graphicx}
8139
8140 \NewDocumentCommand\frameimage{s O{} m} {
8141   \IfBooleanTF #1 {
8142     \begin{frame}[plain]
8143   }{
8144     \begin{frame}
8145   }
8146   \bool_if:NTF \c_notesslides_notes_bool {
8147     \slidewidth=\dimexpr\slidewidth-(2\slideframewidth)\relax
8148   }{
8149     \slidewidth=\textwidth\relax
8150   }
8151   \def\Gin@ewidth{}\setkeys{Gin}{#2}
8152   \tl_if_empty:NTF \Gin@ewidth {
8153     \mhgraphics[width=\slidewidth,#2]{#3}
8154   }{
8155     \mhgraphics[#2]{#3}
8156   }
8157   \end{frame}
8158 }

```

hacking inputref:

\inputref\*

```

8159 \cs_set_eq:NN \__notesslides_inputref:\inputref
8160 \cs_set_protected:Npn \inputref{@ifstar\ninputref\__notesslides_inputref:}

```

```

8161 \bool_if:NTF \c_notesslides_notes_bool {
8162   \newcommand\ninputref[2][]{%
8163     \__notesslides_inputref:[#1]{#2}%
8164   }%
8165 }{%
8166   \newcommand\ninputref[2]{}%
8167 }

```

(End of definition for `\inputref*`. This function is documented on page 107.)

#### 14.1.4 Styling Across Notes/Slides

```

8168 \def\notesslidestitleemph#1{%
8169   {\Large\bf\sf#1}%
8170   \vskip0.1\baselineskip
8171   \leaders\vrule width \textwidth
8172   \vskip0.4pt%
8173   \nointerlineskip
8174 }%
8175
8176 \def\notesslidesfooter{}%
8177
8178 \let\notesslidesfont\sffamily

```

#### 14.1.5 Beamer Compatibility

All of this should be removed and made part of a template

```

8179
8180 \stex_if_do_html:TF{%
8181   \NewDocumentEnvironment{slideshow}{}{%
8182     \begin{stex_annotation_env}{\shtml:slideshow={}}%
8183     \cs_set_protected:Npn \nextslide ##1 {%
8184       \begin{stex_annotation_env}{\shtml:slideshow-slide={}} ##1%
8185       \end{stex_annotation_env}%
8186     }%
8187     \cs_set_protected:Npn \lastslide ##1 {%
8188       \begin{stex_annotation_env}{\shtml:slideshow-slide={}} ##1%
8189       \end{stex_annotation_env}%
8190     }%
8191   }{%
8192     \end{stex_annotation_env}%
8193   }%
8194 }{%
8195   \int_new:N \l__notesslides_slideshow_counter_int
8196   \NewDocumentEnvironment{slideshow}{}{%
8197     \int_zero:N \l__notesslides_slideshow_counter_int
8198     \cs_set_protected:Npn \nextslide ##1 {%
8199       \int_incr:N \l__notesslides_slideshow_counter_int
8200       \only<\int_use:N \l__notesslides_slideshow_counter_int>{##1}%
8201     }%
8202     \cs_set_protected:Npn \lastslide ##1 {%
8203       \int_incr:N \l__notesslides_slideshow_counter_int
8204       \only<\int_use:N \l__notesslides_slideshow_counter_int ->{##1}%
8205     }%
8206 }

```

```

8207     }
8208   }
8209 }
8210 
8211 \bool_if:NT \c_notesslides_notes_bool {
8212   \def\author{\@dblarg\ns@author}
8213   \long\def\ns@author[#1]#2{%
8214     \tl_if_empty:nTF{#1}{%
8215       \def\beamer@shortauthor{#2}
8216     }{%
8217       \def\beamer@shortauthor{#1}
8218     }
8219     \def\@author{#2}
8220   }
8221   \def\title{\@dblarg\ns@title}
8222   \long\def\ns@title[#1]#2{%
8223     \tl_if_empty:nTF{#1}{%
8224       \def\beamer@shorttitle{#2}
8225     }{%
8226       \def\beamer@shorttitle{#1}
8227     }
8228     \def\@title{#2}
8229     \stexdotitle{#2}
8230   }
8231   \def\insertshortauthor{
8232     \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shortauthor\beamer@shortauthor\egroup
8233   }
8234   \def\insertshorttitle{
8235     \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shorttitle\beamer@shorttitle\egroup
8236   }
8237   \stex_if_html_backend:TF{
8238     \def\insertframenumber{\stex_annotate:nn{shtml:framenumber={}}{}}
8239   }{
8240     \def\insertframenumber{\@arabic\c@framenumber}
8241   }
8242   \def\insertshortdate{\today}
8243 }

```

#### 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.

```

8244 \gdef\printexcursions{}
8245 \newcommand\excursionref[2]{% label, text
8246   \bool_if:NT \c_notesslides_notes_bool {
8247     \begin{sparagraph}[title=Excursion]
8248       #2 \sref[fallback=the appendix]{#1}.
8249     \end{sparagraph}
8250   }
8251 }
8252 \newcommand\activate@excursion[2][]{%
8253   \tl_gput_right:Nn\printexcursions{\inputref[#1]{#2}}

```

```

8254 }
8255 \newcommand\excursion[4][]{% repos, label, path, text
8256   \bool_if:NT \c_notesslides_notes_bool {
8257     \activate@excursion[#1]{#3}
8258     \excursionref{#2}{#4}
8259   }
8260 }

```

(End of definition for `\excursion`. This function is documented on page 109.)

### \excursiongroup

```

8261 \keys_define:nn{notesslides / excursiongroup }{
8262   id      .str_set:N = \l_notesslides_excursion_id_str,
8263   intro    .tl_set:N = \l_notesslides_excursion_intro_tl,
8264   archive  .str_set:N = \l_notesslides_excursion_mhrepos_str
8265 }
8266 \cs_new_protected:Nn \__notesslides_excursion_args:n {
8267   \tl_clear:N \l_notesslides_excursion_intro_tl
8268   \str_clear:N \l_notesslides_excursion_id_str
8269   \str_clear:N \l_notesslides_excursion_mhrepos_str
8270   \keys_set:nn {notesslides / excursiongroup }{ #1 }
8271 }
8272 \newcommand\excursiongroup[1][]{%
8273   \__notesslides_excursion_args:n{ #1 }
8274   \tl_if_empty:NF\printexcursions
8275   {\IfInParens{\begin{note}
8276     \begin{sfragment}{Excursions}% TODO pass on id
8277     \ifdefempty{\l_notesslides_excursion_intro_tl}{}{
8278       \exp_args:NNe \use:nn \inputref{\l_notesslides_excursion_mhrepos_str}{%
8279         \l_notesslides_excursion_intro_tl
8280       }%
8281     }%
8282     \printexcursions%
8283     \end{sfragment}
8284   \end{note}}}
8285 }
8286 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi

```

(End of definition for `\excursiongroup`. This function is documented on page 109.)

```

8287 \prop_new:N \g__notesslides_variables_prop
8288 \cs_set_protected:Npn \setSGvar #1 #2 {
8289   \prop_gput:Nnn \g__notesslides_variables_prop {#1}{#2}
8290 }
8291 \cs_set_protected:Npn \useSGvar #1 {
8292   \prop_item:Nn \g__notesslides_variables_prop {#1}
8293 }
8294 \cs_set_protected:Npn \ifSGvar #1 #2 #3 {
8295   \prop_get:NnNF \g__notesslides_variables_prop {#1} \l__notesslides_tmp {
8296     \PackageError{document-structure}{%
8297       {The sTeX Global variable #1 is undefined}%
8298       {set it with \protect\setSGvar}\TODO better error
8299     }%
8300   \tl_if_eq:NnT \l__notesslides_tmp {#2}{ #3 }
8301 }

```

```

8302
8303
8304 </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.

```

8305 <*package>
8306 @@@=problems
8307 \ProvidesExplPackage{problem}{2023/10/13}{3.4.0}{Semantic Markup for Problems}
8308 \RequirePackage{l3keys2e}
8309
8310 \keys_define:nn { problem / pkg }{
8311   notes      .default:n    = { true },
8312   notes      .bool_set:N  = \c__problems_notes_bool,
8313   gnotes     .default:n    = { true },
8314   gnotes     .bool_set:N  = \c__problems_gnotes_bool,
8315   hints      .default:n    = { true },
8316   hints      .bool_set:N  = \c__problems_hints_bool,
8317   solutions   .default:n    = { true },
8318   solutions   .bool_set:N  = \c__problems_solutions_bool,
8319   pts        .default:n    = { true },
8320   pts        .bool_set:N  = \c__problems_pts_bool,
8321   min        .default:n    = { true },
8322   min        .bool_set:N  = \c__problems_min_bool,
8323   %boxed     .default:n    = { true },
8324   %boxed     .bool_set:N  = \c__problems_boxed_bool,
8325   test       .default:n    = { true },
8326   test       .bool_set:N  = \c__problems_test_bool,
8327   unknown     .code:n      = {
8328     \PassOptionsToPackage{\CurrentOption}{stex}
8329   }
8330 }
8331 \newif\ifsolutions
8332
8333 \ProcessKeysOptions{ problem / pkg }
8334 \bool_if:NTF \c__problems_solutions_bool {
8335   \solutionstrue
8336 }{
8337   \solutionsfalse
8338 }
8339 \newif\ifintest
8340 \bool_if:NTF \c__problems_test_bool {
8341   \intesttrue
8342 }{
8343   \intestfalse
8344 }
8345 \RequirePackage{stex}

```

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

```

8347 \AddToHook{begindocument}{
8348   \ExplSyntaxOn\makeatletter
8349   \input{problem-english.ldf}
8350   \ltx@ifpackageloaded{babel}{
8351     \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bblobloaded}
8352     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8353       \input{problem-ngerman.ldf}
8354     }
8355     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8356       \input{problem-finnish.ldf}
8357     }
8358     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8359       \input{problem-french.ldf}
8360     }
8361     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8362       \input{problem-russian.ldf}
8363     }
8364   }{}}
8365   \makeatother\ExplSyntaxOff
8366 }
```

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

### 14.2.2 Problems and Solutions

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

```

8367 \bool_new:N \l_stex_key_autogradable_bool
8368 \stex_keys_define:nnnn{ problem }{
8369   \tl_set:Nn \l_stex_key_pts_tl 0
8370   \tl_set:Nn \l_stex_key_min_tl 0
8371   \str_clear:N \l_stex_key_name_str
8372   \str_clear:N \l_stex_key_mhrepos_str
8373   \bool_set_false:N \l_stex_key_autogradable_bool
8374 }{
8375   pts      .tl_set:N      = \l_stex_key_pts_tl,
8376   min      .tl_set:N      = \l_stex_key_min_tl,
8377   name     .str_set:N     = \l_stex_key_name_str,
8378   autogradable .bool_set:N = \l_stex_key_autogradable_bool,
8379   archive   .code:n = {},
8380   %archive .str_set:N     = \l_stex_key_mhrepos_str,
8381   creators  .code:n = {}
8382   %imports .tl_set:N      = \l__problems_prob_imports_tl,
8383   %refnum   .int_set:N     = \l__problems_prob_refnum_int,
8384 }{id,title,style,uses}
```

Then we set up a counter for problems.

```

\numberproblemsin
8385 \newcounter{sproblem}[section]
8386 \newcommand\numberproblemsin[1]{
8387   \addtoreset{sproblem}{#1}
```

```

8388   \def\thesproblem{\arabic{#1}.\arabic{sproblem}}
8389 }
8390 \numberproblemsin{section}
8391 %\def\theplainsproblem{\arabic{sproblem}}
8392 %\def\thesproblem{\thesection.\theplainsproblem}

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

sproblem (*env.*)

```

8393 \cs_new:Nn \__problems_activate_macros: {
8394   \stex_reactivate_macro:N \solution
8395   \stex_reactivate_macro:N \mcb
8396   \stex_reactivate_macro:N \scb
8397   \stex_reactivate_macro:N \fillinsol
8398   \stex_reactivate_macro:N \hint
8399   \stex_reactivate_macro:N \exnote
8400   \stex_reactivate_macro:N \gnote
8401 }
8402
8403 \newcounter{pts}
8404 \newcounter{min}
8405 \bool_new:N \l__problems_in_problem_bool
8406 \bool_new:N \l__problems_has_pts_bool
8407 \bool_new:N \l__problems_has_min_bool
8408 \bool_set_false:N \l__problems_in_problem_bool
8409 \stex_new_stylable_env:nnnnnn {problem} {0{}}
8410   \bool_if:NT \l__problems_in_problem_bool {
8411     \msg_error:nn{stex}{error/nestedproblem}
8412   }
8413 \cs_if_exist:NTF \l_problem_inputproblem_keys_tl {
8414   \tl_put_left:Nn \l_problem_inputproblem_keys_tl {#1,}
8415   \exp_args:Nno \stex_keys_set:nn{problem} {
8416     \l_problem_inputproblem_keys_tl
8417   }
8418 }{
8419   \stex_keys_set:nn{problem}{#1}
8420 }
8421 \refstepcounter{sproblem}
8422
8423 \stex_if_do_html:T {
8424   \str_if_empty:NT \l_stex_key_name_str {
8425     \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8426     \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8427   }
8428   \exp_args:Nne \begin{stex_annotation_env} {
8429     shtml:problem={\l_stex_key_name_str},
8430     shtml:language={ \l_stex_current_language_str},
8431     shtml:autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
8432   }
8433   \tl_if_empty:NF \l_stex_key_title_tl {
8434     \exp_args:No \stexdoctitle \l_stex_key_title_tl
8435   }
8436   \stex_annotation_invisible:nn{
8437     shtml:problempoints={\l_stex_key_pts_tl}
8438 }{ \l_stex_key_pts_tl }
```

```

8439 }
8440 \tl_set_eq:NN \thistitle \l_stex_key_title_tl
8441
8442
8443 \bool_set_true:N \l__problems_in_problem_bool
8444 \tl_set_eq:NN \l__problems_pts_tl \l_stex_key_pts_tl
8445 \tl_set_eq:NN \l__problems_min_tl \l_stex_key_min_tl
8446 \tl_if_eq:NnTF \l__problems_pts_tl {0}
8447   {\bool_set_false:N \l__problems_has_pts_bool}
8448   {\bool_set_true:N \l__problems_has_pts_bool}
8449 \tl_if_eq:NnTF \l__problems_min_tl {0}
8450   {\bool_set_false:N \l__problems_has_min_bool}
8451   {\bool_set_true:N \l__problems_has_min_bool}
8452 \int_gzero:N \g__problems_subproblem_int
8453
8454 \stex_style_apply:
8455 \stex_do_id:
8456 \__problems_activate_macros:
8457 }{
8458 \addtocounter{pts}{\l__problems_pts_tl}
8459 \addtocounter{min}{\l__problems_min_tl}
8460 \__problems_record_problem:
8461 \stex_style_apply:
8462 \stex_if_do_html:T{ \end{stex_annotation_env} }
8463 }{
8464 \par\noindent\problemheader
8465 \stex_if_do_html:F{
8466   \bool_if:NT \c__problems_pts_bool {
8467     \tl_if_eq:NnF \l__problems_pts_tl {0} {
8468       \marginpar{\l__problems_pts_tl{}~\problem@kw@pts\smallskip}
8469     }
8470   }
8471   \bool_if:NT \c__problems_min_bool {
8472     \tl_if_eq:NnF \l__problems_min_tl {0} {
8473       \marginpar{\l__problems_min_tl{}~\problem@kw@minutes\smallskip}
8474     }
8475   }
8476 }
8477 \par
8478 \stex_ignore_spaces_and_pars:
8479 }{
8480   \par\bigskip
8481 % \bool_if:NT \c__problems_test_bool \pagebreak
8482 }{s}
8483
8484 \tl_set:Nn \problemheader {
8485   \stex_if_do_html:TF{
8486     \tl_if_empty:NF \thistitle {
8487       \stex_annotation:nn{shtml:problemtitle={}}{\textbf{\thistitle}}
8488     }
8489   }{
8490     \textbf{\sproblemautorefname{}\thesproblem}
8491     \tl_if_empty:NF \thistitle {
8492       \textbf{\thistitle}

```

```

8493     }
8494   }
8495 }
8496 }
8497
8498 \cs_new_protected:Nn \__problems_record_problem: {
8499   \exp_args:NNe \iow_now:Nn \@auxout {
8500     \problem@restore {\thesproblem}{\l__problems_pts_tl}{\l__problems_min_tl}
8501   }
8502 }
8503
8504 \cs_new_protected:Npn \problem@restore #1 #2 #3 {}

subproblem (env.)
8505 \int_new:N \g__problems_subproblem_int
8506
8507 \stex_new_stylable_env:nnnnnn {subproblem} {0{}}
8508   \stex_keys_set:nn{problem}{#1}
8509   \bool_if:NF \l__problems_in_problem_bool{
8510     \ifstexhtml\else
8511       \par{\bfseries WARNING~subproblem~to~be~used~in~some~problem}\par
8512     \fi
8513   \__problems_activate_macros:
8514   \bool_set_true:N \l__problems_in_problem_bool
8515   \tl_set:Nn \l__problems_pts_tl{0}
8516   \tl_set:Nn \l__problems_min_tl{0}
8517 }
8518 \str_if_empty:NT \l_stex_key_name_str {
8519   \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8520   \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8521 }
8522 \stex_if_do_html:T {
8523   \str_if_empty:NT \l_stex_key_name_str {
8524     \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8525     \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8526   }
8527   \exp_args:Nne \begin{stex_annotation_env} {
8528     shtml:subproblem={\l_stex_key_name_str},
8529     shtml:language={\l_stex_current_language_str},
8530     shtml:autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
8531   }
8532   \stex_annotation_invisible:n{}
8533   \tl_if_empty:NF \l_stex_key_title_tl {
8534     \exp_args:No \stexdoctitle \l_stex_key_title_tl
8535   }
8536 }
8537 \int_gincr:N \g__problems_subproblem_int
8538 \bool_if:NF \l__problems_has_pts_bool {
8539   \tl_gset:Nx \l__problems_pts_tl {\int_eval:n {\l__problems_pts_tl + \l_stex_key_pts_tl}}
8540 }
8541 \bool_if:NF \l__problems_has_min_bool {
8542   \tl_gset:Nx \l__problems_min_tl {\int_eval:n {\l__problems_min_tl + \l_stex_key_min_tl}}
8543 }
8544 \stex_if_smsmode:F \stex_style_apply:

```

```

8545 }{
8546   \stex_if_smsmode:F \stex_style_apply:
8547   \stex_if_do_html:T{ \end{stex_annotation_env} }
8548 }{
8549   \begin{list}{}{
8550     \setlength\topsep{0pt}
8551     \setlength\parsep{0pt}
8552     \setlength\rightmargin{0pt}
8553   }\item[\int_use:N \g__problems_subproblem_int .]
8554   \bool_if:NT \c__problems_pts_bool {
8555     \bool_if:NF \l__problems_has_pts_bool {
8556       \marginpar{\smallskip\l_stex_key_pts_tl{}~\problem@kw@pts}
8557     }
8558   }
8559   \bool_if:NT \c__problems_min_bool {
8560     \bool_if:NF \l__problems_has_min_bool{
8561       \marginpar{\smallskip\l_stex_key_min_tl{}~\problem@kw@minutes}
8562     }
8563   }
8564 }{
8565   \end{list}
8566 }{}}

\includeproblem
8567 \stex_keys_define:nnnn{ includeproblem }{
8568   \str_clear:N \l_stex_key_mhrepos_str
8569 }{
8570   archive .str_set:N      = \l_stex_key_mhrepos_str,
8571   unknown .code:n = {}
8572 }{}

8573 \NewDocumentCommand\includeproblem{O{} m}{
8574   \group_begin:
8575   \tl_set:Nn \l_problem_inputproblem_keys_tl {#1}
8576   \stex_keys_set:nn{includeproblem}{#1}
8577   \exp_args:Nno \use:nn{\inputref[]}\l_stex_key_mhrepos_str{#2}
8578   \group_end:
8579 }

8580 }
8581

```

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

```

solution (env.)
8582 \int_new:N \g_problem_id_counter
8583 \dim_new:N \l_stex_key_testsphere_dim
8584 \stex_keys_define:nnnn{ solution }{
8585   \str_clear:N \l_stex_key_answerclass_str
8586   \dim_zero:N \l_stex_key_testsphere_dim
8587 }{
8588   testsphere .dim_set:N = \l_stex_key_testsphere_dim,
8589   answerclass .str_set:N = \l_stex_key_answerclass_str
8590 }{id,title,style}
8591
8592 \cs_new_protected:Nn \__problems_solution_start:n {

```

```

8593 \stex_keys_set:nn{ solution }{#1}
8594 \str_if_empty:NT \l_stex_key_id_str {
8595   \int_gincr:N \g_problem_id_counter
8596   \str_set:Nx \l_stex_key_id_str {
8597     SOLUTION_\int_use:N \g_problem_id_counter
8598   }
8599 }
8600 \stex_if_do_html:T{
8601   \begin{stex_annotation_env}[
8602     shtml:solution=\l_stex_key_id_str,
8603     shtml:answerclass={\l_stex_key_answerclass_str}
8604   ]
8605 }
8606 \stex_style_apply:
8607 }
8608
8609 \stex_new_stylable_env:nnnnnnn { solution }{ 0{} }{
8610   \stex_if_do_html:TF{
8611     \__problems_solution_start:n{#1}
8612   }{
8613     \ifsolutions
8614       \__problems_solution_start:n{#1}
8615     \else
8616       \stex_keys_set:nn{ solution }{#1}
8617       \testspace{\l_stex_key_testspace_dim}
8618       \setbox\l_tmpa_box\vbox\bgroup
8619     \fi
8620   }
8621 }{
8622   \stex_if_do_html:TF{
8623     \stex_style_apply:
8624     \end{stex_annotation_env}
8625   }{
8626     \ifsolutions
8627       \stex_style_apply:
8628       \stex_if_do_html:T{
8629         \end{stex_annotation_env}
8630       }
8631     \else
8632       \egroup
8633     \fi
8634   }
8635 }{
8636   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8637   \noindent\emph{\problem@kw@solution\tl_if_empty:NF \l_stex_key_title_tl{%
8638     {-}\l_stex_key_title_tl \str_if_empty:NF \l_stex_key_answerclass_str {%
8639       (Answer Class: \l_stex_key_answerclass_str)
8640     }
8641   } :~}
8642 }{
8643   \par\rule[.3em]{\linewidth}{0.4pt}\newline
8644 }{}}
8645
8646 \stex_deactivate_macro:Nn \solution {sproblem~environments}

```

```

\startsolution
\stopsolution
8647 \cs_new_protected:Npn \startsolution{
8648     \global\solutionstrue
8649 }
8650 \cs_new_protected:Npn \stopsolution{
8651     \global\solutionsfalse
8652 }

(End of definition for \startsolution and \stopsolution. These functions are documented on page
111.)

```

hint (*env.*)

```

8653
8654 \stex_keys_define:nnnn{ problemenv }{}{}{id,title,style}
8655
8656 \cs_new_protected:Nn \__problems_hint_start:n {
8657     \stex_keys_set:nn{ problemenv }{#1}
8658     \str_if_empty:NT \l_stex_key_id_str {
8659         \int_gincr:N \g_problem_id_counter
8660         \str_set:Nx \l_stex_key_id_str {
8661             HINT_\int_use:N \g_problem_id_counter
8662         }
8663     }
8664     \stex_if_do_html:T{
8665         \begin{stex_annotation_env}[
8666             shtml:problemhint=\l_stex_key_id_str
8667         ]
8668     }
8669     \stex_style_apply:
8670 }
8671
8672 \stex_new_stylable_env:nnnnnnn { hint }{ 0{} }{
8673     \stex_if_do_html:TF{
8674         \__problems_hint_start:n{#1}
8675     }{
8676         \bool_if:NTF \c__problems_hints_bool {
8677             \__problems_hint_start:n{#1}
8678         }{
8679             \setbox\l_tmpa_box\vbox\bgroup
8680         }
8681     }
8682 }{
8683     \stex_if_do_html:TF{
8684         \stex_style_apply:
8685         \end{stex_annotation_env}
8686     }{
8687         \bool_if:NTF \c__problems_hints_bool {
8688             \stex_style_apply:
8689             \stex_if_do_html:T{
8690                 \end{stex_annotation_env}
8691             }
8692         }{
8693             \egroup
8694         }

```

```

8695   }
8696 }{
8697   \par\smallskip\rule[.3em]{\ linewidth}{0.4pt}\newline\smallskip
8698   \noindent\emph{\problem@kw@hint\tl_if_empty:NF \l_stex_key_title_tl{%
8699     {~}\l_stex_key_title_tl
8700   } :~}
8701 }{
8702   \par\rule[.3em]{\ linewidth}{0.4pt}\newline
8703 }{}
8704 \stex_deactivate_macro:Nn \hint {sproblem-environments}

exnote (env.)
8705 \cs_new_protected:Nn \__problems_exnote_start:n {
8706   \stex_keys_set:nn{ problemenv }{\#1}
8707   \str_if_empty:NT \l_stex_key_id_str {
8708     \int_gincr:N \g_problem_id_counter
8709     \str_set:Nx \l_stex_key_id_str {
8710       EXNOTE_\int_use:N \g_problem_id_counter
8711     }
8712   }
8713   \stex_if_do_html:T{
8714     \begin{stex_annotation_env}{
8715       shtml:problemnote=\l_stex_key_id_str
8716     }
8717   }
8718   \stex_style_apply:
8719 }
8720
8721 \stex_new_stylable_env:nnnnnnn { exnote }{ 0{} }{
8722   \stex_if_do_html:TF{
8723     \__problems_exnote_start:n{\#1}
8724   }{
8725     \bool_if:NTF \c__problems_notes_bool {
8726       \__problems_exnote_start:n{\#1}
8727     }{
8728       \setbox\l_tmpa_box\vbox\bgroup
8729     }
8730   }
8731 }{
8732   \stex_if_do_html:TF{
8733     \stex_style_apply:
8734     \end{stex_annotation_env}
8735   }{
8736     \bool_if:NTF \c__problems_notes_bool {
8737       \stex_style_apply:
8738       \stex_if_do_html:T{
8739         \end{stex_annotation_env}
8740       }
8741     }{
8742       \egroup
8743     }
8744   }
8745 }{
8746   \par\smallskip\rule[.3em]{\ linewidth}{0.4pt}\newline\smallskip

```

```

8747   \noindent\emph{\problem@kw@note\tl_if_empty:N \l_stex_key_title_tl{
8748     {~}\l_stex_key_title_tl
8749   } :~}
8750 }{
8751   \par\rule[.3em]{\linewidth}{0.4pt}\newline
8752 }{}}
8753 \stex_deactivate_macro:Nn \exnote {sproblem~environments}

gnote (env.)
8754 \int_new:N \l__problems_anscls_int
8755
8756 \cs_new_protected:Nn \__problems_gnote_start:n {
8757   \stex_keys_set:nn{ problemenv }{#1}
8758   \str_if_empty:NT \l_stex_key_id_str {
8759     \int_gincr:N \g_problem_id_counter
8760     \str_set:Nx \l_stex_key_id_str {
8761       GNOTE_ \int_use:N \g_problem_id_counter
8762     }
8763   }
8764
8765 \stex_if_do_html:T{
8766   \begin{stex_annotation_env} {
8767     shtml:problemgnote=\l_stex_key_id_str
8768   }
8769 }
8770 \stex_style_apply:
8771 }
8772
8773 \stex_new_stylable_env:nnnnnnn { gnote }{ 0{} }{
8774   \stex_if_do_html:TF{
8775     \__problems_gnote_start:n{#1}
8776   }{
8777     \bool_if:NTF \c__problems_gnotes_bool {
8778       \__problems_gnote_start:n{#1}
8779     }{
8780       \setbox\l_tmpa_box\vbox\bgroup
8781     }
8782   }
8783 \stex_reactivate_macro:N \anscls
8784 }{
8785   \stex_if_do_html:TF{
8786     \stex_style_apply:
8787     \end{stex_annotation_env}
8788   }{
8789     \bool_if:NTF \c__problems_gnotes_bool {
8790       \stex_style_apply:
8791       \stex_if_do_html:T{
8792         \end{stex_annotation_env}
8793       }
8794     }{
8795       \egroup
8796     }
8797   }{
8798 }

```

```

8799 \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8800 \noindent\emph{\problem@kw@grading\str_if_empty:NF \l_stex_key_title_tl{
8801   {} \l_stex_key_title_tl
8802   } :~}
8803 }{
8804   \par\rule[.3em]{\linewidth}{0.4pt}\newline
8805 }{}
8806 \stex_deactivate_macro:Nn \gnote {sproblem~environments}
8807
8808
8809 \stex_keys_define:nnnn{ anscls }{
8810   \str_clear:N \l_stex_key_pts_str
8811   \tl_clear:N \l_stex_key_feedback_tl
8812 }{
8813   pts .str_set:N = \l_stex_key_pts_str,
8814   feedback .tl_set:N = \l_stex_key_feedback_str,
8815   update .code:n = {}
8816 }{id}
8817 \newcommand \anscls [2] [] {
8818   \stex_keys_set:nn{ anscls }{#1}
8819   \str_if_empty:NT \l_stex_key_id_str {
8820     \int_incr:N \l__problems_anscls_int
8821     \str_set:Nx \l_stex_key_id_str {
8822       AC\int_use:N \l__problems_anscls_int
8823     }
8824   }
8825   \begin{list}{}{
8826     \setlength\topsep{0pt}
8827     \setlength\parsep{0pt}
8828     \setlength\rightmargin{0pt}
8829   }\item[\l_stex_key_id_str]
8830   \stex_if_do_html:TF{
8831     \stex_annotation:nn{
8832       shtml:answerclass={\l_stex_key_id_str}
8833       \str_if_empty:NF \l_stex_key_pts_str{
8834         ,shtml:answerclass-pts={\l_stex_key_pts_str}
8835       }
8836       \str_if_empty:NF \l_stex_key_feedback_str{
8837         ,shtml:answerclass-feedback={\l_stex_key_feedback_str}
8838       }
8839     }{
8840       #2
8841     }
8842   }{#2}
8843   \str_if_empty:NF \l_stex_key_pts_str {\par
8844     ~ \problem@kw@points:~\l_stex_key_pts_str
8845   }
8846   \str_if_empty:NF \l_stex_key_feedback_str {\par
8847     ~ \problem@kw@feedback :~\l_stex_key_feedback_str
8848   }
8849   \end{list}
8850 }
8851 \stex_deactivate_macro:Nn \anscls {gnote~environments}

```

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

```

8852 \def\pts#1{
8853   \bool_if:NT \c__problems_pts_bool {
8854     \stex_annotate:nn{shtml:problempoints={#1}}{\marginpar{\#1~\problem@kw@pts}}
8855   }\hbox_unpack:N\c_empty_box
8856 }
8857 \def\min#1{
8858   \bool_if:NT \c__problems_min_bool {
8859     \stex_annotate:nn{shtml:problemmintes={}}{\marginpar{\#1~\problem@kw@minutes}}
8860   }\hbox_unpack:N\c_empty_box
8861 }

mcb (env.)
8862 \stex_new_stylable_env:nnnnnnn{mcb}{0}{}{
8863   \stex_keys_set:nn{style}{#1}
8864   \stex_if_do_html:T{
8865     \tl_set:Nn\problem_mcc_box_tl{}
8866     \exp_args:Nne \begin{stex_annotate_env}{}{
8867       shtml:multiple-choice-block={}
8868       \clist_if_empty:NF \l_stex_key_style_clist ,{
8869         shtml:styles={\l_stex_key_style_clist}
8870       }
8871     }
8872   }
8873   \stex_deactivate_macro:Nn \mcb {sproblem-environments}
8874   \stex_deactivate_macro:Nn \scb {sproblem-environments}
8875   \stex_deactivate_macro:Nn \solution {sproblem-environments}
8876   \stex_deactivate_macro:Nn \hint {sproblem-environments}
8877   \stex_deactivate_macro:Nn \exnote {sproblem-environments}
8878   \stex_deactivate_macro:Nn \gnote {sproblem-environments}
8879   \stex_reactivate_macro:N \mcc
8880   \cs_set:Nn \__problems_mccline:n{
8881     \begin{list}{}{
8882       \setlength\topsep{0pt}
8883       \setlength\parsep{0pt}
8884       \setlength\rightmargin{0pt}
8885     }\item[\problem_mcc_box_tl] ##1 \end{list}
8886   }
8887   \stex_style_apply:
8888 }
8889 \stex_style_apply:
8890 \stex_if_do_html:T{
8891   \end{stex_annotate_env}
8892 }
8893 }{\par}{}}
8894 \stexstylemcb[inline]{
8895   \cs_set:Nn \__problems_mccline:n{\problem_mcc_box_tl{\~} #1}
8896 }{}

8897 \stex_deactivate_macro:Nn \mcb {sproblem-environments}
we define the keys for the mcc macro
8898 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8899   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
```

```

8901     \bool_set_true:N #1
8902   }{
8903     \bool_set_false:N #1
8904   }
8905 }
8906 \stex_keys_define:nnnn{mcc}{
8907   \tl_clear:N \l_stex_key_feedback_tl
8908   \bool_set_false:N \l_stex_key_T_bool
8909   \tl_clear:N \l_stex_key_Ttext_tl
8910   \tl_clear:N \l_stex_key_Ftext_tl
8911 }{
8912   feedback .tl_set:N      = \l_stex_key_feedback_tl ,
8913   T       .code:n      = {\bool_set_true:N \l_stex_key_T_bool} ,
8914   F       .code:n      = {\bool_set_false:N \l_stex_key_T_bool} ,
8915   Ttext   .tl_set:N      = \l_stex_key_Ttext_tl ,
8916   Ftext   .tl_set:N      = \l_stex_key_Ftext_tl ,
8917 }{id}
8918

\mcc

8919 \tl_set:Nn \problem_mcc_box_tl {
8920   \ltx@ifpackageloaded{amssymb}{$\square$}){
8921     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
8922   }
8923 }
8924 \newcommand\mcc[2][]{%
8925   \stex_keys_set:nn{mcc}{#1}
8926   \tl_set:Nn \l_tmpa_tl{%
8927     \bool_if:NTF \l_stex_key_T_bool {
8928       \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
8929     }{
8930       \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
8931     }
8932     \tl_if_empty:NF \l_stex_key_feedback_tl {
8933       \\\emph{\l_stex_key_feedback_tl}
8934     }
8935   }
8936
8937 \__problems_mccline:n{
8938   \stex_if_do_html:TF{
8939     \stex_annotation:nn{shtml:mcc=}
8940       \bool_if:NTF \l_stex_key_T_bool {true}{false}
8941     }){
8942       #2\stex_annotation:nn{shtml:mcc-solution={}}{\l_tmpa_tl}
8943     }
8944   }{
8945     #2\if\if\footnote{\l_tmpa_tl}\fi
8946   }
8947 }
8948 }
8949 \stex_deactivate_macro:Nn \mcc {mcb~environments}

```

(End of definition for `\mcc`. This function is documented on page 112.)

```

scb (env.)

8950
8951 \tl_set:Nn\problem_scc_box_tl{$\bigcirc$}
8952
8953 \cs_new_protected:Nn \__problems_maybe_inline:n {
8954     \clist_if_in:NnTF \l_stex_key_style_clist {inline} {
8955         \let\__problems_oldpar:\stex_par:
8956         \cs_set:Nn\stex_par:{}}
8957         #1
8958         \let\stex_par:\__problems_oldpar:
8959     }{#1}
8960 }
8961
8962 \stex_new_stylable_env:nnnnnn{scb}{0}{}
8963     \stex_keys_set:nn{style}{#1}
8964     \stex_if_do_html:T{
8965         \__problems_maybe_inline:n{
8966             \exp_args:Nne\begin{stex_annotation_env}{}{
8967                 shtml:single-choice-block={}
8968                 \clist_if_empty:NF \l_stex_key_style_clist ,%
8969                     shtml:styles={\l_stex_key_style_clist}
8970             }
8971         }
8972     }
8973     \tl_set:Nn\problem_scc_box_tl{}
8974 }
8975 \stex_deactivate_macro:Nn \mcb {sproblem-environments}
8976 \stex_deactivate_macro:Nn \scb {sproblem-environments}
8977 \stex_deactivate_macro:Nn \solution {sproblem-environments}
8978 \stex_deactivate_macro:Nn \hint {sproblem-environments}
8979 \stex_deactivate_macro:Nn \exnote {sproblem-environments}
8980 \stex_deactivate_macro:Nn \gnote {sproblem-environments}
8981 \stex_reactivate_macro:N \scc
8982 \cs_set:Nn \__problems_sccline:n{
8983     \begin{list}{}{
8984         \setlength\topsep{0pt}
8985         \setlength\parsep{0pt}
8986         \setlength\rightmargin{0pt}
8987         } \item[\problem_scc_box_tl] ##1 \end{list}
8988 }
8989 \stex_style_apply:
8990 }{
8991     \stex_style_apply:
8992     \stex_if_do_html:T{
8993         \__problems_maybe_inline:n { \end{stex_annotation_env} }
8994     }
8995 }{\par}{}{%
8996 \stexstylescb[inline]{%
8997     \cs_set:Nn \__problems_sccline:n{\problem_scc_box_tl{~} #1}
8998 }{}%
8999 \stex_deactivate_macro:Nn \scb {sproblem-environments}

```

\scc

```

9001 \newcommand\scc[2][]{%
9002   \stex_keys_set:nn{mcc}{#1}%
9003   \tl_set:Nn \l_tmpa_tl{%
9004     \bool_if:NTF \l_stex_key_T_bool {%
9005       \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
9006     }{%
9007       \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
9008     }%
9009     \tl_if_empty:NF \l_stex_key_feedback_tl {%
9010       \\\emph{\l_stex_key_feedback_tl}
9011     }%
9012   }%
9013 }%
9014 %
9015 \__problems_sccline:n{%
9016   \stex_if_do_html:TF{%
9017     \stex_annotate:nn{shtml:scc={%
9018       \bool_if:NTF \l_stex_key_T_bool {true}{false}
9019     }}{%
9020       #2\stex_annotate:nn{shtml:scc-solution={}}{\l_tmpa_tl}
9021     }%
9022   }{%
9023     #2\ifsolutions\footnote{\l_tmpa_tl}\fi
9024   }%
9025 }%
9026 }%
9027 \stex_deactivate_macro:Nn \scc {scb-environments}
9028 %
9029 %
9030 \newcommand\yesTnoF{%
9031   \begin{scb}[style=inline]
9032     \scc[T]{yes}~\scc[F]{no}
9033   \end{scb}
9034 }%
9035 \newcommand\yesFnoT{%
9036   \begin{scb}[style=inline]
9037     \scc[F]{yes}~\scc[T]{no}
9038   \end{scb}
9039 }%
9040 \newcommand\trueTfalseF{%
9041   \begin{scb}[style=inline]
9042     \scc[T]{true}~\scc[F]{false}
9043   \end{scb}
9044 }%
9045 \newcommand\trueFfalseT{%
9046   \begin{scb}[style=inline]
9047     \scc[F]{true}~\scc[T]{false}
9048   \end{scb}
9049 }

```

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

\fillinsol

```

9050 \stex_keys_define:nnnn{fillinsol}{%

```

```

9051   \tl_clear:N \l__problems_fillin_solution_tl
9052   \dim_zero:N \l_stex_key_testspace_dim
9053 }{
9054   testspace .dim_set:N = \l_stex_key_testspace_dim,
9055   exact .code:n = {\__problems_parse_fillin_arg:nnnn{exact}#1 },
9056   numrange .code:n = {\__problems_parse_fillin_arg:nnnn{numrange}#1 },
9057   regex .code:n = {\__problems_parse_fillin_arg:nnnn{regex}#1 }
9058 }{}
9059
9060 \stex_if_html_backend:TF{
9061   \cs_new:Nn \__problems_parse_fillin_arg:nnnn {
9062     \tl_set:Nn \l_tmpa_tl{#3}
9063     \tl_set:Nn \l_tmpb_tl{T}
9064     \stex_annotation_invisible:nn{
9065       shtml:fillin-case={#1},
9066       shtml:fillin-case-value={#2},
9067       shtml:fillin-case-verdict={
9068         \tl_if_eq:NNTF\l_tmpa_tl\l_tmpb_tl{true}{false}
9069       },
9070     }{#4}
9071   }
9072 }{
9073   \cs_new:Nn \__problems_parse_fillin_arg:nnnn {
9074     \tl_put_right:Nn \l__problems_fillin_solution_tl {
9075       #1 & #2 & #3 & #4 \crcr
9076     }
9077   }
9078 }
9079
9080
9081 \newcommand\fillinsol[2][]{%
9082   \stex_if_do_html:F \quad
9083   \mode_if_math:TF{
9084     \hbox{\__problems_fillinsol:nn{#1}{$\,$#2$\,$}}
9085   }{
9086     \__problems_fillinsol:nn{#1}{#2}
9087   }
9088   \stex_if_do_html:F \quad
9089 }
9090
9091 \stex_if_html_backend:TF{
9092   \cs_new_protected:Nn \__problems_fillinsol:nn {
9093     \exp_args:Ne \stex_annotation:nn{shtml:fillinsol={true}}{
9094       \stex_keys_set:nn{fillinsol}{#1}
9095       \stex_annotation_force_break:n{
9096         #2
9097       }
9098       \l__problems_fillin_solution_tl
9099     }
9100   }
9101 }{
9102   \cs_new_protected:Nn \__problems_fillinsol:nn {
9103     \stex_keys_set:nn{fillinsol}{#1}
9104     \ifsolutions

```

```

9105     \textcolor{red}{\fbox{\#2}}
9106     \tl_if_empty:NF \l__problems_fillin_solution_tl {
9107         \footnote{
9108             \halign{ ~##~\hfil & ~##~\hfil & ~##~\hfil & ~##~\hfil \cr
9109                 \textbf{type} & \textbf{case} & \textbf{verdict} & \textbf{feedback} \cr
9110             \l__problems_fillin_solution_tl
9111         }
9112     }
9113 }
9114 \else
9115     \fbox{\dim_compare:nNnTF\l_stex_key_testspace_dim={0pt}{
9116         \phantom{\huge{\#2}}
9117     }{
9118         \hspace{\l_stex_key_testspace_dim}
9119     }}
9120 \fi
9121 }
9122 }
9123
9124 \stex_deactivate_macro:Nn \fillinsol {sproblem~environments}

```

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

### `\testemptypage`

```

9125 \newcommand\testemptypage[1][]{%
9126     \bool_if:NT \c__problems_test_bool {\vfill\begin{center}\hwexam@kw@testemptypage\end{center}}
9127 }

```

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

### `\testspace`

```

9128 \newcommand\testspace[1]{\bool_if:NT \c__problems_test_bool {\vspace*{#1}}}
9129 \newcommand\testsmallspace{\testspace{1cm}}
9130 \newcommand\testmedspace{\testspace{2cm}}
9131 \newcommand\testbigspace{\testspace{3cm}}

```

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

### `\testnewpage`

```

9132 \newcommand\testnewpage{\bool_if:NT \c__problems_test_bool {\newpage}}

```

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

```

9133 
```

## 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.

```

9134 <*package>
9135 \ProvidesExplPackage{hwexam}{2023/10/13}{3.4.0}{homework assignments and exams}

```

```

9136 \RequirePackage{l3keys2e}
9137
9138 \keys_define:nn {hwexam / pkg}{
9139   multiple .default:n = { false },
9140   multiple .bool_set:N = \c_hwexam_multiple_bool,
9141   unknown .code:n = {
9142     \PassOptionsToPackage{\CurrentOption}{problem}
9143   }
9144 }
9145 \ProcessKeysOptions{ hwexam /pkg }
9146 \RequirePackage{problem}

```

- \hwexam\_kw\_\* For multilinguality, we define internal macros for keywords that can be specialized in \*.ldf files.

```

9147 \AddToHook{begindocument}{
9148   \ExplSyntaxOn\makeatletter
9149   \input{hwexam-english.ldf}
9150   \ltx@ifpackageloaded{babel}{
9151     \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bblobloaded}
9152     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
9153       \input{hwexam-ngerman.ldf}
9154     }
9155     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
9156       \input{hwexam-finnish.ldf}
9157     }
9158     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
9159       \input{hwexam-french.ldf}
9160     }
9161     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
9162       \input{hwexam-russian.ldf}
9163     }
9164   }{}
9165   \makeatother\ExplSyntaxOff
9166 }

```

(End of definition for \hwexam\_kw\_\*. This function is documented on page ??.)

### 14.3.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```

assignment (env.)
9167 \stex_keys_define:nnnn{ assignment }{
9168   \tl_clear:N \l_stex_key_number_tl
9169   \tl_clear:N \l_stex_key_given_tl
9170   \tl_clear:N \l_stex_key_due_tl
9171 }{
9172   number .tl_set:N = \l_stex_key_number_tl,
9173   given .tl_set:N = \l_stex_key_given_tl,
9174   due .tl_set:N = \l_stex_key_due_tl,
9175   unknown .code:n = {}
9176 }{id,title,style}

```

```

9177 \newcounter{assignment}
9178 \stex_new_stylable_env:nnnnnnn {assignment}{0{}}
9179   \cs_if_exist:NTF \l_hwexam_includeassignment_keys_tl {
9180     \tl_put_left:Nn \l_hwexam_includeassignment_keys_tl {\#1,}
9181     \exp_args:Nno \stex_keys_set:nn{assignment}{
9182       \l_hwexam_includeassignment_keys_tl
9183     }
9184   }{
9185     \stex_keys_set:nn{assignment}{\#1}
9186   }
9187   \tl_if_empty:NF \l_stex_key_number_tl {
9188     \global\setcounter{assignment}{\int_eval:n{\l_stex_key_number_tl-1}}
9189   }
9190   \global\refstepcounter{assignment}
9191   \setcounter{sproblem}{0}
9192   \def\thesproblem{\theassignment.\arabic{sproblem}}
9193   \stex_style_apply:
9194   \stex_do_id:
9195   }{
9196   }{
9197   \stex_style_apply:
9198   }{
9199   \par\begin{center}
9200   \textbf{\Large\assignmentautorefname~\theassignment}
9201   \tl_if_empty:NF \l_stex_key_title_tl {
9202     \textbf{\l_stex_key_title_tl}
9203   }
9204   }\par\smallskip
9205   \textbf{
9206     \tl_if_empty:NF \l_stex_key_given_tl {
9207       \hwexam@kw@given :~\l_stex_key_given_tl\quad
9208     }
9209     \tl_if_empty:NF \l_stex_key_due_tl {
9210       \hwexam@kw@due :~\l_stex_key_due_tl\quad
9211     }
9212   }
9213   \end{center}
9214   \par\bigskip
9215   }{
9216   \par\pagebreak
9217 }{}
```

\includeassignment

```

9218 \NewDocumentCommand\includeassignment{0{} m} {
9219   \group_begin:
9220   \tl_set:Nn \l_hwexam_includeassignment_keys_tl {\#1}
9221   \stex_keys_set:nn{includeproblem}{\#1}
9222   \exp_args:Nno \use:nn{\inputref[]}\l_stex_key_mhrepos_str{\#2}
9223   \group_end:
9224 }
```

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

Restoring information about problems:

```

9225 \prop_new:N \c_@@_problems_prop
```

```

9226 \tl_set:Nn \c_@@_total_mins_tl {0}
9227 \tl_set:Nn \c_@@_total_pts_tl {0}
9228 \int_new:N \c_@@_total_problems_int
9229 \cs_set_protected:Npn \problem@restore #1 #2 #3 {
9230   \int_gincr:N \c_@@_total_problems_int
9231   \prop_gput:Nnn \c_@@_problems_prop {#1}{#2}{#3}
9232   \tl_gset:Nx \c_@@_total_pts_tl { \int_eval:n { \c_@@_total_pts_tl + #2 } }
9233   \tl_gset:Nx \c_@@_total_mins_tl { \int_eval:n { \c_@@_total_mins_tl + #2 } }
9234 }

```

\correction@table This macro generates the correction table

```

9235 \newcommand\correction@table{
9236   \int_compare:nNnT \c_@@_total_problems_int = 0 {
9237     \int_incr:N \c_@@_total_problems_int
9238     \prop_put:Nnn \c_@@_problems_prop {-}{-}{-}
9239   }
9240   \tl_clear:N \l_tmpa_tl
9241   \tl_clear:N \l_tmpb_tl
9242   \tl_clear:N \l_tmpc_tl
9243   \prop_map_inline:Nn \c_@@_problems_prop {
9244     \tl_put_right:Nn \l_tmpa_tl { ##1 & }
9245     \tl_put_right:Nx \l_tmpb_tl { \use_i:nn ##2 & }
9246     \tl_put_right:Nn \l_tmpc_tl { & }
9247   }
9248   \resizebox{\textwidth}{!}{%
9249   \exp_args:Nne \begin{tabular}{|l|*{\int_use:N \c_@@_total_problems_int}{c}||l|}\hline
9250   &\exp_args:Ne \multicolumn{\int_eval:n{ \c_@@_total_problems_int + 1}}{c||}
9251   {\footnotesize\hwexam@kw@forggrading} &\hline
9252   \hwexam@kw@probs & \l_tmpa_tl \hwexam@kw@sum & \hwexam@kw@grade\\ \hline
9253   \hwexam@kw@pts & \l_tmpb_tl \c_@@_total_pts_tl & \\ \hline
9254   \hwexam@kw@reached & \l_tmpc_tl & \\ [.7cm]\hline
9255   \end{tabular}}}

```

(End of definition for \correction@table. This function is documented on page ??.)

\testheading

```

9256 \def\hwexamheader{\input{hwexam-default.header}}
9257
9258 \def\hwexamminutes{
9259   \tl_if_empty:NTF \hwexam@duration {
9260     {\hwexam@min}~\hwexam@minutes@kw
9261   }{
9262     \hwexam@duration
9263   }
9264 }
9265
9266 \stex_keys_define:nnnn{ hwexam / testheading }{
9267   \tl_clear:N \hwexam@min
9268   \tl_clear:N \hwexam@duration
9269   \tl_clear:N \hwexam@reqpts
9270   \tl_clear:N \hwexam@tools
9271 }{
9272   min .tl_set:N = \hwexam@min,
9273   duration .tl_set:N = \hwexam@duration,

```

```

9274   reqpts .tl_set:N  = \hwexam@reqpts,
9275   tools  .tl_set:N  = \hwexam@tools
9276 }{ }
9277
9278 \newenvironment{testheading}[1][]{%
9279   \stex_keys_set:nn { hwexam / testheading}{#1}
9280
9281   \tl_set_eq:NN \hwexam@totalpts \c_@@_total_pts_tl
9282   \tl_set_eq:NN \hwexam@totalmin \c_@@_total_mins_tl
9283   \tl_set:Nx \hwexam@checktime {\int_eval:n { \hwexam@min - \hwexam@totalmin }}
9284
9285   \newif\if@bonuspoints
9286   \tl_if_empty:NTF \hwexam@reqpts {
9287     \@bonuspointsfalse
9288   }{
9289     \tl_set:Nx \hwexam@bonuspts {
9290       \int_eval:n{\hwexam@totalpts - \hwexam@reqpts}
9291     }
9292     \@bonuspointstrue
9293   }
9294
9295   \makeatletter\hwexamheader\makeatother
9296 }{
9297   \newpage
9298 }

```

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

9299 ⟨/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

9300 ⟨@@=tikzinput⟩

```

9301 <*package>
9302
9303 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% tikzinput.dtx %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9304
9305 \ProvidesExplPackage{tikzinput}{2023/10/13}{3.4.0}{tikzinput package}
9306 \RequirePackage{l3keys2e}
9307
9308 \keys_define:nn { tikzinput } {
9309   image .bool_set:N = \c_tikzinput_image_bool,
9310   image .default:n = false ,
9311   unknown .code:n = {}
9312 }
9313
9314 \ProcessKeysOptions { tikzinput }
9315
9316 \bool_if:NTF \c_tikzinput_image_bool {
9317   \RequirePackage{graphicx}
9318
9319   \providecommand\usetikzlibrary[]{}
9320   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
9321 }
9322 \RequirePackage{tikz}
9323 \RequirePackage{standalone}
9324
9325 \newcommand\tikzinput [2] [] {
9326   \setkeys{Gin}{#1}
9327   \ifx \Gin@ewidth \Gin@exclamation
9328     \ifx \Gin@eheight \Gin@exclamation
9329       \input { #2 }
9330     \else
9331       \resizebox{!}{\Gin@eheight }{
9332         \input { #2 }
9333       }
9334     \fi
9335   \else
9336     \ifx \Gin@eheight \Gin@exclamation
9337       \resizebox{ \Gin@ewidth }{! }{
9338         \input { #2 }
9339       }
9340     \else
9341       \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
9342         \input { #2 }
9343       }
9344     \fi
9345   \fi
9346 }
9347 }
9348
9349 \newcommand\ctikzinput [2] [] {
9350   \begin{center}
9351     \tikzinput [#1] {#2}
9352   \end{center}
9353 }

```

9354 </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                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                           |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| \\$ .....                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 2085, 2088, 2092                                                          |
| \; .....                                                                                                                                                                                                                                                                                                                                                                                                                                                              | <u>7510</u> , 7541, 7554, 7556, 7569                                      |
| @@ commands:                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                           |
| \c_@@_problems_prop .....                                                                                                                                                                                                                                                                                                                                                                                                                                             | 9225, 9231, 9238, 9243                                                    |
| \c_@@_total_mins_t1 ..                                                                                                                                                                                                                                                                                                                                                                                                                                                | 9226, 9233, 9282                                                          |
| \c_@@_total_problems_int .....                                                                                                                                                                                                                                                                                                                                                                                                                                        | 9228, 9230, 9236, 9237, 9249, 9250                                        |
| \c_@@_total_pts_t1 .....                                                                                                                                                                                                                                                                                                                                                                                                                                              | 9227, 9232, 9253, 9281                                                    |
| \\ 53, 77, 1055, 7841, 7846, 8232, 8235,<br>8933, 9011, 9251, 9252, 9253, 9254                                                                                                                                                                                                                                                                                                                                                                                        |                                                                           |
| \{ .....                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 7559                                                                      |
| \} .....                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 7559                                                                      |
| \_ 44, 639, 7873, 9126                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                                           |
| \_comp .....                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 3953,<br>4159, 4511, 5207, 6026, 6050, 6134                               |
| \_customthiscomp .....                                                                                                                                                                                                                                                                                                                                                                                                                                                | 6553, 6554, 6566                                                          |
| \_defcomp .....                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 6058, 6992, 7001                                                          |
| \_stex_html_do_output_bool .....                                                                                                                                                                                                                                                                                                                                                                                                                                      | 278,<br>279, 282, 287, 291, 322, 325, 2154, 7501                          |
| \_thiscomp .....                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 6536, 6547, 6548                                                          |
| \_varcomp 4492, 4791, 4800, 4957, 4995,<br>5241, 5683, 5982, 5995, 6008, 6054                                                                                                                                                                                                                                                                                                                                                                                         |                                                                           |
| \  .....                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 2084, 2087, 2091                                                          |
| <b>A</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                                           |
| \activateexcursion .....                                                                                                                                                                                                                                                                                                                                                                                                                                              | 109                                                                       |
| \addbibresource .....                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 80, 1954                                                                  |
| \addcontentsline .....                                                                                                                                                                                                                                                                                                                                                                                                                                                | 7871                                                                      |
| \addmhbibresource .....                                                                                                                                                                                                                                                                                                                                                                                                                                               | 80, 1948                                                                  |
| \addtocounter .....                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 8047, 8458, 8459                                                          |
| \AddToHook .....                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 1037, 1040,<br>7305, 7372, 7375, 7487, 8347, 9147                         |
| \aftergroup .....                                                                                                                                                                                                                                                                                                                                                                                                                                                     | <u>143</u> , 5195, 7370, 7484                                             |
| \afterprematurestop .....                                                                                                                                                                                                                                                                                                                                                                                                                                             | 108, 7930, 7941                                                           |
| \anscls .....                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 8783, 8817, 8851                                                          |
| \apply .....                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 88                                                                        |
| \arabic .....                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 8388, 8391, 9193                                                          |
| \arg .....                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 46, 93, 5420                                                              |
| \argarraymap .....                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 92, 4220, 4731                                                            |
| \argmap .....                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 92, 4219, 4361, <u>4707</u> , 7542                                        |
| \argsep .....                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 40, 92, 4218,<br>4356, <u>4691</u> , 7510, 7511, 7512, 7518               |
| \assign .....                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 64, <u>129</u> , 2990, 3346, 3473                                         |
| <b>B</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                                           |
| assignment (env.) .....                                                                                                                                                                                                                                                                                                                                                                                                                                               | <u>73</u> , <u>116</u> , 9167                                             |
| \assignmentautorefname .....                                                                                                                                                                                                                                                                                                                                                                                                                                          | 9200                                                                      |
| \assignMorphism .....                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 129, 2991, 3475                                                           |
| \assumption .....                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 7283, <u>7388</u>                                                         |
| \ast .....                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 7533                                                                      |
| \AtBeginDocument .....                                                                                                                                                                                                                                                                                                                                                                                                                                                | 1007,<br>1259, 1445, 1448, 1496, 7742, 7885                               |
| \AtEndDocument .....                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 647, 1497                                                                 |
| \AtEndOfPackageFile .....                                                                                                                                                                                                                                                                                                                                                                                                                                             | 2035, 2049, 2062                                                          |
| \author .....                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 8212                                                                      |
| \autoref .....                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 84, 1641                                                                  |
| <b>C</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                                           |
| \backmatter .....                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 1460, 1461, 1462                                                          |
| \baselineskip .....                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 8170                                                                      |
| \beameritemnestingprefix .....                                                                                                                                                                                                                                                                                                                                                                                                                                        | 8286                                                                      |
| \begin ... 96, <u>135</u> , <u>136</u> , <u>141</u> , 201, 1302,<br>1382, 2046, 2059, 2077, 2234, 2256,<br>2273, 2523, 2781, 2827, 4754, 6887,<br>7197, 7230, 7252, 7290, 7361, 7433,<br>7629, 7874, 8007, 8021, 8031, 8064,<br>8085, 8093, 8131, 8132, 8133, 8134,<br>8135, 8136, 8142, 8144, 8182, 8184,<br>8188, 8247, 8275, 8276, 8428, 8527,<br>8549, 8601, 8665, 8714, 8766, 8825,<br>8866, 8881, 8966, 8983, 9031, 9036,<br>9041, 9046, 9126, 9199, 9249, 9350 |                                                                           |
| \begingroup .....                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 358, 1926                                                                 |
| \bf .....                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 8169                                                                      |
| \bfseries .....                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 8511                                                                      |
| \bgroup .....                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 3796, 4837, 5049, 8020, 8109, 8124,<br>8232, 8235, 8618, 8679, 8728, 8780 |
| \bigcirc .....                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 8951                                                                      |
| \bigskip .....                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 8480, 9214                                                                |
| blindfragment (env.) .....                                                                                                                                                                                                                                                                                                                                                                                                                                            | 82                                                                        |
| bool commands:                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                           |
| \bool_if:NTF 196, 618, 670, 671, 677,<br>1144, 1154, 1156, 1215, 1236, 1449,<br>2137, 2392, 2742, 2775, 2959, 3455,<br>3664, 4852, 4955, 5205, 5239, 5781,<br>5875, 5881, 6016, 6982, 7258, 7306,<br>7344, 7363, 7366, 7368, 7373, 7376,<br>7384, 7426, 7440, 7462, 7488, 7777,<br>7785, 7822, 7901, 7945, 8106, 8112,<br>8146, 8161, 8211, 8246, 8256, 8334,                                                                                                         |                                                                           |

```

8340, 8410, 8431, 8466, 8471, 8481,
8509, 8530, 8538, 8541, 8554, 8555,
8559, 8560, 8676, 8687, 8725, 8736,
8777, 8789, 8853, 8858, 8927, 8940,
9005, 9018, 9126, 9128, 9132, 9316
\bbool_if:nTF ..... 282
\bbool_if_exist:NTF ..... 255, 271
\bbool_lazy_any:nTF ..... 4048, 4088
\bbool_lazy_any_p:n ..... 774
\bbool_new:N ..... 278, 2134, 2377, 2800, 4763, 5175,
5750, 5872, 8367, 8405, 8406, 8407
\bbool_not_p:n ..... 773, 777
\bbool_set_false:N .. 181, 287, 325,
1141, 1165, 2135, 2154, 2378, 2755,
2822, 3428, 4623, 4766, 5179, 5751,
5876, 7116, 7265, 7313, 7341, 7482,
7501, 7502, 7761, 7802, 7961, 8373,
8408, 8447, 8450, 8903, 8908, 8914
\bbool_set_true:N ... 186, 257, 273,
279, 291, 322, 607, 613, 1138, 2153,
2739, 2819, 3440, 3450, 3661, 3797,
4857, 5097, 5176, 5405, 5414, 5530,
5541, 5639, 5724, 5782, 5793, 5826,
5855, 5901, 6097, 6508, 6539, 6556,
6601, 7128, 7295, 7774, 7814, 7959,
7967, 7968, 7969, 7970, 7971, 7972,
8443, 8448, 8451, 8514, 8901, 8913
\bbool_while_do:Nn ..... 1139
\bbool_while_do:nn ..... 772, 1986, 7132, 7144, 7156, 7173, 7185
\l_tmpa_bool .. 3428, 3440, 3450, 3455
box commands:
\box_clear:N ..... 2157
\c_empty_box .. 3978, 3985, 8855, 8860
\l_tmpa_box 2152, 2157, 3676, 4211,
8109, 8124, 8618, 8679, 8728, 8780
boxed ..... 110
C
\catcode ..... 44, 53, 358, 639, 1055,
1694, 2083, 2084, 2085, 2086, 2087,
2088, 2090, 2091, 2092, 2434, 2435
\cdot ..... 7541, 7554
\centering ..... 7875, 8065
\chapter ..... 27, 82, 7910, 7911
\chaptername ..... 7846, 7907, 7908
\chapertitlename ..... 7846
\circ ..... 54
\clearpage ..... 1455, 1465, 1476, 1487
clist commands:
\clist_clear:N 87, 405, 3705, 4137,
5098, 6390, 6489, 6804, 6809, 7390
\clist_count:N ..... 6370, 6380
\clist_count:n ..... 4739, 6472
\clist_get>NN ..... 461, 512
\clist_if_empty:NTF ..... 180, 457, 508, 3813, 4309,
6369, 6415, 6672, 6876, 8868, 8968
\clist_if_in:NnTF ..... 66, 69, 93, 6970, 8352, 8355, 8358,
8361, 8954, 9152, 9155, 9158, 9161
\clist_if_in:nnTF ..... 6498
\clist_item:Nn ..... 4318
\clist_item:nn ..... 4750
\clist_map_function>NN ..... 5099, 6418
\clist_map_function:nN ..... 2692, 2717, 3595, 3615, 3636
\clist_map_inline:Nn ..... 88, 97, 183, 2844, 3815, 4875,
5068, 6197, 6247, 6673, 6786, 6822
\clist_map_inline:nn ..... 369, 418, 5571, 5670, 5699, 6195, 7072
\clist_pop>NN ..... 4320
\clist_put_right:Nn ..... 89, 5112, 5116, 6397, 6516, 6518
\clist_set:Nn 38, 85, 4976, 8351, 9151
\clist_set_eq>NN ..... 83, 6892
\l_tmpa_clist 8351, 8352, 8355, 8358,
8361, 9151, 9152, 9155, 9158, 9161
\clstinputmhlisting ..... 83, 2034
\cmhgraphics ..... 83, 2034
\cmhtikzinput ..... 118, 2034
\columnbox ..... 8083, 8085, 8103
\comp ..... 32, 33, 36, 46, 90, 93,
103, 131, 3953, 4159, 4332, 4492,
4511, 4553, 4559, 4561, 4791, 4800,
4995, 5224, 5225, 5559, 5683, 5903,
5908, 5923, 5982, 5995, 6008, 6026,
6028, 6035, 6134, 6357, 6365, 6548,
6562, 6563, 6992, 6993, 7001, 7507,
7509, 7516, 7518, 7520, 7521, 7526,
7528, 7533, 7535, 7536, 7543, 7544,
7545, 7546, 7549, 7551, 7555, 7556,
7557, 7559, 7569, 7576, 7578, 7580
\compemph ..... 103, 6035
\conclude ..... 7282, 7388
\conclusion 50–52, 99, 101, 6961, 7081, 7097
\copymod ..... 62, 63, 3605, 3607, 3609
\copymodule ..... 3492, 3494
\counterwithin ..... 7914, 7926
\cr ..... 9108, 9109
\crcr ..... 9075
cs commands:
\cs:w ..... 444, 447, 479, 640, 2167,
2396, 2397, 2398, 2405, 2409, 2415,
5303, 6238, 6288, 6290, 6910, 6923
\cs_argument_spec:N ..... 4002

```

```

\cs_end: . 444, 447, 479, 640, 2167,
2396, 2397, 2398, 2407, 2411, 2415,
5303, 6238, 6288, 6290, 6910, 6923
\cs_generate_from_arg_count:Nnnn
..... 3806,
4619, 4866, 5059, 5143, 5344, 5376
\cs_generate_variant:Nn .....
..... 156, 230, 243, 557, 581,
590, 602, 626, 686, 716, 871, 876,
880, 964, 1125, 1150, 1554, 1882,
2199, 2557, 2561, 2566, 2584, 2633,
2655, 2668, 3123, 5200, 5217, 5534
\cs_gset:Npn ..... 2430
\cs_if_eq:NNTF ..... 139,
886, 940, 1421, 2223, 2234, 2237,
2256, 2259, 4004, 4692, 5615, 5617
\cs_if_exist:NTF ..... 308,
1093, 1327, 1331, 1369, 1372, 1396,
1400, 1429, 1435, 1450, 1460, 1473,
1484, 1640, 1788, 1812, 1832, 1838,
1842, 2105, 2645, 2650, 3980, 3987,
3999, 4430, 4438, 4890, 4897, 5084,
5130, 5294, 5320, 5664, 5903, 5908,
6454, 6751, 6754, 6758, 6761, 6766,
6769, 6773, 6776, 7841, 7846, 7849,
7853, 7857, 7861, 7865, 7907, 7910,
7913, 7919, 7922, 7925, 8067, 8075,
8087, 8097, 8232, 8235, 8413, 9180
\cs_new:Nn ..... 208, 431, 545, 551,
558, 567, 583, 592, 765, 885, 2208,
2391, 2439, 2950, 3773, 4107, 4120,
4324, 4391, 4408, 4413, 4568, 5139,
5352, 5449, 6175, 6230, 6426, 6432,
6435, 6445, 6867, 8393, 9061, 9073
\cs_new:Npn 543, 620, 621, 629, 630,
881, 2891, 2922, 4113, 4126, 5595, 5968
\cs_new_nopar:Nn ..... 365, 380
\cs_new_protected:Nn .....
..... 43, 52, 60, 65, 75, 135,
157, 215, 226, 236, 245, 263, 285,
328, 334, 348, 354, 436, 443, 456,
470, 478, 497, 507, 522, 544, 623,
633, 645, 650, 669, 681, 693, 709,
718, 733, 791, 798, 828, 863, 873,
877, 899, 916, 929, 935, 939, 966,
970, 986, 1002, 1008, 1021, 1085,
1092, 1115, 1126, 1135, 1152, 1162,
1171, 1198, 1245, 1253, 1299, 1307,
1311, 1321, 1367, 1380, 1390, 1427,
1507, 1517, 1543, 1583, 1617, 1629,
1639, 1647, 1687, 1745, 1756, 1787,
1793, 1797, 1811, 1818, 1860, 1885,
1901, 1925, 1936, 1948, 1975, 2005,
2079, 2095, 2118, 2122, 2126, 2141,
2145, 2151, 2162, 2170, 2200, 2222,
2228, 2246, 2269, 2278, 2294, 2301,
2314, 2327, 2342, 2350, 2359, 2380,
2402, 2423, 2445, 2449, 2457, 2469,
2554, 2558, 2562, 2571, 2579, 2585,
2603, 2610, 2622, 2636, 2643, 2656,
2670, 2687, 2696, 2707, 2712, 2721,
2732, 2737, 2761, 2779, 2793, 2801,
2842, 2851, 2863, 2867, 2879, 2887,
2895, 2910, 2923, 2969, 2975, 2996,
3039, 3072, 3139, 3158, 3178, 3184,
3189, 3216, 3247, 3272, 3298, 3348,
3392, 3459, 3542, 3570, 3580, 3675,
3683, 3746, 3758, 3777, 3822, 3830,
3884, 3893, 3975, 3984, 3989, 3996,
4017, 4034, 4047, 4058, 4060, 4073,
4084, 4176, 4192, 4243, 4255, 4282,
4303, 4308, 4329, 4340, 4349, 4459,
4501, 4543, 4572, 4579, 4600, 4614,
4650, 4655, 4660, 4691, 4764, 4808,
4822, 4835, 4885, 4903, 4916, 4922,
4932, 4940, 5003, 5017, 5030, 5031,
5033, 5077, 5110, 5129, 5141, 5153,
5182, 5204, 5219, 5238, 5263, 5268,
5273, 5340, 5356, 5362, 5397, 5403,
5411, 5466, 5527, 5536, 5552, 5565,
5599, 5610, 5631, 5663, 5695, 5741,
5744, 5754, 5767, 5780, 5803, 5804,
5805, 5809, 5835, 5838, 5864, 5874,
5962, 6014, 6035, 6129, 6154, 6161,
6178, 6184, 6233, 6293, 6301, 6308,
6313, 6367, 6389, 6396, 6400, 6463,
6488, 6514, 6546, 6552, 6571, 6583,
6594, 6612, 6622, 6662, 6671, 6679,
6688, 6698, 6709, 6723, 6747, 6829,
6881, 6927, 6940, 6978, 7049, 7065,
7141, 7229, 7236, 7240, 7251, 7266,
7383, 7406, 7424, 7739, 7835, 7906,
7918, 7957, 7995, 8019, 8024, 8030,
8039, 8111, 8266, 8498, 8592, 8656,
8705, 8756, 8899, 8953, 9092, 9102
\cs_new_protected:Npn ..... 205, 337, 340, 344,
345, 428, 688, 892, 895, 1241, 1325,
1394, 1410, 1498, 1500, 1831, 1837,
1841, 1856, 2214, 2297, 2373, 2856,
2916, 2954, 2982, 3013, 3027, 3110,
3125, 3259, 3588, 4428, 4585, 4592,
4642, 4698, 4707, 4732, 5083, 5096,
5161, 5167, 5171, 5201, 5251, 5292,
5318, 5443, 5579, 5893, 5913, 5920,
6050, 6054, 6058, 6064, 6068, 6072,
6076, 6080, 6084, 6088, 6092, 6096,
6334, 6356, 6361, 6452, 6471, 6484,
```

6536, 6783, 6820, 7130, 7154, 7171,  
 7183, 7226, 7589, 8504, 8647, 8650  
`\cs_parameter_spec:N` ..... 547, 561  
`\cs_prefix_spec:N` ..... 559  
`\cs_set:Nn` 8880, 8895, 8956, 8982, 8997  
`\cs_set:Npn` ..... 448, 483, 488,  
 636, 637, 820, 821, 822, 823, 944,  
 951, 1094, 1097, 1110, 2176, 2425,  
 2458, 2672, 2674, 2677, 2690, 2715,  
 2740, 2754, 3807, 4020, 4213, 4214,  
 4218, 4219, 4220, 4332, 4362, 4603,  
 4619, 4713, 4738, 4867, 4905, 5060,  
 5143, 5345, 5377, 5697, 6491, 6497,  
 6558, 7498, 7499, 7500, 7817, 8116  
`\cs_set:Npx` ..... 2614  
`\cs_set_eq:NN` 41, 217, 221, 814, 815,  
 816, 1261, 1268, 1386, 2651, 2993,  
 2994, 4000, 4466, 4470, 4617, 4670,  
 4683, 4898, 5131, 5134, 5208, 5210,  
 5242, 5244, 5420, 5423, 5451, 5452,  
 5665, 5668, 5807, 7665, 7677, 8159  
`\cs_set_protected:Nn` ..... 811, 818, 1567, 7836, 7869  
`\cs_set_protected:Npn` .....  
 .... 350, 1280, 1289, 3976, 3977,  
 4356, 4361, 4367, 6062, 7666, 7908,  
 7911, 7920, 7923, 8160, 8183, 8187,  
 8198, 8202, 8288, 8291, 8294, 9229  
`\cs_set_protected:Npx` ..... 6554  
`\cs_undefine:N` ... 212, 222, 269, 1206  
`\l_tmpa_cs` ..... 4213, 4223, 5697, 5715  
`\csname` ..... 267,  
 296, 355, 2081, 2082, 2083, 2084,  
 2085, 2090, 2091, 2092, 2395, 2658  
`\ctikzinput` ..... 118, 9349  
`\CurrentFile` ..... 1024, 1026, 1028  
`\CurrentFilePath` ..... 1024, 1025, 1028  
`\currentgrouplevel` . 227, 249, 250, 251,  
 253, 255, 257, 265, 267, 269, 271, 273  
`\CurrentOption` ..... 10, 7764, 7765,  
 7766, 7767, 7806, 7807, 8328, 9142  
`\Currentsectionlevel` ..... 82, 1289  
`\currentsectionlevel` ..... 82, 1280

**D**

`\DeclareOption` ..... 10  
`\def` ..... 91, 141,  
 302, 306, 309, 311, 357, 560, 1243,  
 1247, 1264, 1269, 2051, 2054, 2065,  
 2069, 3953, 4159, 4180, 4490, 4492,  
 4509, 4511, 4791, 4798, 4800, 4993,  
 4995, 5224, 5225, 5683, 5923, 5952,  
 5966, 5982, 5995, 6008, 6026, 6037,  
 6134, 6561, 6566, 6992, 7001, 7929,

7931, 7996, 7997, 7998, 8002, 8005,  
 8048, 8151, 8168, 8176, 8212, 8213,  
 8215, 8217, 8219, 8221, 8222, 8224,  
 8226, 8228, 8231, 8232, 8234, 8235,  
 8238, 8240, 8242, 8286, 8388, 8391,  
 8392, 8852, 8857, 9193, 9256, 9258  
`\defemph` ..... 103, 6035  
`\Definename` ..... 100, 6936, 6966, 7019, 7025  
`\define` ..... 24, 36,  
 45, 99, 100, 103, 6935, 6965, 7011, 7017  
`\definiendum` ..... 24, 36,  
 45, 99, 100, 103, 6933, 6963, 7005, 7009  
`definiendum` ..... 6978  
`\definiens` ..... 47,  
 57, 64, 99–101, 6951, 7028, 7046, 7047  
`\defnotation` .....  
 .... 36, 45, 100, 6934, 6964, 7000, 7003  
`\detokenize` 246, 355, 1660, 8352, 8355,  
 8358, 8361, 9152, 9155, 9158, 9161  
`dim` commands:  
`\dim_compare:nNnTF` ..... 9115  
`\dim_new:N` ..... 8583  
`\dim_zero:N` ..... 8586, 9052  
`\dimexpr` ..... 8147  
`do` commands:  
`\_do_comp:nNn` .....  
 .... 4368, 6035, 6051, 6055, 6059  
`\dobracket` ..... 92  
`\dobrackets` ..... 92, 5868, 5921  
`\dowithbrackets` ..... 5920  
`\due` ..... 73, 116  
`\duration` ..... 74, 116

**E**

`\edef` ..... 2083, 2084, 2085, 6548, 8013  
`\egroup` .....  
 .... 3820, 4882, 5075, 8027, 8109, 8126,  
 8232, 8235, 8632, 8693, 8742, 8795  
`\eject` ..... 9126  
`\ellipses` ..... 95,  
 96, 4717, 4718, 5111, 5113, 5671, 5709  
`\else` ..... 295, 307, 1889,  
 1972, 2210, 5898, 6077, 8286, 8510,  
 8615, 8631, 9114, 9330, 9335, 9340  
`\emph` 17, 19, 102, 6085, 7213, 7301, 7318,  
 8637, 8698, 8747, 8800, 8933, 9011  
`\end` .. 136, 141, 1308, 1362, 2046, 2059,  
 2077, 2237, 2259, 2282, 2542, 2797,  
 2875, 4756, 6904, 7211, 7237, 7299,  
 7316, 7367, 7369, 7447, 7451, 7653,  
 7879, 7933, 7942, 8015, 8027, 8041,  
 8081, 8095, 8103, 8116, 8118, 8131,  
 8132, 8133, 8134, 8135, 8136, 8157,  
 8185, 8189, 8192, 8249, 8283, 8284,

8462, 8547, 8565, 8624, 8629, 8685,  
 8690, 8734, 8739, 8787, 8792, 8849,  
 8885, 8891, 8987, 8993, 9033, 9038,  
 9043, 9048, 9126, 9213, 9255, 9352  
`\endcsname` ..... 267, 295, 296,  
 355, 2081, 2082, 2083, 2084, 2085,  
 2090, 2091, 2092, 2395, 2658, 8286  
`\endgroup` ..... 360, 362, 1933  
`\endinput` ..... 1768, 2284  
`\ensuremath` ..... 5783, 5794  
 environments:  
     `assignment` ..... 73, 116, 9167  
     `blindfragment` ..... 82  
     `exnote` ..... 1, 8705  
     `extstructure` ..... 96, 6193  
     `extstructure*` ..... 96  
     `frame` ..... 1, 7957  
     `gnote` ..... 1, 8754  
     `hint` ..... 1, 8653  
     `mathstructure` ..... 96, 6107  
     `mcb` ..... 1, 8862  
     `mmtinterface` ..... 7619  
     `nassertion` ..... 1  
     `ndefinition` ..... 1  
     `nexample` ..... 1  
     `note` ..... 1  
     `nparagraph` ..... 1, 1  
     `nsproof` ..... 1  
     `problem` ..... 1  
     `sassertion` ..... 99  
     `scb` ..... 8950  
     `sdefinition` ..... 99  
     `sexample` ..... 99  
     `sfragment` ..... 82  
     `smodule` ..... 86, 2514  
     `solution` ..... 1, 8582  
     `sparagraph` ..... 99  
     `sproblem` ..... 8393  
     `sproof` ..... 101, 7229  
     `stex_annotation_env` ..... 141  
     `subproblem` ..... 8505  
     `subproof` ..... 7322  
     `testheading` ..... 74, 116  
`\eq` ..... 32, 42  
`\eqstep` ..... 7284, 7388  
`\equal` ..... 31  
`\errmessage` ..... 7696, 7710, 7717, 7731  
`\escapechar` ..... 53, 1055  
`\everyeof` ..... 2163  
`\excursion` ..... 109, 8244  
`\excursiongroup` ..... 109, 8261  
`\excursionref` ..... 109, 8245, 8258  
`exnote (env.)` ..... 1, 8705  
`\exnote` ..... 8399, 8753, 8877, 8979

exp commands:

`\exp_after:wN` ..... 355, 444,  
 447, 479, 546, 698, 887, 2165, 2209,  
 2210, 2211, 2351, 2396, 2397, 2398,  
 2405, 2409, 2413, 2414, 2936, 2957,  
 3573, 3810, 4109, 4122, 4721, 4749,  
 4870, 5063, 5300, 5302, 5310, 5312,  
 5358, 5365, 5366, 5367, 5386, 5419,  
 5582, 5584, 5686, 5687, 5688, 5701,  
 5896, 5959, 6005, 6021, 6237, 6287,  
 6290, 6614, 6910, 6923, 6987, 7022  
`\exp_args:N` 534, 539, 553, 554, 886,  
 940, 1028, 1100, 1127, 1651, 1800,  
 1813, 1827, 1843, 1846, 1849, 1851,  
 2345, 2839, 2845, 2935, 2938, 3367,  
 3395, 3401, 3769, 3778, 3938, 4002,  
 4004, 4692, 4837, 5034, 5469, 5615,  
 5617, 5626, 6198, 6248, 6335, 6339,  
 6417, 6498, 6547, 6553, 6787, 6823,  
 6916, 7201, 7408, 7740, 9093, 9250  
`\exp_args:NNe` .....  
 ... 54, 197, 200, 594, 1073, 1077,  
 1083, 2459, 2898, 3547, 4041, 4227,  
 4325, 5501, 5504, 5588, 5635, 5720,  
 6518, 6626, 6681, 6749, 8278, 8499  
`\exp_args:Nne` .....  
 ... 206, 546, 2027, 2098, 2483, 2523,  
 2580, 2781, 2859, 4183, 4334, 4373,  
 4492, 4511, 4800, 4995, 6136, 6573,  
 6614, 6682, 6887, 7102, 7252, 7430,  
 7629, 8428, 8527, 8866, 8966, 9249  
`\exp_args:NNo` .....  
 832, 1175, 3557, 4573, 4575, 4809, 5004  
`\exp_args:NNno` ..... 266, 683, 832  
`\exp_args:Nnno` ..... 1727, 1735  
`\exp_args:NNNx` ..... 1175, 1716  
`\exp_args:NNnx` .....  
 ... 837, 3074, 4809, 5004, 5480, 5486  
`\exp_args:NNo` ..... 38,  
 48, 66, 69, 89, 93, 164, 172, 696,  
 805, 837, 841, 846, 851, 1016, 1178,  
 1658, 1989, 1999, 3207, 3233, 3554,  
 3560, 3564, 4275, 4288, 4631, 5714  
`\exp_args:Nno` ..... 251, 253, 266,  
 378, 2426, 2961, 2964, 3805, 4041,  
 4865, 5058, 5164, 5169, 5253, 6044,  
 6507, 7050, 8415, 8578, 9182, 9222  
`\exp_args:NNx` .....  
 ... 55, 109, 121, 911, 998, 1898,  
 1946, 2127, 2146, 6640, 8352, 8355,  
 8358, 8361, 9152, 9155, 9158, 9161  
`\exp_args:Nnx` ... 246, 1316, 1592,  
 1660, 1898, 1946, 2814, 3015, 3020,  
 3207, 3233, 3759, 3929, 5142, 5343,

|                              |                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                              | <b>F</b>                                                                                                                                                                                                                                                                                                                                                                         |
| \fbox . . . . .              | 9105, 9115                                                                                                                                                                                                                                                                                                                                                                       |
| \fi . . . . .                | 298, 313, 1893, 1908,<br>1916, 1972, 2211, 5206, 5240, 5911,<br>6015, 6077, 6979, 7552, 7935, 8003,<br>8006, 8105, 8286, 8512, 8619, 8633,<br>8945, 9023, 9120, 9334, 9344, 9345                                                                                                                                                                                                 |
| fibboxed . . . . .           | 105                                                                                                                                                                                                                                                                                                                                                                              |
| file commands:               |                                                                                                                                                                                                                                                                                                                                                                                  |
| \file_if_exist:nTF . . . . . | 634, 651, 1164                                                                                                                                                                                                                                                                                                                                                                   |
| \filepath . . . . .          | 138, 139                                                                                                                                                                                                                                                                                                                                                                         |
| \fillinsol . . . . .         | 114, 8397, 9050                                                                                                                                                                                                                                                                                                                                                                  |
| \first . . . . .             | 137                                                                                                                                                                                                                                                                                                                                                                              |
| \fn . . . . .                | 54                                                                                                                                                                                                                                                                                                                                                                               |
| \foo . . . . .               | 15, 54, 88, 141, 143                                                                                                                                                                                                                                                                                                                                                             |
| \fooname . . . . .           | 15                                                                                                                                                                                                                                                                                                                                                                               |
| \footnote . . . . .          | 8945, 9023, 9107                                                                                                                                                                                                                                                                                                                                                                 |
| \footnotesize . . . . .      | 9251                                                                                                                                                                                                                                                                                                                                                                             |
| \foral . . . . .             | 46, 51                                                                                                                                                                                                                                                                                                                                                                           |
| \forall . . . . .            | 46, 7539, 7545, 7556                                                                                                                                                                                                                                                                                                                                                             |
| frame (env.) . . . . .       | 1, 7957                                                                                                                                                                                                                                                                                                                                                                          |
| \frameimage . . . . .        | 108, 8140                                                                                                                                                                                                                                                                                                                                                                        |
| frameimages . . . . .        | 105                                                                                                                                                                                                                                                                                                                                                                              |
| \frametitle . . . . .        | 8057                                                                                                                                                                                                                                                                                                                                                                             |
| \frontmatter . . . . .       | 1450, 1451, 1452                                                                                                                                                                                                                                                                                                                                                                 |
| \fun . . . . .               | 45                                                                                                                                                                                                                                                                                                                                                                               |
| \funspace . . . . .          | 37                                                                                                                                                                                                                                                                                                                                                                               |
|                              | <b>G</b>                                                                                                                                                                                                                                                                                                                                                                         |
| \gdef . . . . .              | 8244                                                                                                                                                                                                                                                                                                                                                                             |
| \given . . . . .             | 73, 116                                                                                                                                                                                                                                                                                                                                                                          |
| \global . . . . .            | 1243, 1247, 1264, 1269, 2082,<br>7585, 7586, 8648, 8651, 9189, 9191                                                                                                                                                                                                                                                                                                              |
| \gnote (env.) . . . . .      | 1, 8754                                                                                                                                                                                                                                                                                                                                                                          |
| \gnote . . . . .             | 8400, 8806, 8878, 8980                                                                                                                                                                                                                                                                                                                                                           |
| \gnotes . . . . .            | 73, 110, 115                                                                                                                                                                                                                                                                                                                                                                     |
| group commands:              |                                                                                                                                                                                                                                                                                                                                                                                  |
| \group_begin: . . . . .      | 43, 52, 635, 1054,<br>1694, 2174, 2433, 2470, 2789, 2809,<br>2833, 3064, 3288, 3329, 3677, 3731,<br>3896, 3898, 3900, 3902, 3962, 4168,<br>4486, 4532, 4794, 4949, 4989, 5118,<br>5122, 5220, 5373, 5529, 5537, 5611,<br>5632, 5680, 5696, 5899, 5936, 5944,<br>5955, 5977, 5988, 6001, 6538, 6555,<br>6911, 6980, 7029, 7082, 7296, 7363,<br>7385, 7448, 7469, 7497, 8575, 9219 |
| \group_end: . . . . .        | 47, 55, 641,<br>1058, 1741, 2197, 2451, 2484, 2798,<br>2814, 2876, 3069, 3294, 3335, 3679,<br>3739, 3896, 3898, 3900, 3902, 3967,<br>3987, 4171, 4497, 4535, 4805, 4961,<br>5000, 5089, 5107, 5119, 5123, 5164,<br>5297, 5308, 5325, 5336, 5358, 5386,                                                                                                                           |

|                                                                                                                   |                                                                                                                                           |
|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| \group_insert_after:N                                                                                             | 256, 272                                                                                                                                  |
| <b>H</b>                                                                                                          |                                                                                                                                           |
| \halign                                                                                                           | 9108                                                                                                                                      |
| \have                                                                                                             | 7388                                                                                                                                      |
| \hbox                                                                                                             | 1255, 3060, 3363, 3796, 3950,<br>3979, 4350, 4837, 5049, 6209, 6793,<br>7217, 7241, 8232, 8235, 8921, 9084                                |
| hbox commands:                                                                                                    |                                                                                                                                           |
| \hbox_set:Nn                                                                                                      | 3676, 4211                                                                                                                                |
| \hbox_unpack:N                                                                                                    | 3978, 3985, 8855, 8860                                                                                                                    |
| \HCode                                                                                                            | 308                                                                                                                                       |
| \hfil                                                                                                             | 7223, 9108                                                                                                                                |
| \hfill                                                                                                            | 7223                                                                                                                                      |
| hint (env.)                                                                                                       | 1, 8653                                                                                                                                   |
| \hint                                                                                                             | 8398, 8704, 8876, 8978                                                                                                                    |
| hints                                                                                                             | 73, 110, 115                                                                                                                              |
| \hline                                                                                                            | 9249, 9251, 9252, 9253, 9254                                                                                                              |
| \href                                                                                                             | 1838                                                                                                                                      |
| \hrule                                                                                                            | 7217, 8921                                                                                                                                |
| \hspace                                                                                                           | 9118                                                                                                                                      |
| \HTML                                                                                                             | 15, 26                                                                                                                                    |
| \huge                                                                                                             | 9116                                                                                                                                      |
| hwexam commands:                                                                                                  |                                                                                                                                           |
| \l_hwexam_includeassignment_-<br>keys_tl                                                                          | 9180, 9181, 9183, 9220                                                                                                                    |
| \hwexam_kw_*                                                                                                      | 9147                                                                                                                                      |
| \c_hwexam_multiple_bool                                                                                           | 9140                                                                                                                                      |
| \hwexamheader                                                                                                     | 9256, 9295                                                                                                                                |
| \hwexamminutes                                                                                                    | 9258                                                                                                                                      |
| \hyperlink                                                                                                        | 1832, 1833                                                                                                                                |
| \hypertarget                                                                                                      | 1812, 1813                                                                                                                                |
| <b>I</b>                                                                                                          |                                                                                                                                           |
| \id                                                                                                               | 123                                                                                                                                       |
| \if                                                                                                               | 2209, 5894                                                                                                                                |
| \IfBooleanTF                                                                                                      | 3723, 4162,<br>4483, 5457, 7334, 7441, 7464, 8141                                                                                         |
| \ifcsname                                                                                                         | 295, 8286                                                                                                                                 |
| \ifdefempty                                                                                                       | 8277                                                                                                                                      |
| \iffalse                                                                                                          | 7538                                                                                                                                      |
| \IfFileExists                                                                                                     | 6, 20, 1721, 1907,<br>1915, 1988, 1998, 3236, 3241, 3261                                                                                  |
| \IfInputref                                                                                                       | 28, 83, 1965, 8275                                                                                                                        |
| \ifinputref                                                                                                       | 28, 83, 1883, 1972                                                                                                                        |
| \ifintest                                                                                                         | 8339                                                                                                                                      |
| \ifmmode                                                                                                          | 6077                                                                                                                                      |
| \ifnotes                                                                                                          | 106, 7821, 7953                                                                                                                           |
| \ifSGvar                                                                                                          | 109, 8294                                                                                                                                 |
| \ifsolutions                                                                                                      | 111, 8331, 8613, 8626, 8945, 9023, 9104                                                                                                   |
| \ifstexhtml                                                                                                       | 28, 81, 140, 295, 8510                                                                                                                    |
| \ifvmode                                                                                                          | 1908, 1916, 5206, 5240, 6015, 6979                                                                                                        |
| \ifx                                                                                                              | 2081, 7932, 8001, 8004, 9327, 9328, 9336                                                                                                  |
| \ignorespaces                                                                                                     | 4805, 5000, 8107                                                                                                                          |
| image                                                                                                             | 117                                                                                                                                       |
| \imply                                                                                                            | 51                                                                                                                                        |
| \importmodule                                                                                                     | 38, 49, 55, 93–95,<br>130, 133, 135, 136, 143, 212, 2987, 3266                                                                            |
| \includeassignment                                                                                                | 9218                                                                                                                                      |
| \includegraphics                                                                                                  | 83, 2044, 9320                                                                                                                            |
| \includeproblem                                                                                                   | 72, 114, 8567                                                                                                                             |
| \indent                                                                                                           | 5206, 5240, 6015, 6979                                                                                                                    |
| \infprec                                                                                                          | 90, 91, 4275, 5866, 5869, 5902, 7509                                                                                                      |
| \inline*                                                                                                          | 100                                                                                                                                       |
| \inlineass                                                                                                        | 50, 55, 58, 100                                                                                                                           |
| \inlinedef                                                                                                        | 50, 100                                                                                                                                   |
| \inlineex                                                                                                         | 100                                                                                                                                       |
| \input                                                                                                            | 27, 80, 135, 40, 317,<br>1031, 1920, 2011, 8349, 8353, 8356,<br>8359, 8362, 9149, 9153, 9156, 9159,<br>9162, 9256, 9329, 9332, 9338, 9342 |
| \inputassignment                                                                                                  | 74, 116                                                                                                                                   |
| \inputref                                                                                                         | 27, 28, 82, 83, 102, 1883,<br>8159, 8160, 8253, 8278, 8578, 9222                                                                          |
| \inputref*                                                                                                        | 107, 8159                                                                                                                                 |
| \inputreffalse                                                                                                    | 1883, 1892                                                                                                                                |
| \inputreftrue                                                                                                     | 1890, 1927                                                                                                                                |
| \insertframenumber                                                                                                | 8238, 8240                                                                                                                                |
| \insertshortauthor                                                                                                | 8231                                                                                                                                      |
| \insertshortdate                                                                                                  | 8242                                                                                                                                      |
| \insertshorttitle                                                                                                 | 8234                                                                                                                                      |
| \inset                                                                                                            | 45                                                                                                                                        |
| int commands:                                                                                                     |                                                                                                                                           |
| \int_case:nn                                                                                                      | 1368                                                                                                                                      |
| \int_case:nnTF                                                                                                    | 1326, 1395, 1428                                                                                                                          |
| \int_compare:nNnTF                                                                                                |                                                                                                                                           |
| 248, 2025, 2097, 3548, 4200, 4235,<br>4745, 5327, 5614, 5625, 5674, 5880,<br>6370, 6380, 6472, 6855, 7163, 9236   |                                                                                                                                           |
| \int_compare_p:nNn                                                                                                |                                                                                                                                           |
| 7133, 7145, 7157, 7174, 7186                                                                                      |                                                                                                                                           |
| \int_decr:N                                                                                                       | 7164, 7192                                                                                                                                |
| \int_eval:n                                                                                                       |                                                                                                                                           |
| 265, 267, 269, 271, 273, 5483, 5489,<br>5879, 6169, 6176, 6303, 8539, 8542,<br>9189, 9232, 9233, 9250, 9283, 9290 |                                                                                                                                           |
| \int_gincr:N                                                                                                      | 1509, 5468,<br>8537, 8595, 8659, 8708, 8759, 9230                                                                                         |
| \int_gset:Nn                                                                                                      | 5432                                                                                                                                      |
| \int_gzero:N                                                                                                      | 5417, 8452                                                                                                                                |
| \int_incr:N                                                                                                       |                                                                                                                                           |
| 1318, 1328, 1332, 1356, 1370,                                                                                     |                                                                                                                                           |

1373, 1376, 1397, 1401, 1432, 1438,  
 3872, 3873, 3874, 3875, 4661, 4744,  
 5580, 7138, 7150, 7161, 7178, 7190,  
 7606, 7880, 8199, 8203, 8820, 9237  
 $\backslash$ int\_new:N ..... 1276,  
 1505, 3828, 4016, 4731, 5410, 5564,  
 5868, 8195, 8505, 8582, 8754, 9228  
 $\backslash$ int\_set:Nn 1412, 1413, 1414, 1415,  
 1416, 1417, 1419, 3044, 3837, 3841,  
 3845, 3849, 3853, 3857, 3861, 3865,  
 3869, 3944, 4023, 4064, 4096, 4634,  
 4739, 4746, 4784, 4908, 5540, 5869,  
 5902, 7131, 7143, 7155, 7172, 7184  
 $\backslash$ int\_step\_function:nN ... 5146, 5346  
 $\backslash$ int\_step\_inline:nn .....  
 ..... 3886, 4267, 4274, 4294, 4555  
 $\backslash$ int\_use:N .....  
 227, 1302, 1382, 1422, 1442, 1510,  
 3374, 3762, 4344, 4462, 4634, 4813,  
 4826, 5008, 5021, 5256, 5432, 5469,  
 5601, 5638, 5723, 5866, 5867, 6846,  
 7609, 7670, 7873, 8200, 8204, 8553,  
 8597, 8661, 8710, 8761, 8822, 9249  
 $\backslash$ int\_zero:N ..... 3831,  
 3832, 4616, 4742, 5570, 7599, 8197  
 $\backslash$ c\_max\_int ..... 5866, 5867  
 $\backslash$ l\_tmpa\_int .....  
 4616, 4619, 4634, 4661, 4742, 4744,  
 4745, 4746, 4750, 7131, 7134, 7137,  
 7138, 7143, 7146, 7149, 7150, 7155,  
 7158, 7161, 7163, 7164, 7166, 7167,  
 7172, 7175, 7178, 7180, 7184, 7187,  
 7190, 7192, 7193, 7599, 7606, 7609  
 intarray commands:  
 $\backslash$ intarray\_gset:Nnn .....  
 ..... 7166, 7180, 7193, 7268  
 $\backslash$ intarray\_gzero:N ..... 7267  
 $\backslash$ intarray\_item:Nn ... 7134, 7137,  
 7146, 7149, 7158, 7167, 7175, 7187  
 $\backslash$ intarray\_new:Nn ..... 7129  
 $\backslash$ interpretmod ..... 3626, 3628, 3630  
 $\backslash$ interpretmodule ..... 3513, 3515  
 $\backslash$ itestfalse ..... 8343  
 $\backslash$ itesttrue ..... 8341  
 invocation commands:  
 $\backslash$ invocation\_macro ..... 123, 125, 131  
 ior commands:  
 $\backslash$ ior\_close:N .... 658, 664, 1059, 1189  
 $\backslash$ ior\_map\_inline:Nn ..... 1174  
 $\backslash$ ior\_new:N ..... 1170  
 $\backslash$ ior\_open:Nn ..... 652, 660, 1172  
 $\backslash$ ior\_open:NnTF ..... 1053  
 $\backslash$ ior\_str\_get:NN ..... 1056  
 $\backslash$ ior\_str\_map\_inline:Nn .... 654, 661  
 $\backslash$ g\_tmpa\_ior ..... 652, 654,  
 658, 660, 661, 664, 1053, 1056, 1059  
 iow commands:  
 $\backslash$ iow\_close:N ..... 647, 657, 1497  
 $\backslash$ iow\_new:N ..... 616, 1495  
 $\backslash$ iow\_now:Nn .....  
 ..... 624, 655, 662, 1523, 1794, 7743, 8499  
 $\backslash$ iow\_open:Nn ..... 646, 653, 1496  
 $\backslash$ g\_tmpa\_iow ..... 653, 655, 657  
 $\backslash$ isassociative ..... 49  
 $\backslash$ iscommutative ..... 49  
 $\backslash$ item ..... 7234, 8553, 8829, 8885, 8987  
 $\backslash$ itshape ..... 7227

**J**

$\backslash$ jobname ..... 77, 141, 646, 651,  
 652, 653, 660, 665, 674, 981, 1496, 1716  
 $\backslash$ join ..... 63

**K**

keys commands:  
 $\backslash$ l\_keys\_choice\_tl ..... 3694  
 $\backslash$ keys\_define:nn ..... 27,  
 378, 7758, 7800, 8261, 8310, 9138, 9308  
 $\backslash$ l\_keys\_key\_str 4145, 4148, 6121, 6124  
 $\backslash$ l\_keys\_key\_tl ..... 4146, 6122  
 $\backslash$ keys\_set:nn ..... 382, 8270  
 keyval commands:  
 $\backslash$ keyval\_parse>NNn ..... 6392

**L**

$\backslash$ label ..... 84, 121, 132, 1535, 8050  
 $\backslash$ labelsep ..... 8009  
 $\backslash$ labelwidth ..... 8010  
 $\backslash$ lambda ..... 7539, 7550, 7551  
 $\backslash$ langle ..... 7526  
 $\backslash$ Large ..... 7818, 8169, 9200  
 $\backslash$ lastslide ..... 8187, 8202  
 $\backslash$ LaTeX ..... 23  
 $\backslash$ latex ..... 23  
 $\backslash$ ldots ..... 7531, 7576, 7578, 7580  
 $\backslash$ leaders ..... 8171  
 $\backslash$ leavevmode ..... 8065  
 $\backslash$ left ..... 5896  
 $\backslash$ leftmargin ..... 8011  
 $\backslash$ let ..... 355, 1451,  
 1452, 1461, 1462, 1724, 2082, 2164,  
 2175, 2182, 2193, 2205, 3312, 3979,  
 3986, 4491, 4510, 4799, 4994, 5223,  
 5941, 5951, 5965, 6027, 6106, 7585,  
 7586, 7745, 7746, 8178, 8955, 8958  
 $\backslash$ libinput ..... 20, 80, 1975  
 $\backslash$ libusepackage ..... 80, 81, 2023, 7949  
 $\backslash$ libusetheme ..... 7948

|                     |                                                                                                                                                                                                                                                                                           |                                                                                                                  |                                                                                                                                 |  |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|--|
| \libusetikzlibrary  | 80, 118, 2079                                                                                                                                                                                                                                                                             | \mmlarg                                                                                                          | 336                                                                                                                             |  |
| \linewidth          | 8636, 8643,<br>8697, 8702, 8746, 8751, 8799, 8804                                                                                                                                                                                                                                         | \mmlintent                                                                                                       | 336                                                                                                                             |  |
| \LoadClass          | 15, 7786, 7788                                                                                                                                                                                                                                                                            | \mmtdef                                                                                                          | 7646, 7656                                                                                                                      |  |
| \long               | 141, 587, 8213, 8222                                                                                                                                                                                                                                                                      | \MMTinclude                                                                                                      | 7590                                                                                                                            |  |
| \lstinputlisting    | 83, 2058                                                                                                                                                                                                                                                                                  | \mmtinterface (env.)                                                                                             | 7619                                                                                                                            |  |
| \lstinputmhlisting  | 83, 2034                                                                                                                                                                                                                                                                                  | \MMTrule                                                                                                         | 7595                                                                                                                            |  |
|                     |                                                                                                                                                                                                                                                                                           | \mname                                                                                                           | 87, 96                                                                                                                          |  |
|                     |                                                                                                                                                                                                                                                                                           | mode commands:                                                                                                   |                                                                                                                                 |  |
|                     |                                                                                                                                                                                                                                                                                           | \mode_if_math:TF                                                                                                 |                                                                                                                                 |  |
|                     |                                                                                                                                                                                                                                                                                           | .. 3363, 3979, 3980, 3987, 5265, 9083                                                                            |                                                                                                                                 |  |
|                     |                                                                                                                                                                                                                                                                                           | \mode_if_vertical:TF                                                                                             | 3978, 3985                                                                                                                      |  |
| \MSC                |                                                                                                                                                                                                                                                                                           | \MSC                                                                                                             | 7589                                                                                                                            |  |
|                     |                                                                                                                                                                                                                                                                                           | msg commands:                                                                                                    |                                                                                                                                 |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_error                                                                                                       | 142                                                                                                                             |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_error:nn                                                                                                    | 1783, 7037, 7090, 8411                                                                                                          |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_error:nnn                                                                                                   |                                                                                                                                 |  |
|                     |                                                                                                                                                                                                                                                                                           | 153, 192, 240, 812, 948, 955, 1977,<br>1980, 2106, 3199, 3255, 3426, 3456,<br>3992, 4694, 4936, 6181, 6658, 6685 |                                                                                                                                 |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_error:nnnn                                                                                                  | 351, 830, 835,<br>854, 889, 961, 1951, 2008, 2918,<br>3006, 3351, 3877, 4453, 4959, 5214,<br>5248, 5494, 5521, 6030, 6295, 6995 |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_fatal:nn                                                                                                    | 1118                                                                                                                            |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_fatal:nnnn                                                                                                  | 1129, 2031, 2100                                                                                                                |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_none:nn                                                                                                     | 79                                                                                                                              |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_redirect_module:nnn                                                                                         | 94                                                                                                                              |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_redirect_name:nnn                                                                                           | 98                                                                                                                              |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_set:nnn                                                                                                     | 76                                                                                                                              |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_warning:nn                                                                                                  | 1072                                                                                                                            |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_warning:nnn                                                                                                 | 1693, 1735                                                                                                                      |  |
|                     |                                                                                                                                                                                                                                                                                           | \msg_warning:nnnn                                                                                                | 433, 1727                                                                                                                       |  |
| \mult               | 40, 41, 91, 92                                                                                                                                                                                                                                                                            | \mult                                                                                                            |                                                                                                                                 |  |
| \multicolumn        |                                                                                                                                                                                                                                                                                           | \multicolumn                                                                                                     | 9250                                                                                                                            |  |
| \multiple           | 73, 115                                                                                                                                                                                                                                                                                   | \multiple                                                                                                        |                                                                                                                                 |  |
|                     |                                                                                                                                                                                                                                                                                           | N                                                                                                                |                                                                                                                                 |  |
| nassertion (env.)   |                                                                                                                                                                                                                                                                                           | nassertion (env.)                                                                                                | 1                                                                                                                               |  |
| \Nat                |                                                                                                                                                                                                                                                                                           | \Nat                                                                                                             | 36                                                                                                                              |  |
| ndefinition (env.)  |                                                                                                                                                                                                                                                                                           | ndefinition (env.)                                                                                               | 1                                                                                                                               |  |
| \neginfpref         | 41, 90, 91, 4245, 4248, 4257,<br>4260, 4273, 5085, 5321, 5332, 5866                                                                                                                                                                                                                       | \neginfpref                                                                                                      |                                                                                                                                 |  |
| \newcommand         | 444, 479,<br>1577, 1958, 1966, 1971, 2016, 2023,<br>2053, 2059, 2067, 2077, 2104, 7492,<br>7692, 7713, 7930, 7999, 8061, 8162,<br>8166, 8245, 8252, 8255, 8272, 8386,<br>8817, 8924, 9002, 9030, 9035, 9040,<br>9045, 9081, 9125, 9128, 9129, 9130,<br>9131, 9132, 9235, 9320, 9325, 9349 | \newcommand                                                                                                      |                                                                                                                                 |  |
| \newcounter         | 7789, 7790, 7791, 7792,<br>7914, 7926, 8385, 8403, 8404, 9178                                                                                                                                                                                                                             | \newcounter                                                                                                      |                                                                                                                                 |  |
| \NewDocumentCommand | 447, 1541, 1773, 1778, 1825,                                                                                                                                                                                                                                                              | \NewDocumentCommand                                                                                              |                                                                                                                                 |  |

```

1897, 1945, 2488, 2567, 4948, 5456,
5935, 5943, 5954, 5976, 5987, 6000,
6910, 7000, 7005, 7011, 7019, 7028,
7071, 7081, 7099, 7425, 7461, 7476,
7595, 7656, 7948, 8140, 8574, 9218
\NewDocumentEnvironment . . . .
    482, 1348, 1357, 7481, 7619, 8181, 8196
\newenvironment . . . .
    1470, 1481, 7195, 7402, 8113, 8115, 9278
\newif 296, 319, 1883, 7821, 8331, 8339, 9285
\newlabel . . . .
    7745, 7746
\newlength . . .
    7951, 7952, 7955
\newline . . .
    8636, 8643, 8697, 8702, 8746, 8751, 8799, 8804
\newpage . . .
    8047, 9132, 9297
\newsavebox . . .
    8083
nexample (env.) . . .
    1
\nextslide . . .
    8183, 8198
\ninputref . . .
    8160, 8162, 8166
\nobreak . . .
    7223
\noindent . . .
    1303, 1908, 1916, 6894, 7213, 8026, 8040, 8063, 8081, 8464, 8637, 8698, 8747, 8800
\nointerlineskip . . .
    8173
\notation . . .
    32, 33, 41, 88, 90, 95, 126, 129, 135, 2986, 4152, 7507, 7510, 7511, 7512, 7525, 7527, 7545, 7546, 7550, 7554, 7556, 7557, 7562, 7565
note (env.) . . .
    1
notes . . .
    73, 105, 110, 115
\notesfalse . . .
    7833
notesslides commands:
    \c_notesslides_notes_bool . . .
        ... 7760, 7761, 7774, 7777, 7785, 7801, 7802, 7814, 7822, 7945, 8106, 8112, 8146, 8161, 8211, 8246, 8256
    \c_notesslides_sectocframes_bool . . .
        ... 7803, 7901
    \c_notesslides_topsect_str 7804, 7827, 7830, 7886, 7889, 7890, 7893
notesslides internal commands:
    \c_notesslides_class_str . . .
        ... 7756, 7759, 7765, 7786
    \c_notesslides_define_chapter: . . .
        ... 7891, 7894, 7906
    \c_notesslides_define_part: . . .
        ... 7895, 7918
    \c_notesslides_do_label:n 7836, 7872
    \c_notesslides_do_sectocframes: . . .
        ... 7835, 7902
    \c_notesslides_do_yes_param:Nn . . .
        ... 7957, 7976, 7979, 7982, 7985, 7988, 7991
    \c__notesslides_docopt_str . . .
        ... 7762
    \c__notesslides_document_str . . .
        ... 7929, 7932
    \c__notesslides_eat: . . .
        ... 8116, 8120, 8123
    \c__notesslides_excursion_args:n . . .
        ... 8266, 8273
    \l__notesslides_excursion_id_str . . .
        ... 8262, 8268
    \l__notesslides_excursion_intro_-tl . . .
        ... 8263, 8267, 8277, 8279
    \l__notesslides_excursion_-mhrepos_str . . .
        ... 8264, 8269, 8278
    \l__notesslides_frame_allowdisplaybreaks_-bool . . .
        ... 7968, 7979
    \l__notesslides_frame_allowframebreaks_-bool . . .
        ... 7967, 7976
    \l__notesslides_frame_box_begin: . . .
        ... 8019, 8030, 8053
    \l__notesslides_frame_box_end: . . .
        ... 8024, 8039, 8055
    \l__notesslides_frame_fragile_-bool . . .
        ... 7969, 7982
    \l__notesslides_frame_label_str . . .
        ... 7966, 7974, 8049, 8050
    \l__notesslides_frame_shrink_-bool . . .
        ... 7970, 7985
    \l__notesslides_frame_squeeze_-bool . . .
        ... 7971, 7988
    \l__notesslides_frame_t_bool . . .
        ... 7972, 7991
    \c__notesslides_inputref: . . .
        ... 8159, 8160, 8163
    \c__notesslides_notes_env:nnnn . . .
        ... 8111, 8131, 8132, 8133, 8134, 8135, 8136
    \l__notesslides_num . . .
        ... 7839, 7841, 7844, 7846, 7849, 7850, 7853, 7854, 7857, 7858, 7861, 7862, 7865, 7866, 7873
    \c__notesslides_setup_itemize: . . .
        ... 7995, 8052
    \l__notesslides_slideshow_-counter_int . . .
        ... 8195, 8197, 8199, 8200, 8203, 8204
    \l__notesslides_tmp . . .
        ... 8295, 8300
    \g__notesslides_variables_prop . . .
        ... 8287, 8289, 8292, 8295
notesslidesfont . . .
    8022, 8037, 8178
notesslidesfooter . . .
    8026, 8040, 8176
notesslidestitleemph . . .
    8059, 8168
notesttrue . . .
    7823
nparagraph (env.) . . .
    1, 1
nsproof (env.) . . .
    1
\null . . .
    7223
\num . . .
    143

```

\number ..... 73, 116  
\numberline ..... 7871  
\numberproblemsin ..... 8385  
  
**O**  
\objective ..... 7713  
\OMDoc ..... 26  
\omdoc ..... 15  
\only ..... 8200, 8204  
\oplus ..... 7517, 7518  
  
**P**  
\PackageError ..... 8296  
\pagebreak ..... 8481, 9216  
\pagenumbering ..... 1457, 1467, 1478, 1489  
\par .. 142, 359, 1301, 1305, 7196, 7223,  
    7266, 7323, 8026, 8040, 8063, 8068,  
    8076, 8081, 8088, 8098, 8464, 8477,  
    8480, 8511, 8636, 8643, 8697, 8702,  
    8746, 8751, 8799, 8804, 8843, 8846,  
    8893, 8995, 9199, 9204, 9214, 9216  
\paragraph ..... 27, 82  
\parsep ..... 7232, 8551, 8827, 8883, 8985  
\part ..... 27, 82, 7922, 7923  
\partname ..... 7841, 7919, 7920  
\parttitlename ..... 7841  
\PassOptionsToClass ..... 7764, 7765  
\PassOptionsToPackage .....  
    10, 7766, 7767, 7778,  
    7781, 7806, 7807, 7824, 8328, 9142  
\pause ..... 8061  
\PDF ..... 15, 26  
\pdfbookmark ..... 7873  
\pdfdest ..... 1537  
peek commands:  
\peek\_charcode:NTF .....  
    ... 3581, 5079, 5092, 5155, 5157,  
    5279, 5286, 6317, 6323, 6464, 6475  
\peek\_charcode\_remove:NTF .....  
    ... 5078, 5154, 5270, 5275,  
    5277, 5284, 5357, 5745, 6322, 6572  
\phantom ..... 9116  
\Pi ..... 7539  
\plus ..... 39–41, 44, 90–92  
\precondition ..... 7692  
\prematurestop ..... 108, 7929  
\premise ..... 52, 101, 6962, 7099, 7106  
prg commands:  
\prg\_new\_conditional:Nnn .....  
    ... 231, 281, 533,  
    538, 748, 758, 2136, 2546, 2550,  
    3654, 3665, 3669, 5500, 5613, 5624  
\prg\_new\_protected\_conditional:Nnn ..... 768  
  
\prg\_return\_false: ..... 233, 283, 536,  
    541, 749, 753, 759, 761, 784, 788,  
    2137, 2548, 2552, 3655, 3670, 5518,  
    5522, 5619, 5620, 5621, 5627, 5628  
\prg\_return\_true: ..... 233, 283,  
    536, 541, 751, 753, 761, 788, 2137,  
    2548, 2552, 3666, 5516, 5619, 5627  
\printexcursion ..... 109  
\printexcursions ..... 8244, 8253, 8274, 8282  
problem (env.) ..... 1  
problem commands:  
\g\_problem\_id\_counter .....  
    ... 8582, 8595, 8597,  
    8659, 8661, 8708, 8710, 8759, 8761  
\l\_problem\_inputproblem\_keys\_tl .....  
    ... 8413, 8414, 8416, 8576  
\problem\_mcc\_box\_tl .....  
    ... 8865, 8885, 8895, 8919  
\problem\_scc\_box\_tl .....  
    ... 8951, 8973, 8987, 8997  
problem@kw@points commands:  
\problem@kw@points: ..... 8844  
\problemheader ..... 8464, 8484  
problems internal commands:  
\\_\_problems\_activate\_macros: ...  
    ... 8393, 8456, 8513  
\l\_\_problems\_anscls\_int .....  
    ... 8754, 8820, 8822  
\c\_\_problems\_boxed\_bool ..... 8324  
\\_\_problems\_do\_yes\_param:Nn ... 8899  
\\_\_problems\_exnote\_start:n .....  
    ... 8705, 8723, 8726  
\l\_\_problems\_fillin\_solution\_tl .....  
    ... 9051, 9074, 9098, 9106, 9110  
\\_\_problems\_fillinsol:nn .....  
    ... 9084, 9086, 9092, 9102  
\\_\_problems\_gnote\_start:n .....  
    ... 8756, 8775, 8778  
\c\_\_problems\_gnotes\_bool .....  
    ... 8314, 8777, 8789  
\l\_\_problems\_has\_min\_bool .....  
    ... 8407, 8450, 8451, 8541, 8560  
\l\_\_problems\_has\_pts\_bool .....  
    ... 8406, 8447, 8448, 8538, 8555  
\\_\_problems\_hint\_start:n .....  
    ... 8656, 8674, 8677  
\c\_\_problems\_hints\_bool .....  
    ... 8316, 8676, 8687  
\l\_\_problems\_in\_problem\_bool ...  
    ... 8405, 8408, 8410, 8443, 8509, 8514  
\\_\_problems\_maybe\_inline:n .....  
    ... 8953, 8965, 8993  
\\_\_problems\_mccline:n 8880, 8895, 8937

```

\c__problems_min_bool . . . . .
    ..... 8322, 8471, 8559, 8858
\l__problems_min_tl . . 8445, 8449,
    8459, 8472, 8473, 8500, 8516, 8542
\c__problems_notes_bool . . .
    ..... 8312, 8725, 8736
\__problems_oldpar: . . . 8955, 8958
\__problems_parse_fillin_-
    arg:nnnn 9055, 9056, 9057, 9061, 9073
\l__problems_path_seq . . .
    .. 8425, 8426, 8519, 8520, 8524, 8525
\l__problems_prob_imports_tl . 8382
\l__problems_prob_refnum_int . 8383
\c__problems_pts_bool . . .
    ..... 8320, 8466, 8554, 8853
\l__problems_pts_tl . . 8444, 8446,
    8458, 8467, 8468, 8500, 8515, 8539
\__problems_record_problem: . .
    ..... 8460, 8498
\__problems_sccline:n 8982, 8997, 9015
\__problems_solution_start:n . .
    ..... 8592, 8611, 8614
\c__problems_solutions_bool . .
    ..... 8318, 8334
\g__problems_subproblem_int . .
    ..... 8452, 8505, 8537, 8553
\c__problems_test_bool . .
    .. 8326, 8340, 8481, 9126, 9128, 9132
\ProcessKeysOptions . . .
    .... 39, 7770, 7810, 8333, 9145, 9314
\ProcessOptions . . . . . 11
\prod . . . . . 7546, 7557
\prop . . . . . 42
prop commands:
    \prop_clear:N . 1173, 2880, 2881, 6663
    \prop_const_from_keyval:Nn . 109, 121
    \prop_gclear:N . .
        ..... 2319, 2320, 2321, 2431, 5415
\prop_get:NnN . . . . . 1207
\prop_get:NnNTF . .
    ..... 188, 903, 904, 1799, 1903,
        1904, 1979, 2934, 2955, 5501, 8295
\prop_gput:Nnn . . .
    ..... 2459, 2580, 2592, 2594, 2627, 5444,
        5480, 5486, 5504, 7067, 8289, 9231
\prop_gset_eq:NN . . . 1190, 1221, 1224
\prop_gset_from_keyval:Nn . .
    ..... 1193, 1218, 2424, 2426, 5428
\prop_if_exist:NTF . . .
    ..... 901, 1099, 1116, 1602,
        1696, 1798, 1863, 1976, 3079, 5427
\prop_if_in:NnTF . . .
    137, 172, 805, 6623, 6680, 6710, 6724
\prop_item:Nn . . .
    ..... 138, 921, 1100, 1213, 1620,
        1699, 1865, 1868, 1877, 2041, 2055,
        2070, 2860, 3016, 3021, 3082, 3107,
        3118, 3133, 6615, 6655, 6683, 8292
\prop_map_break: . . . . . 2620
\prop_map_break:n 2639, 2675, 2678,
    2756, 3040, 4061, 4924, 4927, 7054
\prop_map_function:NN . . . 2417, 2947
\prop_map_inline:Nn . . . 2427, 2604,
    2637, 2681, 2699, 2702, 2724, 2727,
    2765, 2911, 2930, 2970, 2998, 3002,
        4077, 4917, 6277, 6503, 7051, 9243
\prop_new:N . . . . . 8287, 9225
\prop_put:Nnn . . . 1181, 1182, 1183,
    1184, 1185, 2898, 2902, 3370, 3404,
    4810, 5005, 6689, 6712, 6726, 9238
\prop_remove:Nn . . . . . 3466
\prop_set_eq:NN 1088, 1205, 1209, 1611
\prop_to_keyval:N . . .
    ... 1191, 1194, 1219, 2396, 2397,
        2398, 2405, 2409, 2925, 2928, 5429
\protect . . . . . 7871, 8298
\protected . . . . . 123,
    141, 357, 571, 572, 1269, 5952, 5966
\providecommand . . . 2039, 2046, 7937, 9319
\ProvidesExplClass . . . . . 5, 7753
\ProvidesExplPackage . . .
    ..... 19, 7797, 8307, 9135, 9305
\pts . . . . . 8852
pts . . . . . 73, 110, 115

Q
\quad . . . . . 7850, 7854, 7858,
    7862, 7866, 9082, 9088, 9207, 9210
quark commands:
    \quark_new:N . . . . . 2160, 2443
quark internal commands:
    \q_stex_smsmode_break . . .
        ..... 2160, 2163, 2223

R
\range . . . . . 7526
\realization . . . . . 3537, 3539
\realize . . . . . 63, 64, 3647, 3649, 3651
\ref . . . . . 84, 1643
\refstepcounter . . . . . 8421, 9191
\relax . . . . . 358, 639, 640, 1452, 1462,
    1537, 1694, 2081, 2167, 2205, 2209,
    2434, 2435, 3979, 3986, 8147, 8149
\renamedecl . . . . . 64, 129, 2992, 3390, 3474
\renewcommand . . . . . 7946, 8047, 8057
\renewenvironment . . .
    ... 8000, 8044, 8062, 8084, 8107, 8109

```

repo commands:  
     \repo\_prop ..... 138, 139  
     \reqpts ..... 74, 116  
     \requiremodule ..... 93, 94, 2988, 3339  
     \RequirePackage .....  
         ... 4, 13, 18, 23, 24, 146, 148, 197,  
             200, 5892, 7754, 7772, 7794, 7798,  
             7812, 7825, 7826, 8138, 8308, 8346,  
             9136, 9146, 9306, 9317, 9322, 9323  
     \resizebox ..... 9248, 9331, 9337, 9341  
     \rhd ..... 8002, 8005  
     \right ..... 5897  
     \Rightarrow ..... 7458  
     \rightmargin ..... 7233, 8552, 8828, 8884, 8986  
     \rule ..... 8636, 8643,  
             8697, 8702, 8746, 8751, 8799, 8804  
 rustex commands:  
     \rustex\_direct\_HTML:N .....  
         ... 8069, 8077, 8089, 8099  
     \rustex\_if:TF ..... 8067, 8068,  
             8075, 8076, 8087, 8088, 8097, 8098  
     \rustexBREAK ..... 2133

**S**

sassertion (env.) ..... 99  
 scb (env.) ..... 8950  
   \scb ..... 8396, 8874, 8976, 9000  
   \scct ..... 8981, 9001  
   \scriptsize ..... 7999  
   \scriptstyle ..... 8005  
   sdefinition (env.) ..... 99  
   \second ..... 137  
   \section ..... 26, 27, 82  
   \sectiontitleemph ..... 7817, 7876  
   sectocframes ..... 105  
   \selectlanguage ..... 140, 141  
 seq commands:  
     \seq\_clear:N ..... 182, 682,  
         721, 1982, 2177, 2688, 2713, 2738,  
         2843, 2882, 4198, 4554, 4715, 5569,  
         5633, 5698, 5707, 6143, 6194, 6244,  
         6402, 6664, 6665, 6785, 6821, 7596  
     \seq\_count:N ..... 2025, 2097, 3548, 6855  
     \seq\_gclear:N ..... 2171, 2172, 5416  
     \seq\_gclear\_new:N ..... 984, 985  
     \seq\_get:NN ..... 1015  
     \seq\_get\_right:NN ..... 158, 8426, 8520, 8525  
     \seq\_gpop:NN ..... 1010  
     \seq\_gpush:Nn ..... 998  
     \seq\_gput\_left:Nn ..... 1546, 1551  
     \seq\_gput\_right:Nn .....  
         ... 1549, 2127, 2146, 5445  
     \seq\_gset\_eq:NN .....  
         ... 997, 1013, 1017, 2870, 2871, 2872, 7735  
     \seq\_gset\_split:Nnn ..... 5434  
     \seq\_if\_empty:NTF ..... 169, 719, 737,  
         749, 759, 788, 834, 840, 845, 850,  
         1009, 1012, 1117, 1950, 2007, 4038,  
         6870, 7203, 7254, 7410, 7604, 7734  
     \seq\_if\_empty\_p:N ..... 775, 776, 1986  
     \seq\_if\_exist:NTF ..... 1591, 5433  
     \seq\_if\_in:NnTF ..... 1544, 1545, 1658,  
         1660, 1989, 1999, 2271, 2280, 2657,  
         2697, 2722, 2763, 6515, 6626, 6749  
     \seq\_item:Nn ..... 750,  
         760, 2026, 2098, 4311, 4317, 5502, 6856  
     \seq\_map\_break: ..... 1593, 1597, 2675  
     \seq\_map\_break:n ..... 1597, 2678, 4061  
     \seq\_map\_function:NN ..... 723, 727, 6414  
     \seq\_map\_inline:Nn .....  
         ... 264, 918, 922, 1596,  
             1631, 1953, 2010, 2186, 2680, 4075,  
             4716, 4743, 5708, 6207, 6254, 6263,  
             6791, 6929, 6956, 7274, 7329, 7605  
     \seq\_new:N .....  
         ... 225, 1504, 1550, 2117, 2140, 2290  
     \seq\_pop:NN ..... 720  
     \seq\_pop\_left:NN ..... 168, 703,  
         781, 782, 833, 838, 844, 849, 919,  
         923, 925, 1993, 3550, 3552, 3559, 3563  
     \seq\_pop\_left:NNTF .....  
         ... 1177, 4285, 4287, 4295  
     \seq\_pop\_right:NN ..... 160, 170,  
         738, 793, 795, 800, 802, 804, 1145,  
         2365, 3075, 4037, 4574, 4576, 6405  
     \seq\_push:Nn ..... 726  
     \seq\_put\_left:Nn .....  
         ... 704, 724, 2698, 2723, 6711, 6725  
     \seq\_put\_right:Nn .....  
         ... 164, 190, 228, 741, 796,  
             806, 809, 981, 1990, 1994, 2000,  
             2324, 2354, 2367, 2659, 2764, 2846,  
             2977, 4268, 4275, 4296, 4298, 4556,  
             4718, 4720, 5600, 5636, 5672, 5679,  
             5704, 5710, 5714, 5721, 6144, 6199,  
             6203, 6249, 6681, 6788, 6824, 6838  
     \seq\_reverse:N ..... 6406  
     \seq\_set\_eq:NN ..... 729, 769, 770, 792,  
         799, 917, 980, 1983, 4725, 4753, 5719  
     \seq\_set\_split:Nnn ..... 159,  
         683, 794, 801, 832, 837, 1137, 1176,  
         1984, 2364, 3074, 3547, 3557, 4036,  
         4284, 4288, 4573, 4575, 6404, 7597  
     \seq\_use:Nn .....  
         ... 195, 198, 201, 766, 796, 809,  
             837, 1179, 2369, 3076, 3555, 3561,  
             3565, 4042, 4354, 4559, 4702, 4726,  
             5435, 5558, 6871, 7204, 7255, 7411

```

\l_tmpa_seq ..... 159,
160, 164, 168, 169, 170, 182, 190,
195, 198, 201, 4554, 4556, 4559,
4573, 4574, 4575, 4576, 4715, 4718,
4720, 4725, 4753, 6402, 6404, 6405,
6406, 6414, 7596, 7597, 7604, 7605
\seqmap ..... 96, 5171, 5626
\setbox ..... 135, 8109, 8124, 8618, 8679, 8728, 8780
\setcounter ..... 9189, 9192
\setkeys ..... 2043, 2057, 2072, 8151, 9326
\setlength ..... 7231, 7232, 7233,
7951, 7952, 7956, 8009, 8010, 8011,
8550, 8551, 8552, 8826, 8827, 8828,
8882, 8883, 8884, 8984, 8985, 8986
\setlicensing ..... 107
\setmetatheory ..... 2469
\setnotation ..... 33, 91, 4428
\setsectionlevel ..... 27,
82, 121, 1410, 7828, 7830, 7887, 7889
\setSGvar ..... 109, 8288, 8298
\setsidelogo ..... 107
\setsource ..... 107
\sexample (env.) ..... 99
\sf ..... 8169
\sffamily ..... 8178
\sfragment (env.) ..... 82
sfragment commands:
    \sfragment_do_level:nn ..... 1299, 1311, 1327, 1331,
1335, 1336, 1337, 1338, 1339, 7869
    \sfragment_end: ..... 1307, 1321, 1353
\skipfragment ..... 82, 1394, 1398, 1402
\slideframewidth ..... 7955, 7956, 8032, 8147
\slideheight ..... 7952
\slides ..... 105
\slidewidth ..... 7951, 8035, 8065, 8147, 8149, 8153
\smacro ..... 92, 93
\small ..... 7227
\smallskip ..... 7223, 8468, 8473, 8556,
8561, 8636, 8697, 8746, 8799, 9204
\smodule (env.) ..... 86, 2514
\smodule ..... 2989
\Sn ..... 20, 89, 5923
\sn ..... 20, 24, 89, 5923
\Sns ..... 20, 89, 5923
\sns ..... 20, 89, 5923
\solution ..... 73
\solution (env.) ..... 1, 8582
\solution ..... 8394, 8646, 8875, 8977
\solutions ..... 73, 110, 115
\solutionsfalse ..... 8337, 8651
\solutionstrue ..... 8335, 8648
\source ..... 137
\sparagraph (env.) ..... 99
\spfblock ..... 7286, 7486
\spfjust ..... 7287, 7492, 7495
\spfsketch ..... 7195
\spfsketchenvautorefname ..... 7213
\spfstep ..... 7281, 7388
\spfstepautorefname ..... 7322, 7403
\spfstepenvautorefname ..... 7403
\sproblem (env.) ..... 8393
\sproblemautorefname ..... 8490
\sproof (env.) ..... 101, 7229
\sproofautorefname ..... 7301
\sproofend ..... 7215, 7303, 7320
\square ..... 7216, 8920
\sr ..... 20, 24, 88, 5923
\sref ..... 84, 85, 99, 1555, 8248
\sreflabel ..... 84, 86, 121, 1504
\srefsetin ..... 84, 1577
\srefsym ..... 89, 1825
\srefsymuri ..... 89, 1856
\startsolutions ..... 111, 8647
\stepcounter ..... 1391, 7870, 8046
\stex ..... 14, 15, 81
\stex ..... 15, 24, 81
stex commands:
    \l_stex_aB_args_seq ..... 4702, 4716, 4725, 4726, 4743, 4753,
5558, 5569, 5600, 5633, 5636, 5672,
5679, 5698, 5704, 5708, 5719, 5721
    \stex_activate_module:n ..... 122, 2335, 2384,
2656, 2656, 2668, 2838, 3181, 3279,
3322, 3522, 6215, 6271, 6796, 7642
    \stex_add_definiens:nn ..... 2994, 7041, 7049, 7276, 7331
    \stex_add_definiens_inner:nnnnnnn ..... 7054, 7065
\stex_add_module_notation:nnnnn ..... 122
\l_stex_all_modules_seq ..... 122, 2177, 2290,
2324, 2354, 2657, 2659, 2680, 4075
\l_stex_allow_semantic_bool ..... 4955, 5097, 5175, 5176, 5179, 5205,
5239, 5405, 5414, 5530, 5541, 5639,
5724, 5782, 5793, 5826, 5855, 6016,
6097, 6508, 6539, 6556, 6601, 6982
\stex_annotate:nn ..... 338, 341, 1282, 1291, 1303,
1747, 1967, 1968, 3363, 3799, 3802,
3809, 3814, 3816, 4357, 4363, 4372,
4383, 4394, 4395, 4398, 4399, 4403,
4859, 4862, 4869, 4873, 4876, 5052,
5055, 5062, 5066, 5069, 5100, 5546,

```

5553, 5647, 5756, 5769, 5786, 5811,  
 5819, 5840, 5848, 6042, 6428, 6436,  
 6441, 6649, 6895, 6916, 6993, 7059,  
 7092, 7104, 7201, 7243, 7246, 7288,  
 7342, 7351, 7355, 7408, 7477, 7493,  
 7602, 7607, 8238, 8487, 8831, 8854,  
 8859, 8939, 8942, 9017, 9020, 9093  
`\stex_annotation:nmn` ..... 140, 141  
`\stex_annotation_force_break:n` ...  
                        ..... 327, 328, 334,  
 1304, 3362, 3796, 3814, 4856, 4874,  
 5050, 5067, 5554, 5652, 5761, 5774,  
 5791, 5816, 5824, 5845, 5853, 6440,  
 6895, 6917, 7060, 7093, 7603, 9095  
`\stex_annotation_invisible:n` .....  
                        ..... 141, 329,  
 331, 1361, 1383, 2531, 2788, 2832,  
 5202, 5458, 7241, 7600, 7637, 8532  
`\stex_annotation_invisible:nn` 1255,  
 1422, 1430, 1436, 1442, 1908, 1916,  
 3060, 3284, 3325, 3361, 3395, 3436,  
 3446, 3778, 4350, 4837, 5034, 5792,  
 5825, 5854, 6209, 6265, 6793, 7075,  
 7591, 7638, 7706, 7727, 8436, 9064  
`\stex_annotation_invisible:nnn` .. 141  
`\l_stex_argnames_seq` ..... 124  
`\stex_args_end:` ..... 5419,  
 5443, 5446, 5968, 5972, 6022, 6988  
`\stex_assign_do:n` .....  
                        ..... 2853, 3343, 3348, 3576  
`\l_stex_assoc_args_count` .....  
                        ..... 124, 3828, 3832, 3874, 3875  
`\l_stex_brackets_dones_bool` ...  
                        ... 4623, 5872, 5875, 5876, 5900, 5901  
`\stex_capitalize:n` .....  
                        ..... 1293, 5959, 5962, 6005, 7022  
`\stex_check_term:n` .....  
                        ..... 124, 125, 2176, 3355,  
 3674, 3675, 3683, 3894, 4330, 7499  
`\c_stex_check_terms_bool` .....  
                        ..... 33, 3661, 3664, 7502  
`\stex_close_module:` ... 122, 2391,  
 2391, 2540, 2795, 7583, 7650, 7652  
`\l_stex_current_archive` ..... 1085  
`\l_stex_current_archive_prop` ...  
                        ..... 134, 139, 921, 931,  
 936, 1088, 1099, 1100, 1109, 1198,  
 1602, 1603, 1696, 1699, 1798, 1799,  
 1863, 1865, 1868, 1903, 1904, 1976,  
 1979, 2041, 2055, 2070, 3079, 3082  
`\l_stex_current_args_tl` .....  
 131, 4545, 4549, 5228, 5418, 5419, 5422  
`\l_stex_current_arity_str` .....  
                        ..... 131, 4555, 5117, 5144,  
                        5146, 5227, 5327, 5345, 5346, 5377  
`\l_stex_current_doc_uri` 139, 965,  
 967, 968, 1514, 1519, 1521, 1525,  
 1591, 1593, 1633, 1650, 1651, 6874  
`\l_stex_current_domain_str` .....  
                        ..... 129, 2802, 2805, 2807,  
 2815, 2820, 2829, 2838, 2869, 2884,  
 2896, 2933, 2945, 3413, 3461, 3482,  
 3502, 3522, 3526, 3599, 3619, 3640  
`\g_stex_current_file` .....  
                        . 137–139, 158, 930, 932, 937, 985,  
 988, 990, 992, 993, 994, 997, 998,  
 1013, 1016, 1017, 1606, 1704, 2343,  
 3089, 3095, 3096, 3144, 3163, 3207,  
 3233, 7734, 7735, 8425, 8519, 8524  
`\l_stex_current_language_str` 139,  
 140, 134, 136, 171, 176, 2525, 3194,  
 3197, 3221, 3224, 7631, 8430, 8529  
`\l_stex_current_module` ..... 123, 125  
`\l_stex_current_module_str` .....  
                        . 122, 143, 2178, 2289, 2306, 2323,  
 2324, 2339, 2352, 2353, 2354, 2355,  
 2394, 2395, 2396, 2397, 2398, 2404,  
 2406, 2410, 2415, 2417, 2524, 2534,  
 2547, 2555, 2559, 2580, 2586, 2592,  
 2594, 2604, 2611, 2616, 2625, 2627,  
 2637, 2661, 2664, 2783, 2794, 2828,  
 2839, 2936, 2939, 3368, 3402, 3429,  
 3438, 3448, 3732, 3770, 3779, 3878,  
 3939, 3948, 3957, 3963, 4502, 4521,  
 4522, 6100, 6138, 6143, 6144, 6164,  
 6256, 6278, 6839, 6862, 7050, 7051,  
 7052, 7053, 7066, 7067, 7621, 7622,  
 7625, 7630, 7649, 7651, 7664, 7678  
`\l_stex_current_ns_uri` ..... 139,  
 971, 972, 2361, 2504, 3145, 3150,  
 3151, 3164, 3169, 3170, 7503, 7584  
`\l_stex_current_redo_tl` .....  
                        ..... 5209, 5222, 5233, 5236,  
 5243, 5373, 6373, 6411, 6446, 6596  
`\l_stex_current_return_tl` .....  
 131, 3986, 5229, 5295, 5307, 5328, 5378  
`\stex_current_section_level` ....  
                        ..... 1278, 1284, 1293, 1319, 7881  
`\l_stex_current_symbol_str` .....  
                        ..... 131, 4182, 4212,  
 4331, 4371, 4382, 4491, 4510, 4573,  
 4799, 4951, 4953, 4959, 4994, 5084,  
 5087, 5102, 5130, 5131, 5214, 5226,  
 5248, 5264, 5293, 5294, 5297, 5303,  
 5319, 5320, 5323, 5412, 5494, 5521,  
 5538, 5640, 5641, 5684, 5725, 5726,  
 5758, 5771, 5788, 5813, 5821, 5842,  
 5850, 5903, 5908, 5981, 5994, 6007,

```

    6017, 6030, 6038, 6042, 6044, 6099,
    6454, 6455, 6983, 6991, 6993, 6995
\l_stex_current_term_tl 4956, 5177,
    5231, 5539, 5642, 5643, 5644, 5727,
    5755, 5768, 5783, 5794, 5810, 5827,
    5839, 5856, 6025, 6447, 6453, 6643
\stex_current_this: . 5223, 6096, 6106
\l_stex_current_this_tl .....
    ..... 6098, 6102, 6116, 6119
\l_stex_current_type_tl .....
    ..... 131, 5099, 5230, 6315,
    6370, 6377, 6380, 6384, 6699, 6748
\stex_deactivate_macro:Nn .....
    ..... 142, 348,
    348, 2576, 2983, 2984, 2985, 2986,
    2987, 2988, 2989, 3270, 3339, 3473,
    3474, 3475, 3492, 3513, 3537, 3605,
    3626, 3647, 3743, 3971, 4189, 4540,
    6151, 6225, 6287, 7003, 7009, 7017,
    7025, 7046, 7079, 7097, 7106, 7381,
    7454, 7474, 7479, 7486, 7495, 7593,
    7617, 7690, 8646, 8704, 8753, 8806,
    8851, 8873, 8874, 8875, 8876, 8877,
    8878, 8898, 8949, 8975, 8976, 8977,
    8978, 8979, 8980, 9000, 9027, 9124
\stex_debug:nn .....
    120, 65, 65, 95, 99, 176, 195, 246,
    513, 638, 706, 713, 860, 930, 933,
    968, 972, 983, 1024, 1032, 1060,
    1065, 1084, 1087, 1120, 1163, 1191,
    1202, 1212, 1232, 1248, 1263, 1520,
    1585, 1590, 1630, 1634, 1650, 1653,
    1659, 1661, 1665, 1668, 1674, 1676,
    1679, 1688, 1689, 1715, 1762, 1764,
    1766, 1861, 1867, 1872, 1879, 2229,
    2231, 2247, 2249, 2253, 2270, 2272,
    2279, 2281, 2303, 2316, 2329, 2331,
    2353, 2393, 2471, 2473, 2482, 2586,
    2611, 2619, 2623, 2658, 2924, 2956,
    2960, 2963, 3073, 3078, 3080, 3088,
    3095, 3100, 3104, 3143, 3162, 3179,
    3185, 3192, 3208, 3219, 3234, 3252,
    3260, 3262, 3349, 3356, 3393, 3414,
    3430, 3434, 3444, 3462, 3464, 3546,
    3549, 3553, 3895, 3897, 3899, 3901,
    4018, 4035, 4074, 4085, 4087, 4240,
    4272, 4283, 4333, 4521, 4525, 4601,
    4610, 4615, 4636, 4785, 4904, 4944,
    5163, 5168, 5264, 5269, 5274, 5293,
    5296, 5299, 5319, 5371, 5380, 5404,
    5412, 5528, 5879, 6162, 6234, 6314,
    6854, 6857, 6859, 6863, 7039, 7052,
    7066, 7101, 7275, 7330, 7429, 7626
\c_stex_debug_clist .....
    ..... 28, 38, 66, 69, 83, 87, 89, 93, 97
\c_stex_default_metatheory 2175, 7586
\l_stex_default_notation .....
    ..... 4553, 4558, 4561, 4563, 4564,
    4581, 5134, 5308, 5313, 5334, 5668
\stex_do_default_notation: .....
    ..... 4543, 5133, 5306, 5667
\stex_do_default_notation_op: .....
    ..... 4543, 4544, 4579, 5331
\_stex_do_deprecation:n .....
    ..... 120, 431, 431, 2309, 3719, 3823
\stex_do_for_list: .....
    .. 2993, 6820, 6835, 7199, 7270, 7325
\stex_do_id: .....
    ..... 132, 436,
    436, 1351, 6899, 6914, 7200, 7279,
    7339, 7348, 7357, 7437, 8455, 9195
\stex_do_up_to_module:n .....
    ..... 123, 2554, 2554, 2557, 2564, 3321, 6302
\l_stex_docheader_sect .....
    ..... 121, 1276, 1302, 1318, 1326, 1328,
    1332, 1356, 1368, 1370, 1373, 1376,
    1382, 1395, 1397, 1401, 1412, 1413,
    1414, 1415, 1416, 1417, 1419, 1422,
    1428, 1432, 1438, 1442, 7873, 7880
\stex_eat_exclamation_point: ...
    ..... 3774, 5297, 5308, 5744, 5746
\stex_end: .....
    ..... 420, 428, 2891, 2899
\stex_every_file: .....
    ..... 999, 1002, 1007, 1019, 7738
\stex_every_module:n .....
    ..... 122, 2291, 2294,
    2578, 3309, 3493, 3514, 3538, 3606,
    3627, 3648, 3744, 3972, 4190, 4541,
    6152, 6226, 6289, 6299, 7594, 7618
\g_stex_every_module_tl .....
    ..... 2292, 2295, 2307
\l_stex_every_symbol_tl 5178, 5183,
    5185, 5186, 5192, 5193, 5196, 5234
\stex_execute_in_module:n .....
    ..... 122, 2308, 2382, 2562,
    2562, 2566, 2597, 2629, 3278, 3521,
    3956, 6142, 6202, 6214, 6253, 6270
\stex_fatal_error:n .....
    ..... 142
\stex_fatal_error:nnn .....
    ..... 142
\l_stex_feature_name_str .....
    ..... 129, 2835, 2839,
    2868, 2939, 2944, 2960, 2961, 2963,
    2964, 3007, 3352, 3368, 3402, 3406
\stex_file_in_smsmode:nn .....
    ... 135, 2151, 2170, 2199, 2345, 3248
\stex_file_resolve:Nn .....
    ..... 137,
    138, 687, 693, 709, 716, 878, 979,
    988, 990, 993, 1048, 1075, 1077,
    1605, 1623, 1713, 3096, 3190, 3217

```

```

\stex_file_set:Nn .. 137, 138, 681,
681, 686, 700, 711, 874, 921, 1016, 1073
\stex_file_split_off_ext:NN .....
..... 137, 791, 791, 896, 2343, 3089
\stex_file_split_off_lang:NN ...
..... 137, 791,
798, 893, 2344, 3090, 8425, 8519, 8524
\stex_file_use:N .....
137,
138, 706, 713, 765, 765, 842, 847,
852, 906, 912, 927, 930, 983, 994,
998, 1053, 1073, 1078, 1083, 1127,
1130, 1163, 1164, 1166, 1200, 1606,
1607, 1619, 1698, 1704, 1709, 1714,
1716, 1864, 1871, 1876, 1888, 1891,
1915, 1920, 1931, 1987, 1997, 2037,
2041, 2051, 2055, 2065, 2070, 2346,
3092, 3095, 3096, 3098, 3119, 3134,
3144, 3163, 3191, 3207, 3218, 3233
\c_stex_filepath_sep_str .....
103
\stex_filestack_pop: .....
... 137, 1008, 1008, 1033, 1040, 2196
\stex_filestack_push:n .....
..... 137, 986, 986, 1026, 1028, 2180
\l_stex_fors_seq .....
2843,
2846, 6821, 6824, 6838, 6855, 6856,
6870, 6871, 6929, 6956, 7203, 7204,
7254, 7255, 7274, 7329, 7410, 7411
\stex_get_env:Nn .....
.....
142, 51, 52, 60, 81, 300, 604,
610, 975, 977, 1044, 1046, 1051, 3658
\stex_get_in_morphism:n .....
131, 2845, 2996, 2996, 3342, 3386, 3571
\_stex_get_mathstructure:n .....
..... 2804, 6179, 6184
\stex_get_mathstructure:n .....
..... 6178, 6178, 6196, 6245, 6784
\l_stex_get_structure_module_str
.. 2805, 6180, 6185, 6189, 6203, 6254
\_stex_get_symbol:n . 3990, 3996, 6186
\stex_get_symbol:n .....
124,
125, 131, 1826, 3989, 3989, 4154,
4429, 4945, 5252, 5938, 5946, 5957,
6823, 6981, 7032, 7085, 7694, 7715
\l_stex_get_symbol_args_tl .....
..... 124, 126,
234, 3045, 3375, 3763, 3885, 3887,
3888, 4024, 4065, 4097, 4108, 4110,
4123, 4549, 4814, 4827, 4909, 5009,
5022, 5257, 5422, 6167, 6847, 7671
\l_stex_get_symbol_arity_int ...
..... 124, 126, 234,
3044, 3374, 3762, 3807, 3831, 3837,
3841, 3845, 3849, 3853, 3857, 3861,
3865, 3869, 3872, 3873, 3874, 3875,
3886, 3944, 4016, 4023, 4064, 4096,
4200, 4235, 4267, 4274, 4294, 4344,
4462, 4784, 4813, 4826, 4867, 4908,
5008, 5021, 5060, 5256, 6846, 7670
\l_stex_get_symbol_def_tl .....
..... 124, 2864, 3046,
3350, 4025, 4066, 4098, 4910, 5258
\l_stex_get_symbol_invoke_cs ...
..... 124, 3049, 3379,
4028, 4069, 4101, 4913, 5261, 6188
\l_stex_get_symbol_macro_str ...
..... 131, 3043, 3373
\l_stex_get_symbol_mod_str .....
.....
124, 1828, 2847, 2858, 3041,
3357, 3361, 3371, 3396, 3405, 3948,
3954, 3997, 4021, 4062, 4094, 4160,
4164, 4179, 4342, 4431, 4435, 4439,
4443, 4450, 4454, 4467, 4471, 4522,
4529, 4906, 5254, 6018, 6825, 6984,
7034, 7087, 7680, 7686, 7707, 7728
\l_stex_get_symbol_name_str .....
..... 124, 1828,
2847, 2858, 2997, 3001, 3005, 3042,
3349, 3351, 3357, 3361, 3371, 3393,
3396, 3405, 3406, 3949, 3954, 3991,
3998, 4022, 4063, 4095, 4160, 4164,
4178, 4179, 4342, 4431, 4435, 4439,
4443, 4450, 4454, 4467, 4471, 4479,
4484, 4487, 4523, 4529, 4907, 4933,
4935, 4941, 4943, 5255, 5948, 5959,
5981, 5992, 5994, 6005, 6007, 6018,
6019, 6020, 6021, 6187, 6825, 6984,
6985, 6986, 6987, 7014, 7022, 7030,
7034, 7075, 7083, 7087, 7681, 7686,
7693, 7695, 7707, 7714, 7716, 7728
\stex_get_symbol_or_var:n .....
..... 125, 4903, 4940
\l_stex_get_symbol_return_tl ...
..... 124, 3048, 3378, 4027,
4068, 4100, 4786, 4912, 4975, 5260
\l_stex_get_symbol_type_tl .....
124, 3047, 3377, 4026, 4067, 4099,
4911, 5259, 6189, 6197, 6247, 6786
\stex_get_var:n .....
125, 4478,
4903, 4932, 5979, 5990, 6003, 7073
\c_stex_home_file .....
138, 1042
\stex_if_check_terms: .....
..... 124, 3654, 3665, 3669
\stex_if_check_terms:TF 124, 3653,
3674, 4156, 4481, 4788, 4984, 4987
\stex_if_check_terms_p: ... 124, 3653
\stex_if_do_html: .....
281
\stex_if_do_html:TF .....
..... 140, 278, 290, 1254,

```

```

1281, 1290, 1536, 2522, 2542, 2780,
2796, 2826, 2874, 3059, 3283, 3324,
3360, 3394, 3753, 3940, 3952, 4158,
4528, 4782, 4790, 4836, 4983, 6208,
6264, 6792, 6850, 7074, 7271, 7289,
7299, 7316, 7326, 7360, 7367, 7369,
7407, 7628, 7653, 7685, 8180, 8423,
8462, 8465, 8485, 8522, 8547, 8600,
8610, 8622, 8628, 8664, 8673, 8683,
8689, 8713, 8722, 8732, 8738, 8765,
8774, 8785, 8791, 8830, 8864, 8890,
8938, 8964, 8992, 9016, 9082, 9088
\stex_if_do_html_p: ..... 140
\stex_if_file_absolute:N 137, 748, 758
\stex_if_file_absolute:NTF .....
..... 137, 746, 992, 1076
\stex_if_file_absolute_p:N . 137, 746
\stex_if_file_starts_with:NN 137, 768
\stex_if_file_starts_with:NNTF ..
..... 137, 768, 1199
\stex_if_html_backend:TF .....
..... 140, 295, 320, 327, 336,
619, 1254, 1298, 1360, 1379, 1426,
1938, 1965, 3653, 5206, 5240, 5752,
5808, 5837, 6015, 6041, 6893, 6903,
6979, 7773, 7813, 7938, 8018, 8066,
8074, 8086, 8096, 8237, 9060, 9091
\stex_if_html_backend_p: ... 140, 295
\stex_if_in_module: ..... 2546
\stex_if_in_module:TF .....
..... 122, 2298, 2546, 2562, 4460
\stex_if_in_module_p: .... 122, 2546
\stex_if_module_exists:n ..... 2550
\stex_if_module_exists:nTF .....
..... 122, 2315, 2328, 2550, 3147, 3150,
3166, 3169, 3254, 6198, 6248, 6787
\stex_if_module_exists_p:n 122, 2550
\stex_if_smsmode: ..... 2136
\stex_if_smsmode:TF .....
..... 135, 437, 1246, 1522, 2134,
2201, 2334, 2533, 2541, 3063, 3206,
3232, 3287, 3328, 3480, 3487, 3500,
3508, 3524, 3532, 3597, 3617, 3638,
3730, 3961, 4167, 4531, 6886, 6902,
6915, 6928, 6955, 7058, 8544, 8546
\stex_if_smsmode_p: ..... 135, 2134
\stex_ignore_spaces_and_pars: ...
. 142, 357, 357, 360, 2572, 2573, 8478
\l_stex_import_archive_str . 130,
2474, 2479, 2811, 3056, 3081, 3085,
3105, 3106, 3107, 3275, 3302, 3318
\stex_import_module_uri:nn .....
..... 130, 2472, 2808,
3054, 3072, 3072, 3273, 3299, 3316
\l_stex_import_name_str .....
..... 130, 2476, 2481, 2813, 3058, 3066,
3075, 3277, 3291, 3304, 3320, 3332
\l_stex_import_ns_str .... 2482,
2485, 2815, 3061, 3065, 3148, 3151,
3154, 3167, 3170, 3173, 3181, 3279,
3282, 3285, 3290, 3322, 3326, 3331
\l_stex_import_path_str .....
..... 130, 2475, 2480,
2812, 3057, 3076, 3087, 3091, 3095,
3096, 3097, 3100, 3276, 3303, 3319
\stex_import_require_module:nnn .
..... 130, 2478, 2810,
3055, 3110, 3110, 3123, 3274, 3317
\stex_import_require_module_-
safe:nnn ..... 3125, 3301
\l_stex_import_uri_str .....
..... 130, 3086, 3107, 3113,
3118, 3128, 3133, 3141, 3143, 3147,
3148, 3154, 3160, 3162, 3166, 3167,
3173, 3192, 3199, 3219, 3254, 3255
\stex_in_archive:nn 134, 1092, 1092,
1886, 1937, 1962, 2020, 2073, 2111
\stex_in_invisible_html_bool ...
..... 3797, 4857, 5750, 5751, 5781
\l_stex_in_meta_bool 2377, 2378, 2383
\l_stex_inpararray_bool ..... 5881
\stex_input_with_hooks:n .....
.. 1021, 1888, 1891, 1912, 1929, 1931
\_stex_invoke_notation:w .....
..... 5169, 5286, 5287, 5292, 5329
\stex_invoke_outer_field: .... 6172
\stex_invoke_sequence: .....
..... 5013, 5026, 5077, 5077, 5618
\stex_invoke_sequence_in: .... 132
\stex_invoke_sequence_range: ... 132
\stex_invoke_structure: .....
..... 6140, 6188, 6308
\stex_invoke_symbol ..... 131
\stex_invoke_symbol: .....
..... 125, 131, 3767,
4818, 4831, 5263, 5263, 6848, 7675
\_stex_invoke_symbol:nnnnnnnnN ...
..... 123, 131,
2615, 4005, 4019, 4020, 5204, 5204,
5217, 5253, 6714, 6728, 6736, 6741
\stex_invoke_symbol:nnnnnnnn\l_-
stex_current_symbol_str .... 131
\stex_invoke_text_symbol: 3936, 3984
\stex_invoke_variable:nnnnnn . 5238
\stex_invoke_variable:nnnnnnN ..
..... 131, 4824, 5019, 5238, 5616
\stex_is_sequentialized:n .....
..... 5610, 5637, 5712, 5722

```

```

\stex_iterate_break: 123, 2671, 2674
\stex_iterate_break:n . 123, 2671,
   2677, 2754, 3435, 3445, 3460, 4093
\stex_iterate_morphisms:nn .....
   ..... 2737, 2737, 3413, 3429
\stex_iterate_notations:nn .....
   ..... 126, 2712, 2712, 2933, 6748
\stex_iterate_symbols:n .....
   ..... 123, 2670, 2670, 4086
\stex_iterate_symbols:nn .....
   ..... 123, 2687, 2687, 2896, 3463, 6156, 6699
\l_stex_key_answerclass_str .....
   ..... 8585, 8589, 8603, 8638, 8639
\l_stex_key_archive_str .....
   ..... 132, 385, 388, 1569, 1587,
   1601, 1610, 1611, 1688, 1695, 1709
\l_stex_key_argnames_clist .... 124
\l_stex_key_args_str .....
   ..... 124, 3686, 3691, 3700, 3736,
   3780, 3833, 3838, 3842, 3846, 3850,
   3854, 3858, 3862, 3866, 3870, 3889,
   4506, 4839, 4970, 4971, 5036, 6805
\l_stex_key_argtypes_clist .....
   ..... 3705, 3710, 3813, 3815,
   3902, 4872, 4875, 5065, 5068, 6809
\l_stex_key_assoc_str 3689, 3694,
   3787, 3788, 4843, 4844, 5040, 5041
\l_stex_key_autogradable_bool ...
   ..... 8367, 8373, 8378, 8431, 8530
\l_stex_key_continues_tl . 7112, 7121
\l_stex_key_def_tl 125, 3702, 3714,
   3735, 3764, 3801, 3802, 3898, 3909,
   3913, 3933, 4505, 4815, 4828, 4861,
   4862, 5010, 5023, 5054, 5055, 6807
\l_stex_key_deprecate_str .....
   ..... 411, 413, 432, 433
\l_stex_key_due_tl .....
   ..... 9170, 9174, 9209, 9210
\l_stex_key_feedback_str .....
   ..... 8814, 8836, 8837, 8846, 8847
\l_stex_key_feedback_tl ... 8811,
   8907, 8912, 8932, 8933, 9010, 9011
\l_stex_key_file_str .....
   ..... 132, 386, 389, 1570,
   1586, 1589, 1606, 1621, 1649, 1664,
   1675, 1688, 1700, 1704, 1710, 1782
\l_stex_key_for_clist 2844, 6804,
   6812, 6822, 7109, 7118, 7390, 7395
\l_stex_key_for_str .....
   ..... 6892
\l_stex_key_from_tl .....
   ..... 7110, 7119
\l_stex_key_Ftext_tl .....
   ..... 8910, 8916, 8930, 9008
\l_stex_key_functions_tl . 7114, 7122
\l_stex_key_given_tl .....
   ..... 9169, 9173, 9206, 9207
\l_stex_key_hide_bool 7116, 7125, 7258
\l_stex_key_id_str .....
   ..... 132, 393, 395, 438,
   439, 6873, 6874, 6941, 6943, 7336,
   7345, 7434, 8594, 8596, 8602, 8658,
   8660, 8666, 8707, 8709, 8715, 8758,
   8760, 8767, 8819, 8821, 8829, 8832
\l_stex_key_intent_args_clist ...
   ..... 4137, 4143, 4309, 4318, 4320
\l_stex_key_intent_str .....
   ..... 3946, 4136, 4142, 4232, 4312
\l_stex_key_macroname_str .....
   .. 6803, 6813, 6830, 6832, 6841, 6845
\l_stex_key_method_tl .....
   .. 7115, 7124, 7245, 7246, 7392, 7396
\l_stex_key_mhrepos_str .....
   .. 8372, 8380, 8568, 8570, 8578, 9222
\l_stex_key_min_tl .....
   ..... 8370, 8376, 8445, 8542, 8561
\l_stex_key_name_str .....
   ..... 124, 125, 3699, 3707,
   3732, 3733, 3747, 3748, 3761, 3770,
   3779, 3823, 3878, 3907, 3911, 3920,
   3921, 3923, 3931, 3939, 3949, 3957,
   3963, 3964, 4479, 4502, 4503, 4521,
   4523, 4774, 4775, 4785, 4792, 4795,
   4810, 4812, 4825, 4838, 4887, 4890,
   4891, 4895, 4897, 4898, 4967, 4968,
   4990, 5005, 5007, 5020, 5035, 6802,
   6811, 6831, 6832, 6836, 6839, 6845,
   6853, 6862, 6891, 7391, 7398, 7414,
   7415, 7428, 7429, 7430, 7659, 7660,
   7669, 7681, 8371, 8377, 8424, 8426,
   8429, 8518, 8520, 8523, 8525, 8528
\l_stex_key_number_tl .....
   ..... 9168, 9172, 9188, 9189
\l_stex_key_op_tl .....
   .. 3945, 4135, 4141, 4193, 4194,
   4195, 4202, 4203, 4210, 4215, 4346,
   4380, 4384, 4442, 4447, 4464, 4469,
   4472, 4894, 4896, 4899, 4977, 4979
\l_stex_key_post_tl .. 5927, 5931,
   5948, 5959, 5992, 6005, 7014, 7022
\l_stex_key_pre_tl .. 5926, 5930,
   5948, 5959, 5992, 6005, 7014, 7022
\l_stex_key_prec_str .. 126, 3947,
   4134, 4140, 4244, 4247, 4250, 4256,
   4259, 4262, 4265, 4271, 4283, 4284
\l_stex_key_proofend_tl .....
   ..... 7111, 7120, 7222, 7223
\l_stex_key_proot_t1 .....
   ..... 5928

```

```

\l_stex_key_pts_str ..... 132, 133,
.. 8810, 8813, 8833, 8834, 8843, 8844
\l_stex_key_pts_tl ..... 8369,
8375, 8437, 8438, 8444, 8539, 8556
\l_stex_key_reorder_str 3688, 3692,
3790, 3791, 4849, 4850, 5046, 5047
\l_stex_key_return_tl ..... 3703, 3709, 3766, 3804, 3807,
3900, 4786, 4817, 4830, 4864, 4867,
4975, 5012, 5025, 5057, 5060, 6808
\l_stex_key_role_str ..... 3687, 3695, 3793, 3794,
3925, 4846, 4847, 5043, 5044, 6842
\l_stex_key_root_tl ..... 5932
\l_stex_key_short_tl ..... 1313, 1316, 1343, 1345
\l_stex_key_sig_str ..... 122, 2304, 2346, 2494, 2509, 2526
\l_stex_key_style_clist 133, 405,
407, 457, 461, 508, 512, 6876, 6877,
6970, 8868, 8869, 8954, 8968, 8969
\l_stex_key_T_bool ..... 8908,
8913, 8914, 8927, 8940, 9005, 9018
\l_stex_key_term_tl ..... 7113, 7123, 7242, 7243, 7393, 7397
\l_stex_key_testspace_dim ..... 8583, 8586,
8588, 8617, 9052, 9054, 9115, 9118
\l_stex_key_title_str ..... 6814
\l_stex_key_title_tl ..... 399, 401, 1571, 1749, 1750,
2516, 6890, 6895, 8433, 8434, 8440,
8533, 8534, 8637, 8638, 8698, 8699,
8747, 8748, 8800, 8801, 9201, 9202
\l_stex_key_Ttext_tl ..... 8909, 8915, 8928, 9006
\l_stex_key_type_tl ..... 125,
3701, 3713, 3734, 3765, 3798, 3799,
3896, 3908, 3912, 4504, 4816, 4829,
4858, 4859, 5051, 5052, 6806, 6815
\l_stex_key_variant_str ..... 126, 4133, 4139, 4146,
4148, 4177, 4230, 4343, 4352, 4384,
4489, 4508, 4797, 4888, 4896, 4992
\l_stex_key_wikidata_str ..... 3704, 3711, 3784, 3785
\stex_keys_define:nnnn ..... 132, 365, 365,
384, 392, 398, 404, 410, 416, 1342,
1555, 1561, 2493, 3685, 3698, 3906,
4132, 4499, 4765, 5925, 6115, 6801,
7108, 7389, 7965, 8368, 8567, 8584,
8654, 8809, 8906, 9050, 9167, 9266
\stex_keys_set:nn ..... 132, 133,
380, 380, 1349, 1573, 1774, 1779,
2515, 3722, 3917, 3918, 4153, 4477,
4518, 4772, 4965, 5937, 5945, 5956,
5978, 5989, 6002, 6130, 6883, 6912,
7006, 7012, 7020, 7198, 7269, 7324,
7427, 7623, 7657, 8045, 8415, 8419,
8508, 8577, 8593, 8616, 8657, 8706,
8757, 8818, 8863, 8925, 8963, 9003,
9094, 9103, 9182, 9186, 9221, 9279
\stex_kpsewhich:Nn 142, 43, 43, 54, 61
\c_stex_language_abbrevs_prop ... 139, 109
\stex_language_from_file: ..... 140, 157, 157, 1004
\c_stex_languages_clist . 29, 180, 183
\c_stex_languages_prop ..... 139, 109, 137, 138, 172, 188, 805
\g_stex_last_feature_str ..... 128, 2794, 6156
\stex_macro_body:N . 141, 543, 545, 563
\stex_macro_definition:N 141, 558, 558
\l_stex_mroname_str ..... 125, 227, 2784,
2785, 3720, 3724, 3726, 3760, 3781,
3782, 3919, 3930, 4519, 4773, 4811,
4823, 4840, 4841, 4966, 5006, 5018,
5037, 5038, 6141, 6841, 7658, 7668
\_stex_main_archive: .... 1198, 1237
\c_stex_main_archive_prop . 134, 1198
\c_stex_main_file ..... 137, 138, 974, 1013, 1078, 7735
\stex_map_args:N ..... 3808, 4107, 4107, 4209, 4305,
4376, 4551, 4868, 5061, 5424, 6170
\stex_map_notation_args:N ..... 4107, 4120, 4409
\stex_map_uri:Nnnnn 138, 811, 818, 2361
\c_stex_mathhub_file 138, 918, 1042,
1117, 1127, 1130, 1140, 1199, 1619,
1698, 1709, 1864, 1876, 1888, 1891,
1915, 1920, 1931, 1983, 2037, 2041,
2051, 2055, 2065, 2070, 3119, 3134
\c_stex_mathhub_main_manifest_-
prop ..... 1205, 1206
\stex_mathml_arg:nn ..... 141
\stex_mathml_intent:nn ..... 141
\_stex_maybe_brackets:nn ..... 4604, 4621, 5085, 5321, 5332, 5874
\stex_metagroup_do_in:nn ..... 123, 143, 231, 236,
243, 2555, 3367, 3401, 7665, 7666, 7677
\stex_metagroup_new:n ..... 143, 225, 226, 230, 2306, 2355, 2839

```

```

\l_stex_metatheory_uri 2175, 2381,
2384, 2468, 2484, 2498, 2500, 2527,
2528, 7584, 7585, 7586, 7633, 7634
\c_stex_module_ ..... 123, 125
\stex_module_add_code:n .....
123, 2558, 2558, 2561, 2563, 2573, 7641
\stex_module_add_morphism:nnn .. 122
\stex_module_add_morphism:nnnn ..
..... 130, 2579, 2579,
2584, 2943, 3281, 6212, 6268, 7644
\stex_module_add_notation:nnnnn .
..... 125, 2622,
2622, 2633, 2935, 2938, 4341, 4461
\stex_module_add_symbol:nnnnnnN .
..... 122, 2585
\stex_module_add_symbol:nnnnnnnN
.... 123, 2585, 2957, 2961, 2964,
3759, 3929, 6136, 6168, 6844, 7667
\stex_module_setup:n .....
122, 2297, 2297, 2520, 2790, 7504, 7624
\stex_module_setup_top_nosig:n .
..... 2305, 2314, 2374, 7620
\l_stex_morphism_morphisms_seq ..
..... 129, 2872, 2882, 2977
\l_stex_morphism_renames_prop ..
..... 129, 2871, 2881,
2928, 2934, 2947, 2955, 3002, 3404
\l_stex_morphism_symbols_prop ..
..... 129, 2860,
2870, 2880, 2898, 2902, 2911, 2925,
2930, 2998, 3016, 3021, 3370, 3466
\stex_new_statement:nn ..... 6820
\stex_new_statement:nnn .....
..... 6881, 6947, 6953, 6968, 6969
\stex_new_stylable_cmd:nnnn .....
..... 133, 443, 443,
3053, 3266, 3315, 3341, 3385, 3411,
3592, 3612, 3633, 3721, 3916, 4152,
4476, 4517, 4771, 4964, 7196, 7590
\stex_new_stylable_env:nnnnnnn ..
..... 133, 443, 478, 2514, 3476,
3498, 3518, 6107, 6193, 6242, 6882,
7293, 7311, 7323, 8409, 8507, 8609,
8672, 8721, 8773, 8862, 8962, 9179
\stex_next_symbol:n .....
.. 5181, 5182, 5200, 6519, 6641, 7001
\c_stex_no_frontmatter_bool 35, 1449
\l_stex_notation_ ..... 126
\stex_notation_add: .....
.. 3951, 4157, 4329, 4340, 4527, 7684
\l_stex_notation_args_t1 .. 4121,
4199, 4233, 4304, 4310, 4316, 4525
\stex_notation_check: 4156, 4329,
4329, 4481, 4526, 4788, 4987, 7683
\stex_notation_do_html:n . 3954,
4160, 4329, 4349, 4529, 4792, 7686
\l_stex_notation_downprec . 5540,
5868, 5869, 5879, 5880, 5900, 5902
\l_stex_notation_macrocode_cs ...
..... 125–127, 4183, 4228, 4240,
4334, 4345, 4374, 4434, 4463, 4468,
4492, 4511, 4800, 4889, 4892, 4995
\stex_notation_make_args: 4184,
4329, 4335, 4408, 4493, 4512, 4801
\stex_notation_parse:n .....
..... 3950, 4155, 4192, 4192,
4480, 4524, 4787, 4978, 4981, 7682
\stex_notation_parse_and_then:nw
..... 126, 128
\stex_notation_set_default:n ...
..... 4163, 4428, 4449, 4459, 4484
\stex_notation_top:nnw ..... 128
\stex_par: ..... 8955, 8956, 8958
\stex_persist:n .....
122, 141, 616, 620, 621, 623, 626,
629, 630, 636, 637, 1192, 1217, 2403
\c_stex_persist_mode_bool .....
..... 141, 31, 607, 670, 1236
\stex_persist_read_now: .....
..... 669, 1216, 7737
\c_stex_persist_write_mode_bool .
141, 32, 613, 618, 671, 677, 1215, 2392
\stex_pseudogroup:nn ..... 142,
143, 205, 205, 286, 1096, 2660, 5413
\stex_pseudogroup_restore:N .....
..... 142, 143, 205, 208, 1109, 2664
\stex_pseudogroup_with:nn .....
..... 142, 215, 215, 819,
941, 2383, 2671, 2689, 2714, 4019,
4700, 4709, 4734, 5900, 5914, 6036
\c_stex_pwd_file .....
138, 974, 994, 1199, 1200, 1716, 1871
\stex_reactivate_macro:N 142, 348,
354, 2578, 2990, 2991, 2992, 3309,
3311, 3494, 3515, 3539, 3607, 3628,
3649, 3744, 3972, 4190, 4541, 6152,
6227, 6290, 6933, 6934, 6935, 6936,
6937, 6951, 6960, 6961, 6962, 6963,
6964, 6965, 6966, 7280, 7281, 7282,
7283, 7284, 7285, 7286, 7287, 7594,
7618, 7646, 8394, 8395, 8396, 8397,
8398, 8399, 8400, 8783, 8879, 8981
\stex_ref_new_doc_target:n .....
.. 121, 132, 439, 1504, 1517, 1541
\stex_ref_new_id:n . 1507, 1518, 6942
\stex_ref_new_sym_target:n .....
121, 1787, 1811, 1820, 6930, 6957, 6991
\stex_ref_new_sym_target:nn 121, 1818

```

```

\stex_ref_new_symbol:n .....
    ..... 121, 1787, 2428, 3769, 3938
\l_stex_ref_url_str . 1514, 1535, 1537
\_stex_renamedecl_do:nn .....
    ..... 3387, 3392, 3589
\stex_require_archive:n .....
    ..... 130, 134, 1086, 1115, 1115,
        1125, 1610, 1875, 3106, 3116, 3131
\stex_resolve_path_pair:Nnn .....
    ..... 134, 1860, 1860, 1882
\_stex_return_args:nn .....
    ..... 3773, 3808, 4868, 5061
\l_stex_return_notation_tl .....
    ..... 5221, 5398, 5399, 6357,
        6365, 6527, 6528, 6608, 6634, 6635
\stex_set_current_archive:n .....
    ... 134, 1085, 1085, 1105, 1211, 3250
\stex_set_current_namespace: ...
    ..... 139, 970, 970, 1005
\stex_set_document_uri: .....
    ..... 139, 966, 966, 1003
\stex_set_language:n .....
    ..... 140, 135, 135, 156, 173, 2507
\stex_set_notation_macro:nnnnn ..
    ..... 125, 126, 2622, 2639, 2643, 2643, 2655
\stex_sms_allow:N .....
    ..... 135, 2118,
        2118, 2129, 2130, 2131, 2132, 2133
\stex_sms_allow_env:n .....
    ..... 136, 2118, 2126, 2545, 3496, 3517,
        3541, 6150, 6224, 6286, 6907, 7655
\stex_sms_allow_escape:N .....
    ..... 136, 2118, 2122, 2492, 2577, 3308,
        3346, 3390, 3609, 3630, 3651, 3745,
        3973, 4191, 4542, 6923, 7047, 7691
\stex_sms_allow_import:Nn .....
    ..... 136, 2139, 2141, 3310
\stex_sms_allow_import_env:nn ...
    ..... 136, 2139, 2145
\g_stex_sms_import_code .....
    ..... 136, 2150, 2173, 2191, 3300
\stex_smsmode_do: .....
    ..... 136, 2164, 2166, 2200, 2205, 2490,
        2538, 2574, 3268, 3337, 3344, 3388,
        3485, 3505, 3529, 3603, 3624, 3645,
        3741, 3969, 4173, 4537, 6109, 6218,
        6275, 6900, 6921, 7044, 7647, 7688
\_stex_split_slash: .....
    ..... 5968, 5972, 6021, 6987
\_stex_sref_do_aux:n .....
    ..... 1793, 1800, 1804, 1807, 7500
\stex_str_if_ends_with:nn .. 136, 533
\stex_str_if_ends_with:nnTF .....
    ..... 136, 533, 987, 3420, 3443, 4076
\stex_str_if_ends_with_p:nn .....
    ..... 136, 533, 4051, 4091
\stex_str_if_starts_with:nn 136, 538
\stex_str_if_starts_with:nnTF .....
    ..... 136, 419, 538, 2897, 6335, 6339, 7050
\stex_str_if_starts_with_p:nn ...
    ..... 136, 538
\stex_structural_feature_-
    module:nn ... 128, 2779, 2779, 6146
\stex_structural_feature_module_-
    end: 128, 2779, 2793, 6111, 6220, 6282
\stex_structural_feature_-
    morphism:nnn ... 2800
\stex_structural_feature_-
    morphism:nnnn ... 130
\stex_structural_feature_-
    morphism:nnnnn ... 129, 2801,
        3478, 3499, 3520, 3593, 3613, 3634
\stex_structural_feature_-
    morphism_check_total: .....
    ..... 2910, 3507, 3531, 3622, 3643
\stex_structural_feature_-
    morphism_end: . 129, 2800, 2867,
        3490, 3511, 3535, 3602, 3623, 3644
\stex_style_apply: .....
    ..... 133, 134, 443, 448, 483,
        488, 2536, 2541, 3068, 3293, 3334,
        3483, 3488, 3503, 3509, 3527, 3533,
        3600, 3620, 3641, 3738, 3966, 4170,
        4496, 4534, 4804, 4999, 6897, 6903,
        7208, 7278, 7298, 7315, 7335, 7353,
        7356, 7365, 8454, 8461, 8544, 8546,
        8606, 8623, 8627, 8669, 8684, 8688,
        8718, 8733, 8737, 8770, 8786, 8790,
        8887, 8889, 8989, 8991, 9194, 9197
\stex_suppress_html:n .....
    ..... 140, 285, 285, 2151, 4223
\_stex_symdecl_check_terms: ...
    ... 125, 3751, 3893, 3893, 3928, 4779
\stex_symdecl_do: .....
    ..... 124, 125, 3750, 3822,
        3822, 3927, 4778, 4973, 6843, 7662
\stex_symdecl_html: .....
    ..... 3754, 3777, 3941, 6850
\stex_symdecl_top:n .....
    ..... 125, 3728, 3746, 3746, 4520
\stex_symdef_styledefs: . 4501, 4533
\stex_term_arg:nnn .....
    ..... 5531, 5535, 5542, 5552
\stex_term_arg:nnnn .....
    ..... 4424, 4665, 4673,
        5535, 5536, 5567, 5601, 5638, 5723
\stex_term_arg_aB:nnnn .....
    ..... 4678, 4686, 4693, 5557, 5565

```

`\_stex_term_do_aB_clist`: .. 4700,  
 4701, 4710, 4714, 4735, 4740, 5562,  
 5574, 5603, 5653, 5682, 5730, 5733  
`\_stex_term_oma:nnn` .....  
 ..... 4617, 5423, 5808, 5809, 5835  
`\_stex_term_oma_or_omb:nnn` .....  
 ..... 4617, 4622, 4670, 4683  
`\_stex_term_omb:nnn` ..... 4670,  
 4683, 5451, 5452, 5837, 5838, 5864  
`\_stex_term_oms:nnn` .....  
 ..... 4384, 5208, 5210, 5744,  
 5754, 5803, 5807, 5939, 5949, 5960  
`\_stex_term_oms_or_omv:nnn` .....  
 ..... 3986, 4605, 5086, 5208,  
 5210, 5242, 5244, 5322, 5333, 5406,  
 5744, 5807, 6374, 6412, 6450, 6506  
`\_stex_term_omv:nnn` .....  
 ..... 4957, 5242, 5244,  
 5744, 5767, 5804, 5983, 5996, 6009  
`\stex_undefined`: ..... 41  
`\stex_uri_add_module:Nn` .....  
 ..... 139, 939, 939, 964, 1519, 7584  
`\stex_uri_from_current_file:Nn` ..  
 ..... 139, 929, 929, 967  
`\stex_uri_from_current_file_-_nolang:Nn` ..... 139, 929, 935, 971  
`\stex_uri_from_repo_file` ..... 139  
`\stex_uri_from_repo_file:NNNn` ...  
 ..... 138, 139, 892, 895, 931, 1624  
`\stex_uri_from_repo_file_-_nolang:NNNn` ..... 139, 892, 892, 936  
`\stex_uri_resolve:Nn` ..... 138,  
 877, 877, 880, 2484, 2500, 2504, 7503  
`\stex_uri_set:Nn` 138, 827, 873, 876, 927  
`\stex_uri_use:N` ..... 138, 860,  
 881, 885, 933, 968, 972, 1514, 1520,  
 1521, 1525, 1591, 1593, 1626, 1633,  
 1650, 1651, 2384, 2528, 3145, 3150,  
 3151, 3164, 3169, 3170, 6874, 7634  
`\_stex_vardecl_notation_macro`: ..  
 ... 125, 4482, 4789, 4885, 4885, 4988  
`\_stex_variable:nnnnnnN` . 4764, 4905  
`\l_stex_variables_prop` .....  
 ..... 125, 4762, 4810, 4917, 5005  
 stex internal commands:  
`\_stex_annotation_env_str` ... 300, 301  
`\_stex_aux_apply_patch:n` .. 449, 456  
`\_stex_aux_apply_patch_begin:n` ..  
 ..... 484, 507  
`\_stex_aux_apply_patch_end:n` ..  
 ..... 489, 522  
`\_stex_aux_args:n` . 547, 551, 554, 557  
`\_stex_aux_end`: ..... 543, 544, 547  
`\_stex_aux_params:n` 561, 592, 599, 602  
`\_stex_aux_patch:nnn` ..... 445, 470  
`\_stex_aux_patch:nnnn` ..... 480, 497  
`\_stex_aux_prefix:n` .. 559, 567, 581  
`\_stex_aux_prefix_long:n` .....  
 ..... 573, 577, 583, 590  
`\_stex_aux_split_at_bracket:w` ..  
 ..... 420, 428  
`\_stex_aux_start`: ..... 543, 546  
`\l\_stex_aux_tl` .....  
 ..... 370, 371, 372, 374, 377, 378  
`\_stex_debug_:nn` ..... 67, 70, 75  
`\l\_stex_debug_cl` ..... 83, 85, 88  
`\_stex_debug_env_str` ..... 81, 82, 85  
`\_stex_doc_check_topsect`: .....  
 ..... 1427, 1433, 1439, 1445  
`\_stex_doc_do_section:n` .....  
 ..... 1325, 1329, 1333, 1350  
`\_stex_doc_maketitle`: ... 1268, 1273  
`\_stex_doc_orig_backmatter` ....  
 ..... 1461, 1464, 1482  
`\_stex_doc_orig_frontmatter` ....  
 ..... 1451, 1454, 1471  
`\_stex_doc_set_title:n` .. 1245, 1261  
`\_stex_doc_skip_fragment:n` ....  
 ..... 1390, 1396,  
 1400, 1404, 1405, 1406, 1407, 1408  
`\_stex_doc_skip_section`: .....  
 ..... 1358, 1380, 1386  
`\_stex_doc_skip_section_i`: ....  
 ..... 1367, 1370, 1373, 1381, 1386  
`\_stex_doc_title_html`: .....  
 ..... 1250, 1253, 1265  
`\g\_stex_doc_title_tl` .....  
 .. 1240, 1242, 1249, 1255, 1260, 1263  
`\_stex_expr_aB_arg:nnnnn` 5572, 5579  
`\_stex_expr_aB_simple_arg:nnnnn`  
 ..... 5590, 5599  
`\_stex_expr_add_prop_arg:nw` ...  
 ..... 5419, 5443, 5446  
`\_stex_expr_arg:n` ..... 5420, 5456  
`\l\_stex_expr_arg_counter_int` ...  
 ..... 5410, 5417, 5432, 5468, 5469  
`\_stex_expr_arg_do:nnn` .....  
 ..... 5470, 5476, 5492, 5527, 5534  
`\_stex_expr_arg_inner:nn` .....  
 ..... 5459, 5462, 5466, 5472  
`\_stex_expr_assoc_make_seq:nnn` ..  
 ..... 5634, 5663, 5742  
`\_stex_expr_assoc_seq:nnnnnnn` ..  
 ..... 5583, 5631  
`\_stex_expr_check:n` ..... 5500  
`\_stex_expr_check:nTF` ... 5469, 5475  
`\_stex_expr_check_b:nn` .. 5424, 5449

```

\l__stex_expr_count_int .....
    .. 5564, 5570, 5580, 5601, 5638, 5723
\l__stex_expr_cs .... 5665, 5668, 5688
\l__stex_expr_customs_prop .....
    ..... 5415, 5427, 5428,
    5429, 5444, 5480, 5486, 5501, 5504
\l__stex_expr_customs_seq .....
    .. 5416, 5433, 5434, 5435, 5445, 5502
\l__stex_expr_do_aB_clist: .....
    ..... 5557, 5562, 5603, 5682
\l__stex_expr_do_ab_next:nnn .....
    ..... 5423, 5425, 5451, 5452
\l__stex_expr_do_headterm:nn .....
    ..... 5732, 5764, 5777, 5780, 5805
\l__stex_expr_do_ref:nNn ... 5939,
    5947, 5958, 5980, 5991, 6004, 6014
\l__stex_expr_do_seqmap:nnnnnn ...
    ..... 5588, 5695
\l__stex_expr_end: ..... 5585, 5595
\l__stex_expr_gobble:nnnnnnnn ...
    ..... 5585, 5595
\l__stex_expr_iarg_t1 5675, 5677, 5688
\l__stex_expr_invoke_custom:n ...
    ..... 5270, 5284, 5411
\l__stex_expr_invoke_math: 5265, 5273
\l__stex_expr_invoke_op_custom:n ...
    ..... 5270, 5277, 5403
\l__stex_expr_invoke_op_notation:w ...
    ..... 5279, 5280, 5318
\l__stex_expr_invoke_return: .....
    ..... 5359, 5362
\l__stex_expr_invoke_return_-
    maybe:n ..... 5301, 5311, 5340
\l__stex_expr_invoke_return_next: .....
    ..... 5347, 5356
\l__stex_expr_invoke_text: 5265, 5268
\l__stex_expr_is_seqmap:n .... 5624
\l__stex_expr_is_seqmap:nTF ... 5587
\l__stex_expr_is_varseq:n .... 5613
\l__stex_expr_is_varseq:nTF 5581, 5700
\l__stex_expr_left_bracket_str ..
    ..... 5870, 5896, 5904, 5914, 5915
\l__stex_expr_old_seq .....
    ..... 5707, 5710, 5714, 5719
\l__stex_expr_reset_t1 .....
    ..... 5181, 5184, 5188, 5189, 5195
\l__stex_expr_ret_cs .....
    ..... 5344, 5349, 5376, 5381, 5386
\l__stex_expr_return_arg:n 5346, 5352
\l__stex_expr_return_args_t1 ...
    .. 5341, 5353, 5358, 5367, 5383, 5388
\l__stex_expr_return_notation:n ...
    ..... 5364, 5397
\l__stex_expr_return_this_t1 ...
    ..... 5342, 5358,
    5363, 5367, 5371, 5372, 5382, 5390
\l__stex_expr_right_bracket_str ...
    ..... 5871, 5897, 5909, 5914, 5916
\l__stex_expr_setup:nnnnn ...
    ..... 5207, 5219, 5241
\l__stex_expr_varseq_in_map:nnnnnnn ...
    ..... 5702, 5741
\l__stex_features_add_definiens:nn ...
    ..... 2851, 2994
\l__stex_features_break: ... 2852, 2856
\l__stex_features_check_break:nnnnnnnn ...
    ..... 3015, 3020, 3029, 3032, 3039
\l__stex_features_clean:nnw 2891, 2899
\l__stex_features_do_decls: 2883, 2895
\l__stex_features_do_elaboration: ...
    ..... 2873, 2923
\l__stex_features_do_for_list: ...
    ..... 2842, 2993
\l__stex_features_do_morph:nnnn ...
    ..... 2971, 2975
\l__stex_features_do_morphisms:n ...
    ..... 2884, 2969, 2979
\l__stex_features_elab_check: ...
    ..... 2931, 2954
\l__stex_features_feature_str ...
    ..... 2834, 2946
\l__stex_features_get_check:nnnn ...
    ..... 2999, 3027
\l__stex_features_implicit_bool ...
    ..... 2800, 2819, 2822, 2959
\l__stex_features_reactivate: ...
    ..... 2837, 2982
\l__stex_features_rename:nn 2947, 2950
\l__stex_features_rename_all: .. 2887
\l__stex_features_renamed_-
    check:nnnn ..... 3003, 3013
\l__stex_features_set_definiens_-
    macros: ..... 2852, 2856
\l__stex_features_set_definiens_-
    macros_i:nnnnnnn ..... 2859, 2863
\l__stex_features_setup: ... 2836, 2879
\l__stex_features_split_qm:w ...
    ..... 2922, 2940
\l__stex_features_tmp ...
    ..... 2934, 2936, 2955, 2956, 2957
\l__stex_features_total_check: ...
    ..... 2912, 2916
\l__stex_groups_do: .... 256, 263, 272
\l__stex_groups_do_in:nn 238, 245, 266
\l__stex_groups_exists:n ..... 231
\l__stex_groups_exists:nTF ..... 237
\l__stex_groups_ids_seq 225, 228, 264

```

```

\__stex_groups_tmp .... 247, 254, 260
\l__stex_importmodule_archive_-
    str 3112, 3117, 3127, 3132, 3249, 3250
\__stex_importmodule_check_-
    file:nn ..... 3193,
    3194, 3195, 3196, 3197, 3198, 3220,
    3221, 3222, 3223, 3224, 3225, 3259
\__stex_importmodule_get_from_-
    file:nnn ..... 3153, 3189
\__stex_importmodule_get_from_-
    file_safe:nnn ..... 3172, 3216
\__stex_importmodule_get_-
    module:nnn ..... 3114, 3120, 3178
\__stex_importmodule_get_module_-
    safe:nnn ..... 3129, 3135, 3184
\__stex_importmodule_get_module_-
    uri:nnn ..... 3139, 3180
\__stex_importmodule_get_module_-
    uri_safe:nnn ..... 3158, 3186
\__stex_importmodule_import_-
    module:nn ..... 3267, 3272, 3312
\__stex_importmodule_import_-
    module_presms:nn ..... 3298, 3312
\__stex_importmodule_load_file:n
    ..... 3210, 3213, 3237, 3242, 3247
\l__stex_importmodule_path_seq ..
    ..... 3089, 3090, 3092
\__stex_importmodule_seq .....
    ..... 3074, 3075, 3076
\l__stex_importmodule_seq .....
    .. 3096, 3098, 3190, 3191, 3217, 3218
\l__stex_importmodule_str .....
    ..... 3119, 3120, 3134,
    3135, 3191, 3207, 3218, 3233, 3236,
    3241, 3248, 3252, 3260, 3261, 3263
\__stex_inputs_bibresource:n ...
    ..... 1948, 1960, 1962
\l__stex_inputs_gin_repo_str ...
    ..... 2034, 2064, 2068, 2073
\l__stex_inputs_id_seq .....
    ..... 1984, 1986, 1993
\l__stex_inputs_id_str ... 1979, 1984
\__stex_inputs_inputref:nn 1936, 1946
\__stex_inputs_inputref_html:nn .
    ..... 1901, 1939
\__stex_inputs_inputref_pdf:nn ..
    ..... 1925, 1940
\__stex_inputs_libinput:n .....
    ..... 2005, 2018, 2020
\l__stex_inputs_libinput_files_-
    seq ..... 1950,
    1953, 1982, 1989, 1990, 1999, 2000,
    2007, 2010, 2025, 2026, 2097, 2098
\__stex_inputs_mhinput:nn 1885, 1898
\l__stex_inputs_path_seq .....
    ..... 1983, 1987, 1994, 1997
\l__stex_inputs_path_str .....
    ..... 1987, 1988, 1989, 1990,
    1993, 1994, 1997, 1998, 1999, 2000
\l__stex_inputs_tmp_str .. 2026, 2028
\__stex_inputs_up_archive:nn ...
    ..... 1949, 1975, 2006, 2024, 2096
\__stex_inputs_usetikzlibrary:n .
    ..... 2095, 2109, 2111
\__stex_inputs_usetikzlibrary_-
    i:nn ..... 2079, 2098
\l__stex_iterate_continue_bool ..
    ..... 2739, 2742, 2755, 2775
\__stex_iterate_it_decl_check:nnnn
    ..... 2700, 2707
\__stex_iterate_it_decl_i:n .....
    ..... 2692, 2696, 2709
\__stex_iterate_it_not_check:nnnn
    ..... 2728, 2732
\__stex_iterate_it_not_i:n .....
    ..... 2717, 2721, 2734
\__stex_iterate_iterate_morphism:nn
    ..... 2745, 2749, 2758, 2761
\l__stex_iterate_mods_seq .....
    ..... 2688, 2697, 2698,
    2713, 2722, 2723, 2738, 2763, 2764
\__stex_iterate_morphism_cs:nnnn
    ..... 2740, 2766
\__stex_iterate_not_cs:nnnnn ...
    ..... 2714, 2715, 2725
\__stex_iterate_sym_cs:nnnnnnnnN
    .. 2671, 2672, 2682, 2689, 2690, 2703
\l__stex_iterate_todo_tl .....
    ..... 2744, 2748, 2762, 2775
\l__stex_lang_lang_str . 138, 141, 148
\l__stex_lang_str .. 170, 171, 172, 173
\l__stex_lang_turkish_bool .....
    ..... 181, 186, 196
\l__stex_mathhub_bool .... 1138,
    1139, 1141, 1144, 1154, 1156, 1165
\__stex_mathhub_check_manifest: .
    ..... 1143, 1152
\__stex_mathhub_check_manifest:n
    ..... 1153, 1155, 1157, 1162
\l__stex_mathhub_cs .. 1093, 1094,
    1097, 1100, 1102, 1106, 1110, 1111
\__stex_mathhub_do_manifest:n ...
    ..... 1121, 1126
\__stex_mathhub_find_manifest:n .
    ..... 1127, 1135, 1150, 1200
\l__stex_mathhub_key 1177, 1178, 1180
\c__stex_mathhub_manifest_ior ...
    ..... 1170, 1172, 1174, 1189

```

```

\l__stex_mathhub_manifest_str . . .
    ..... 1128, 1136, 1166, 1172, 1201
\l__stex_mathhub_parse_manifest:n
    ..... 1132, 1171, 1204
\l__stex_mathhub_prop . . .
    ..... 1173, 1181, 1182,
        1183, 1184, 1185, 1190, 1191, 1194
\l__stex_mathhub_seq . . .
    ..... 1137, 1140, 1145,
        1163, 1164, 1166, 1176, 1177, 1179
\l__stex_mathhub_str . . .
    1044, 1046, 1048, 1051, 1052, 1056,
    1057, 1062, 1068, 1071, 1075, 1078,
    1208, 1209, 1211, 1218, 1222, 1225
\l__stex_mathhub_tl . . .
    1145
\l__stex_mathhub_val . . .
    .. 1179, 1181, 1182, 1183, 1184, 1185
\l__stex_module_setup_end: 2352, 2373
\l__stex_module_setup_get_uri_-
    str:n . . .
        2302, 2359
\l__stex_module_setup_load_meta: ..
    ..... 2311, 2380
\l__stex_module_setup_load_sig: ..
    ..... 2332, 2342
\l__stex_module_setup_ns_str . . .
    .. 2303, 2305, 2360, 2362, 2369
\l__stex_module_setup_seg . . .
    ..... 2365, 2366, 2367
\l__stex_module_setup_seq . . .
    ..... 2364, 2365, 2367, 2369
\l__stex_module_setup_setup_-
    nested:n . . .
        2298, 2350
\l__stex_module_setup_setup_top:n
    ..... 2298, 2301
\l__stex_module_setup_setup_top_-
    sig:n . . .
        2305, 2327
\l__stex_module_setup_sigfile . . .
    ..... 2343, 2344, 2346
\l__stex_module_setup_split_-
    module:n . . .
        2352, 2373
\l__stex_modules_activate_-
    i:nnnnnnnn . . .
        2606, 2610
\l__stex_modules_activate_not:nn . . .
    ..... 2630, 2636
\l__stex_modules_activate_sym:n . . .
    ..... 2598, 2603
\l__stex_modules_export:n . 2569, 2571
\l__stex_modules_persist_module: .
    ..... 2392, 2402, 7498
\l__stex_modules_persist_nots_-
    i:nn . . .
        2418, 2439
\l__stex_modules_restore_mod_str . . .
    ..... 2432, 2460
\l__stex_modules_restore_module:nnnn
    ..... 2404, 2423
\l__stex_modules_restore_nots:n . .
    ..... 2436, 2445
\l__stex_modules_restore_nots_i:n
    ..... 2446, 2449, 2466
\l__stex_modules_restore_nots_-
    ii:nnnnm . . .
        2453, 2457
\l__stex_modules_set_metatheory:nn
    ..... 2469, 2489
\l__stex_modules_tl . . .
    2425, 2426
\l__stex_modules_tl . . .
    2458, 2463
\l__stex_morphisms_ass_tl . . .
    ..... 3545, 3561, 3565, 3575, 3576
\l__stex_morphisms_do_morph_-
    assign:nnn . . .
        3418, 3421, 3459
\l__stex_morphisms_do_parsed_-
    assign: . . .
        3567, 3570
\l__stex_morphisms_do_parsed_-
    newname: . . .
        3573, 3580
\l__stex_morphisms_do_parsed_-
    newname:w . . .
        3582, 3584, 3588
\l__stex_morphisms_end: . . .
    3573, 3588
\l__stex_morphisms_morphism_dom_-
    str . . .
        3412, 3425, 3437, 3447, 3461
\l__stex_morphisms_name_str . . .
    ..... 3543, 3552,
        3553, 3554, 3558, 3559, 3560, 3571
\l__stex_morphisms_newname_str . . .
    ..... 3544, 3563, 3564, 3572, 3573
\l__stex_morphisms_next_tl . . .
    ..... 3550, 3555, 3557
\l__stex_morphisms_parse_assign:n
    ..... 3542, 3595, 3615, 3636
\l__stex_morphisms_seq . . .
    ..... 3547, 3548, 3550, 3552,
        3555, 3557, 3559, 3561, 3563, 3565
\l__stex_notations_add:nnnnn . . .
    ..... 4651, 4656, 4660
\l__stex_notations_add_last:nnnnn
    ..... 4644, 4655
\l__stex_notations_add_missing_-
    args:nn . . .
        4209, 4324
\l__stex_notations_add_next:nnnnnn
    ..... 4646, 4650
\l__stex_notations_after_tl . . .
    ..... 4630, 4657
\l__stex_notations_args_end: . . .
    ..... 4110, 4113, 4116,
        4123, 4126, 4129, 4639, 4642, 4652
\l__stex_notations_args_tl . . .
    ..... 4546, 4550, 4563
\l__stex_notations_augment_arg:nn
    ..... 4551, 4568

```

```

\__stex_notations_check_aB_-
    arg:Nn ..... 4691, 4699, 4708, 4733
\l__stex_notations_clist_count_-
    int ..... 4731, 4739, 4745
\l__stex_notations_code_tl .....
    ... 4587, 4602, 4610, 4618, 4632,
        4633, 4636, 4664, 4672, 4677, 4685
\__stex_notations_complex:nnnnnn
    ..... 4596, 4614
\__stex_notations_const_precs: ...
    ..... 4201, 4243
\l__stex_notations_cs .....
    ..... 4603, 4608, 4619, 4628
\__stex_notations_do_argname:nn ...
    ..... 4305, 4308
\__stex_notations_do_argnames: ...
    ..... 4279, 4303
\__stex_notations_fun_precs: ...
    ..... 4206, 4255
\__stex_notations_make_arg:nnnn ...
    ..... 4409, 4413
\__stex_notations_make_arg_-
    html:nn ..... 4376, 4391
\__stex_notations_make_name: ...
    ..... 4548, 4572, 4580
\__stex_notations_map_args_i:w ...
    ..... 4109, 4113, 4116
\__stex_notations_map_args_ii:w ...
    ..... 4122, 4126, 4129
\__stex_notations_map_cs: ...
    ..... 4362, 4364,
        4711, 4713, 4721, 4736, 4738, 4749
\l__stex_notations_missing_str ..
    ..... 4207, 4325
\l__stex_notations_missing_tl ...
    ..... 4208, 4237, 4326
\l__stex_notations_name_str ...
    ..... 4574, 4575, 4576, 4581
\l__stex_notations_opprec_tl 4231,
    4245, 4248, 4250, 4257, 4260, 4262,
    4266, 4273, 4286, 4292, 4298, 4353
\__stex_notations_parse_notation_-
    args:nnnnw ..... 4639, 4642, 4652
\__stex_notations_parse_precs: ...
    ..... 4277, 4282
\l__stex_notations_pre_tl .....
    ..... 4617, 4632, 4669, 4682
\l__stex_notations_precs_seq ...
    ..... 4198, 4268,
        4275, 4296, 4298, 4311, 4317, 4354
\__stex_notations_process_-
    notation:nnnnnn ..... 4586, 4592
\l__stex_notations_seq .....
    ..... 4284, 4285, 4287, 4288, 4295
\__stex_notations_simple:nnnnn ...
    ..... 4594, 4600
\l__stex_notations_str .....
    ... 4285, 4286, 4287, 4289, 4295, 4296
\__stex_notations_styledefs: ...
    ..... 4169, 4176
\__stex_path_: .....
    ..... 688, 698
\l__stex_path_a_seq .....
    769, 775, 781
\l__stex_path_a_tl .....
    781, 783
\__stex_path_auth:n .....
    ..... 138, 811, 814, 815,
        816, 819, 820, 865, 881, 886, 940, 942
\l__stex_path_auth_str .....
    ..... 833, 841, 846, 851
\l__stex_path_b_seq 770, 776, 782, 788
\l__stex_path_b_tl .....
    782, 783
\__stex_path_canonicalize:N ...
    ..... 701, 712, 718
\__stex_path_colonslash 827, 832, 837
\l__stex_path_do_hooks_pre_tl ...
    ..... 1022, 1023, 1035, 1038
\__stex_path_dodots:n .. 723, 727, 733
\l__stex_path_file . 900, 906, 912, 917
\__stex_path_from_repo_file:NNNNn
    ..... 893, 896, 899
\l__stex_path_maybein_str .....
    ..... 750, 751, 752
\l__stex_path_mod .....
    844, 847, 852
\__stex_path_module:n .....
    138,
        815, 819, 822, 867, 881, 942, 944, 946
\l__stex_path_name .....
    849, 852
\__stex_path_name:n .....
    138,
        816, 819, 823, 868, 881, 942, 951, 953
\l__stex_path_path .....
    ..... 838, 839, 842, 847, 852, 921, 922
\__stex_path_path:n .....
    ..... 138, 814, 819, 821, 866, 881, 942
\__stex_path_relativize:N .. 909, 916
\l__stex_path_return_tl .....
    ..... 771, 777, 784, 787, 789
\l__stex_path_seq 721, 724, 729, 737,
    738, 741, 794, 795, 796, 801, 802,
    804, 806, 809, 917, 919, 923, 925, 927
\g__stex_path_stack .....
    ..... 984, 998, 1009, 1010, 1012, 1015
\l__stex_path_str .....
    689, 690,
        691, 694, 696, 697, 698, 700, 703,
        710, 711, 720, 722, 726, 738, 793,
        794, 795, 800, 801, 802, 804, 805,
        806, 975, 977, 979, 1010, 1015, 1016
\l__stex_path_tl .....
    919, 923, 925
\l__stex_path_uri .....
    ..... 902, 903, 904, 909, 927
\__stex_path_uri_set:NnN 828, 874, 878

```

```

\__stex_path_uri_set:Nnnnn . . .
    .... 841, 846, 851, 863, 871, 905, 911
\__stex_path_uri_use:w . . . . . 881, 887
\l__stex_path_win_drive . . .
    .... . . . . . 690, 695, 702, 704
\__stex_path_win_take:w . . . . . 688, 698
\__stex_persist_env_str . . .
    .... . . . . . 604, 605, 606, 610, 611, 612
\__stex_persist_load_file:n . . .
    .... . . . . . 633, 665, 674
\__stex_persist_read_and_write: . .
    .... . . . . . 650, 672
\c__stex_persist_sms_iow . . .
    .... . . . . . 616, 624, 646, 647, 662
\__stex_persist_write_only: . .
    .... . . . . . 645, 659, 666, 677
\__stex_proof_add_counter: 7171, 7352
\__stex_proof_begin_proof:nn . .
    .... . . . . . 7266, 7294, 7312
\l__stex_proof_counter_intarray .
    .... . . . . . 7129, 7134,
        7137, 7146, 7149, 7158, 7166, 7167,
        7175, 7180, 7187, 7193, 7267, 7268
\__stex_proof_end_list: . . .
    .. . . . . . 7236, 7307, 7368, 7378, 7444, 7467
\__stex_proof_html: . . 7240, 7262, 7418
\__stex_proof_html_env:n . . .
    .... . . . . . 7251, 7272, 7327
\l__stex_proof_in_spfblock_bool .
    .... . . . . . 7265, 7295, 7306, 7313, 7341,
        7344, 7363, 7366, 7368, 7373, 7376,
        7384, 7426, 7440, 7462, 7482, 7488
\__stex_proof_inblock_restore: . .
    .... . . . . . 7370, 7383, 7484
\__stex_proof_inc_counter: . .
    .... . . . . . 7154, 7366, 7457, 7459
\l__stex_proof_inc_counter_bool 7128
\__stex_proof_insert_number: . .
    .... . . . . . 7130, 7350, 7457, 7459
\__stex_proof_make_step_macro:Nnnnn .
    .... . . . . . 7424, 7457, 7458, 7459
\__stex_proof_number_as_string:N .
    .... . . . . . 7141, 7337, 7346, 7435
\__stex_proof_proof_box_tl 7111, 7215
\__stex_proof_remove_counter: . .
    .... . . . . . 7183, 7377
\__stex_proof_start_list:n . .
    .... . . . . . 7229, 7295, 7350, 7444, 7467
\__stex_proof_step_html:nn 7406,
    7442, 7444, 7450, 7465, 7467, 7471
\__stex_refs_add_doc_ref:nn . .
    .... . . . . . 1521, 1543, 1554
\l__stex_refs_default_archive-
    str . . . . . 1563, 1569, 1578
\l__stex_refs_default_file_str . .
    .... . . . . . 1564, 1570, 1579
\l__stex_refs_default_title_tl . .
    .... . . . . . 1565, 1571, 1580
\__stex_refs_do_autoref:n . .
    .... . . . . . 1639, 1651, 1662,
        1666, 1677, 1691, 1717, 1728, 1736
\__stex_refs_do_internal_link:nn .
    .... . . . . . 1831, 1843, 1849
\__stex_refs_do_return:nnnn . .
    .... . . . . . 1745, 1759, 1767
\__stex_refs_do_sref:nn .. 1647, 1776
\__stex_refs_do_sref_in:n . .
    .... . . . . . 1655, 1670, 1681, 1687, 1785
\__stex_refs_do_url_link:nn . .
    .... . . . . . 1837, 1851
\l__stex_refs_file . . .
    .. . . . . . 1605, 1607, 1623, 1625, 1713, 1714
\l__stex_refs_file_str . . .
    1697, 1703, 1708, 1713, 1714, 1715,
    1716, 1721, 1725, 1727, 1735, 1747
\g__stex_refs_files_seq . . .
    .... . . . . . 1504, 1544, 1549, 1596, 1658
\__stex_refs_find_uri:n . . .
    .... . . . . . 1583, 1775, 1780
\__stex_refs_find_uri_in_-
    file:nnn . . . . . 1592, 1597, 1629
\__stex_refs_find_uri_in_prop_-
    file:N . . . . . 1603, 1612, 1617
\l__stex_refs_id_str . . .
    .... . . . . . 1723, 1758, 1764, 1765
\c__stex_refs_iow . . .
    .... . . . . . 1495, 1496, 1497, 1523
\__stex_refs_new_symbol:n 1789, 1797
\l__stex_refs_prop . . . . . 1611, 1612
\__stex_refs_restore_target:nnnnn
    .... . . . . . 1724, 1756
\l__stex_refs_return_tl . . .
    .... . . . . . 1722, 1726, 1732, 1746
\__stex_refs_set_keys_b:n . . .
    .... . . . . . 1567, 1654, 1669, 1680, 1781
\l__stex_refs_str 1510, 1512, 1514,
    1519, 1521, 1526, 1799, 1801, 6943
\__stex_refs_sym_aux:nn . . .
    .... . . . . . 1828, 1841, 1846, 1857
\l__stex_refs_unnamed_counter_-
    int . . . . . 1505, 1509, 1510
\l__stex_refs_uri . . .
    .... . . . . . 1519, 1520, 1624, 1626
\l__stex_refs_uri_str . . .
    ... . . . . . 1584, 1595, 1607, 1618, 1623,
        1626, 1633, 1648, 1658, 1659, 1660,
        1661, 1662, 1665, 1666, 1668, 1674,

```

```

1676, 1677, 1679, 1689, 1690, 1691,
1718, 1729, 1737, 1757, 1762, 1763
\__stex_seqs_add: ..... 4985, 5003
\l__stex_seqs_args_t1 .... 5061, 5063
\__stex_seqs_check_terms: 4984, 5031
\l__stex_seqs_clist .....
..... 5098, 5105, 5112, 5116
\l__stex_seqs_cs .....
..... 5059, 5063, 5119, 5123,
5131, 5134, 5143, 5150, 5163, 5164
\__stex_seqs_do_all:w ... 5157, 5167
\__stex_seqs_do_first: ... 5080, 5141
\__stex_seqs_do_first_arg:n ...
..... 5139, 5146
\__stex_seqs_do_first_next: ...
..... 5148, 5153
\__stex_seqs_do_one:w ... 5155, 5161
\__stex_seqs_do_op:w ... 5079, 5083
\__stex_seqs_doop_arg:n .. 5099, 5110
\__stex_seqs_doop_range:w 5092, 5096
\l__stex_seqs_first_args_t1 ...
..... 5145, 5163, 5164, 5168, 5169
\__stex_seqs_get_index_notation:n
..... 5091, 5129, 5162
\__stex_seqs_html: ..... 4983, 5033
\__stex_seqs_macro: ..... 4986, 5017
\__stex_seqs_make_args: .. 4996, 5030
\l__stex_seqs_range_clist ...
..... 4976, 5011, 5024
\g__stex_smsemode_allowed_escape-
tl ..... 2116, 2123, 2252
\g__stex_smsemode_allowed_import-
env_seq 2140, 2146, 2186, 2235, 2238
\g__stex_smsemode_allowed_import-
tl ..... 2139, 2142, 2183, 2230
\g__stex_smsemode_allowed_tl ...
..... 2115, 2119, 2248
\g__stex_smsemode_allowedenvs_seq
..... 2117, 2127, 2257, 2260
\g__stex_smsemode_bool ...
..... 2134, 2135, 2137, 2153
\__stex_smsemode_check_begin:Nn ..
..... 2235, 2257, 2269
\__stex_smsemode_check_cs:NNn ...
..... 2208, 2216
\__stex_smsemode_check_end:Nn ...
..... 2238, 2260, 2278
\__stex_smsemode_do:w .....
..... 2202, 2214, 2216, 2218,
2240, 2250, 2262, 2275, 2282, 2285
\__stex_smsemode_do_aux:N . 2216, 2222
\__stex_smsemode_do_aux_curr:N ...
..... 2182, 2193, 2224
\__stex_smsemode_do_aux_imports:N
..... 2182, 2228
\__stex_smsemode_do_aux_normal:N .
..... 2193, 2246
\l__stex_smsemode_importmodules_-
seq .....
..... 2171
\__stex_smsemode_in_smsemode:n ...
..... 2151, 2181, 2191, 2192
\l__stex_smsemode_sigmodules_seq 2172
\__stex_smsemode_smsemode_do: ...
..... 2164, 2200
\__stex_smsemode_start_smsemode:n .
..... 2162, 2189, 2194
\__stex_statements_do_defref:nn .
..... 6978, 7007, 7013, 7021
\__stex_statements_force_id: ...
..... 6837, 6940, 6948, 6971
\__stex_statements_html_keyvals:nn
..... 6867, 6888, 6916
\__stex_statements_setup:nn ...
.. 6829, 6949, 6954, 6968, 6972, 6975
\__stex_statements_setup_def: ...
..... 6927, 6950, 6973
\l__stex_statements_uri_str ...
6852, 6856, 6857, 6862, 6863, 7033,
7036, 7039, 7041, 7086, 7089, 7092
\l__stex_structures_assigned_seq
..... 6515, 6626, 6665, 6681
\__stex_structures_begin:nn ...
..... 6108, 6129, 6206, 6261
\__stex_structures_check_-
def:nnnnnnnn .....
..... 6280, 6293
\l__stex_structures_clist ...
..... 6489, 6510, 6516, 6518
\l__stex_structures_comp_cs ...
..... 6558, 6563
\l__stex_structures_cs ...
..... 6491, 6497, 6504
\__stex_structures_current_type:
..... 6445, 6576, 6647
\l__stex_structures_current_-
type_t1 .... 6372, 6408, 6448, 6453
\__stex_structures_do_assign:nn .
..... 6392, 6396
\__stex_structures_do_assign_-
list:n .....
..... 6368, 6389
\__stex_structures_do_decl:nnnnnnnn
..... 6703, 6723
\__stex_structures_do_decl_-
nomacro:nnnnnnnn .... 6701, 6709
\__stex_structures_do_exernals:
..... 6112, 6154, 6221, 6283
\__stex_structures_end: ...
..... 6336, 6340, 6356, 6361

```

```

\l__stex_structures_exstruct_-
    name_str ..... 6256, 6261, 6305
\l__stex_structures_extend_-
    structure:nn ..... 6233, 6256
\l__stex_structures_extend_-
    structure_i:NnnnnnnnN . 6230, 6238
\l__stex_structures_external_-
    decl:nnnn ..... 6157, 6161
\l__stex_structures_extmod_str ..
    ..... 6231, 6235
\l__stex_structures_extname_-
    count ..... 6299, 6303, 6305
\l__stex_structures_field_name_-
    str ..... 6613, 6617, 6618, 6649
\l__stex_structures_fields_clist
    ..... 6369,
    6390, 6397, 6415, 6418, 6672, 6673
\l__stex_structures_get_field_-
    name:n ..... 6612, 6624
\l__stex_structures_imports_seq .
    ..... 6194, 6199, 6207,
    6244, 6249, 6263, 6785, 6788, 6791
\l__stex_structures_invocation_-
    type:n ..... 6320, 6367
\l__stex_structures_invoke_-
    field:n ..... 6590, 6622
\l__stex_structures_invoke_maybe_-
    field:nn ..... 6579, 6583
\l__stex_structures_invoke_this:n
    ..... 6329, 6571
\l__stex_structures_invoke_top:n .
    ..... 6310,
    6313, 6337, 6341, 6344, 6347, 6349
\l__stex_structures_make_mod:n ..
    ..... 6414, 6426
\l__stex_structures_make_oml:n ..
    ..... 6418, 6432
\l__stex_structures_make_oml:nn ..
    ..... 6433, 6435
\l__stex_structures_make_prop: ...
    ..... 6457, 6584, 6662
\l__stex_structures_make_prop_-
    assign: ..... 6458, 6587, 6671
\l__stex_structures_make_prop_-
    assign:nn ..... 6674, 6679
\l__stex_structures_make_prop_-
    assign_replace:nnnn ... 6682, 6688
\l__stex_structures_make_type:n ..
    ..... 6377, 6382, 6384, 6400
\l__stex_structures_maybe_-
    notation:w .... 6324, 6326, 6452
\l__stex_structures_merge:nw ...
    ..... 6318, 6334
\l__stex_structures_more_-
    nextsymbol_t1 ... 6625, 6627, 6652
\l__stex_structures_name_str 6100,
    6117, 6122, 6124, 6131, 6132, 6137,
    6138, 6143, 6144, 6147, 6163, 6169
\l__stex_structures_new_extstruct_-
    name: ..... 6243, 6301
\l__stex_structures_present: ....
    ..... 6463, 6588
\l__stex_structures_present:nn ...
    ..... 6467, 6473, 6478, 6485, 6488
\l__stex_structures_present_-
    entry:nn ..... 6493, 6499, 6514
\l__stex_structures_present_i:w ..
    ..... 6459, 6465, 6471
\l__stex_structures_present_ii:nw
    ..... 6476, 6484
\l__stex_structures_prop .....
    6503, 6615, 6623, 6655, 6663, 6680,
    6683, 6689, 6710, 6712, 6724, 6726
\l__stex_structures_prop_do_-
    decls: ..... 6667, 6698
\l__stex_structures_prop_do_-
    notations: ..... 6668, 6747
\l__stex_structures_redo_t1 ...
    .. 6642, 6666, 6692, 6739, 6750, 6765
\l__stex_structures_replace_-
    this_t1 ..... 6155
\l__stex_structures_seq .....
    ..... 6664, 6711, 6725, 6749
\l__stex_structures_set_comp_t1 .
    ..... 6309, 6358, 6362, 6520, 6637
\l__stex_structures_set_custom_-
    comp:n ..... 6363, 6552
\l__stex_structures_set_customcomp:
    ..... 6336, 6361
\l__stex_structures_set_this:n ...
    ..... 6585, 6594
\l__stex_structures_set_thiscomp:
    ..... 6309, 6546
\l__stex_structures_set_thisnotation:
    ..... 6340, 6356
\l__stex_structures_shift_-
    argls:nn ..... 6170, 6175
\l__stex_structures_this_t1 ...
    6321, 6521, 6522, 6525, 6540, 6597,
    6600, 6607, 6628, 6629, 6632, 6646
\l__stex_symdecl_add_decl: 3752, 3758
\l__stex_symdecl_args_t1 . 3808, 3810
\l__stex_symdecl_cs .....
    .. 3806, 3810, 4000, 4002, 4004, 4030
\l__stex_symdecl_do_args: . 3825, 3884
\l__stex_symdecl_env_str .....
    ..... 3658, 3659, 3660

```

`\__stex_symdecl_get_from_one_-` ..... 4039, 4084  
`\__stex_symdecl_get_symbol_from_-`  
`cs: ..... 4006, 4017`  
`\__stex_symdecl_get_symbol_from_-`  
`modules:nn ..... 4041, 4073`  
`\__stex_symdecl_get_symbol_from_-`  
`string:n ... 4007, 4009, 4012, 4034`  
`\l__stex_symdecl_name ... 4037, 4043`  
`\__stex_symdecl_parse_arity: ...`  
`..... 3824, 3830`  
`\l__stex_symdecl_seq .....`  
`..... 4036, 4037, 4038, 4042`  
`\__stex_symdecl_set_textsymdecl_-`  
`macro:nnn ..... 3957, 3975`  
`\__stex_symdecl_sym_from_str_-`  
`i:nnnn ..... 4047, 4078`  
`\__stex_symdecl_sym_i_finish:nnnnnnN`  
`..... 4053, 4060`  
`\__stex_symdecl_sym_i_gobble:nnnnnn`  
`..... 4055, 4058`  
`\__stex_vars_add: .....` 4780, 4808  
`\l__stex_vars_args_tl ... 4868, 4870`  
`\l__stex_vars_bind_bool .....`  
`..... 4763, 4766, 4768, 4852`  
`\__stex_vars_check_var:nnnnnnnnN`  
`..... 4918, 4922`  
`\l__stex_vars_cs ... 4866, 4870`  
`\__stex_vars_get_var:n .....`  
`..... 4916, 4934, 4942`  
`\__stex_vars_html: .....` 4782, 4835  
`\__stex_vars_macro: .....` 4781, 4822  
`\__stex_vars_set_vars:nnnnnnN ...`  
`..... 4903, 4924, 4927`  
`\sTeX/ComputerScience/Software ... 15`  
`stex_annotation_env (env.) ..... 141`  
`\stexcommentfont .....`  
`... 7226, 7296, 7363, 7385, 7448, 7469`  
`\stexdoctitle .....` 1241, 1300,  
`1312, 2518, 8058, 8229, 8434, 8534`  
`\STEXexport .....` 49, 86, 87, 121, 2567  
`\stexhtmlfalse .....` 324  
`\stexhtmltrue .....` 321  
`\STEXInternalAssocArgMarkerI ... 127`  
`\STEXInternalAssocArgMarkerII ... 127`  
`\STEXInternalNotation ... 4229, 4563, 4583`  
`\STEXInternalSetSrefSymURL .....`  
`..... 1500, 1801, 1804, 1807`  
`\STEXInternalSrefRestoreTarget .....`  
`..... 1498, 1524, 1724`  
`\STEXInternalSymbolAfterInvokationTL`  
`..... 127, 131`  
`\STEXInternalTermMathArgiii ... 127`  
`\STEXInternalTermMathAssocArgiiii ... 127`  
`\STEXInternalTermMathOMAiii .....` 127  
`\STEXInternalTermMathOMBiii .....` 127  
`\STEXInternalTermMathOMSOrOMViii ... 127`  
`\STEXinvisble . 81, 4326, 5201, 7276, 7331`  
`\STEXRestoreNotsEnd ... 2419, 2443, 2450`  
`\stexstyle .....` 133  
`\stexstyleassertion .....` 104  
`\stexstyleassign .....` 104  
`\stexstyleassignMorphism .....` 104  
`\stexstylecopymod .....` 104  
`\stexstylecopymodule .....` 104  
`\stexstyledefinition .....` 104  
`\stexstyleexample .....` 104  
`\stexstyleextstructure .....` 104  
`\stexstyleimportmodule .....` 104, 133  
`\stexstyleinterpretmod .....` 104  
`\stexstyleinterpretmodule .....` 104  
`\stexstylemathstructure .....` 104  
`\stexstylemcb .....` 8894  
`\stexstyleMMTinclude .....` 104  
`\stexstylemodule .....` 104, 134  
`\stexstylenotation .....` 104  
`\stexstyleparagraph .....` 104  
`\stexstyleproof .....` 104  
`\stexstylerealization .....` 104  
`\stexstylerealize .....` 104  
`\stexstylerenamedecl .....` 104  
`\stexstyleremovemodecl .....` 104  
`\stexstylerequiremodule .....` 104  
`\stexstylescb .....` 8996  
`\stexstylepsfsketch .....` 104  
`\stexstylesubproof .....` 104  
`\stexstylesymdecl .....` 104  
`\stexstylesymdef .....` 104  
`\stexstyletextsymdecl .....` 104  
`\stexstyleusemodule .....` 104  
`\stexstylevardef .....` 104  
`\stexstylevarannotation .....` 104  
`\stexstylevarseq .....` 104  
`\stopsolutions .....` 111, 8647  
`str commands:`  
`\c_backslash_str .....` 104, 696  
`\c_colon_str ... 827, 882, 1176, 2362`  
`\c_dollar_str .....` 4318  
`\c_hash_str .....` 594, 2624, 4325  
`\c_percent_str .....` 54, 1044  
`\str_case:Nn .....` 1180  
`\str_case:nn .....` 4662, 5450  
`\str_case:nnTF ... 1411, 3834, 4392,`  
`..... 4414, 5478, 5506, 7698, 7719, 7837`  
`\str_clear:N ... 385, 386, 393, 411,`  
`..... 695, 902, 1062, 1136, 1584, 1902,`  
`..... 2068, 2178, 2360, 2494, 2802, 2997,`  
`..... 3085, 3112, 3127, 3412, 3543, 3544,`  
`..... 3686, 3687, 3688, 3689, 3699, 3700,`

3704, 3724, 3907, 3946, 3947, 3997,  
 3998, 4133, 4134, 4135, 4136, 4906,  
 4933, 4941, 5538, 6117, 6185, 6802,  
 6803, 6805, 6852, 7030, 7083, 7142,  
 7391, 7622, 7693, 7714, 7966, 8268,  
 8269, 8371, 8372, 8568, 8585, 8810  
 $\backslash$ str\_const:Nn ..... 106, 7756  
 $\backslash$ str\_count:n ..... 535, 540  
 $\backslash$ str\_gset:Nn ..... 1501, 1814, 1822  
 $\backslash$ str\_gset\_eq:NN ..... 1057, 2868, 2869  
 $\backslash$ str\_if\_empty:NTF ..... 82,  
 432, 438, 605, 611, 702, 722, 909,  
 1050, 1052, 1071, 1128, 1201, 1589,  
 1595, 1601, 1648, 1649, 1664, 1675,  
 1695, 1718, 1729, 1737, 1757, 1782,  
 1848, 2304, 2547, 2784, 2807, 3001,  
 3005, 3087, 3249, 3425, 3558, 3572,  
 3659, 3747, 3781, 3784, 3787, 3790,  
 3793, 3920, 3991, 4145, 4244, 4256,  
 4265, 4312, 4774, 4840, 4843, 4846,  
 4849, 4935, 4943, 4967, 4970, 5037,  
 5040, 5043, 5046, 6038, 6121, 6131,  
 6180, 6187, 6617, 6830, 6831, 6836,  
 6853, 6873, 6941, 7036, 7089, 7336,  
 7345, 7414, 7428, 7434, 7659, 7695,  
 7716, 7827, 7886, 8049, 8424, 8518,  
 8523, 8594, 8638, 8658, 8707, 8758,  
 8800, 8819, 8833, 8836, 8843, 8846  
 $\backslash$ str\_if\_empty:nTF ..... 471, 498,  
 682, 734, 829, 1508, 1862, 2497, 3398  
 $\backslash$ str\_if\_eq:NNTF ..... 171, 783  
 $\backslash$ str\_if\_eq:nnTF .....  
 ..... 145, 161, 162, 163, 185,  
 301, 534, 539, 569, 585, 594, 606,  
 612, 697, 735, 736, 752, 1632, 1716,  
 1758, 1763, 1765, 1819, 2284, 2366,  
 2605, 2638, 2743, 3014, 3019, 3028,  
 3031, 3207, 3233, 3417, 3433, 3660,  
 4247, 4259, 4271, 4923, 4926, 5117,  
 7053, 7890, 7893, 7958, 8117, 8900  
 $\backslash$ str\_if\_eq\_p:nn ..... 4049, 4050, 4089, 4090  
 $\backslash$ str\_if\_exist:NTF ..... 232, 1845  
 $\backslash$ str\_if\_in:NnTF ..... 4325, 6019, 6985  
 $\backslash$ str\_item:Nn ..... 697, 752, 3889  
 $\backslash$ str\_lowercase:n ..... 8900  
 $\backslash$ str\_map\_break: ..... 3835  
 $\backslash$ str\_map\_break:n ..... 3836, 3840, 3844,  
 3848, 3852, 3856, 3860, 3864, 3868  
 $\backslash$ str\_map\_inline:Nn ..... 3833  
 $\backslash$ str\_new:N .....  
 ... 134, 1563, 1564, 2034, 2289, 3720  
 $\backslash$ str\_put\_right:Nn ..... 7149  
 $\backslash$ str\_range:Nnn ..... 2028  
 $\backslash$ str\_range:nnn ..... 535, 540, 574  
 $\backslash$ str\_replace\_all:Nnn ..... 696  
 $\backslash$ str\_set:Nn ..... 48,  
 56, 136, 184, 227, 689, 690, 691,  
 694, 710, 827, 1083, 1166, 1178,  
 1179, 1510, 1512, 1514, 1578, 1579,  
 1607, 1618, 1626, 1633, 1697, 1703,  
 1708, 1714, 1723, 1864, 1871, 1876,  
 1987, 1997, 2026, 2064, 2323, 2339,  
 2362, 2432, 2661, 2815, 2820, 2823,  
 2834, 2857, 2858, 3041, 3042, 3043,  
 3076, 3081, 3086, 3091, 3097, 3105,  
 3107, 3113, 3117, 3118, 3119, 3128,  
 3132, 3133, 3134, 3141, 3148, 3151,  
 3154, 3160, 3167, 3170, 3173, 3191,  
 3218, 3263, 3461, 3554, 3560, 3564,  
 3694, 3726, 3748, 3838, 3842, 3846,  
 3850, 3854, 3858, 3862, 3866, 3870,  
 3919, 3921, 3923, 3925, 4021, 4022,  
 4062, 4063, 4094, 4095, 4146, 4178,  
 4207, 4212, 4331, 4371, 4382, 4519,  
 4773, 4775, 4907, 4951, 4953, 4966,  
 4968, 4971, 5226, 5227, 5640, 5684,  
 5725, 6017, 6020, 6099, 6122, 6132,  
 6141, 6235, 6299, 6303, 6305, 6613,  
 6618, 6842, 6856, 6862, 6983, 6986,  
 7033, 7086, 7338, 7347, 7436, 7658,  
 7660, 8596, 8660, 8709, 8760, 8821  
 $\backslash$ str\_set\_eq:NN ..... 104,  
 1068, 1569, 1570, 2534, 2805, 2835,  
 3948, 3949, 4148, 4177, 4479, 4489,  
 4508, 4522, 4523, 4797, 4992, 5981,  
 5994, 6007, 6124, 6189, 6832, 6841,  
 6891, 6943, 7322, 7403, 7621, 7625,  
 7649, 7651, 7664, 7678, 7680, 7681  
 $\backslash$ str\_uppercase:n ..... 7958  
 $\backslash$ l\_tmptpa\_str ..... 158, 159, 160,  
 161, 162, 163, 164, 168, 184, 188,  
 189, 190, 192, 1902, 1903, 1904,  
 1909, 1917, 2820, 2823, 2828, 2835  
 $\backslash$ subparagraph ..... 27, 82  
subproblem (env.) ..... 8505  
subproof (env.) ..... 7322  
 $\backslash$ subproof ..... 7280, 7381  
 $\backslash$ subproofautorefname ..... 7322  
 $\backslash$ subsection ..... 26, 27, 82  
 $\backslash$ subsubsection ..... 27, 82  
 $\backslash$ svar ..... 95, 3774, 4948  
 $\backslash$ symbol ..... 89  
 $\backslash$ symdecl ..... 23,  
 24, 31, 32, 39, 40, 47, 87–89, 99,  
 104, 124, 129, 135, 142, 143, 2983,  
 3720, 7522, 7524, 7553, 7561, 7564  
 $\backslash$ symdef ..... 32, 33, 35, 40, 41, 88,  
 95, 124, 129, 135, 2985, 4499, 7506,

|                          |                                                    |
|--------------------------|----------------------------------------------------|
| \@onlypreamble           | 139, 1421                                          |
| \@rustexfalse            | 297                                                |
| \@title                  | 1270, 1271, 8228                                   |
| \activate@excursion      | 8252, 8257                                         |
| \bblobbed                | 8351, 9151                                         |
| \beamer@shortauthor      | 8215, 8217, 8232                                   |
| \beamer@shorttitle       | 8224, 8226, 8235                                   |
| \c@chapter               | 7913                                               |
| \c@framenumber           | 8240                                               |
| \c@part                  | 7925                                               |
| \compemph@uri            | 103, 131, 4368, 6027, 6035                         |
| \correction@table        | 9235                                               |
| \defemph@uri             | 103, 6035                                          |
| \define@key              | 2036, 2050, 2063                                   |
| \Gin@eheight             | 9328, 9331, 9336, 9341                             |
| \Gin@ewidth              | 8151, 8152, 9327, 9337, 9341                       |
| \Gin@exclamation         | 9327, 9328, 9336                                   |
| \Gin@mrepos              | 2037, 2040, 2044, 2065, 2069, 2074                 |
| \hwexam@bonuspts         | 9289                                               |
| \hwexam@checktime        | 9283                                               |
| \hwexam@duration         | 9259, 9262, 9268, 9273                             |
| \hwexam@kw@due           | 9210                                               |
| \hwexam@kw@forggrading   | 9251                                               |
| \hwexam@kw@given         | 9207                                               |
| \hwexam@kw@grade         | 9252                                               |
| \hwexam@kw@probs         | 9252                                               |
| \hwexam@kw@pts           | 9253                                               |
| \hwexam@kw@reached       | 9254                                               |
| \hwexam@kw@sum           | 9252                                               |
| \hwexam@kw@testemptypage | 9126                                               |
| \hwexam@min              | 9260, 9267, 9272, 9283                             |
| \hwexam@minutes@kw       | 9260                                               |
| \hwexam@reqpts           | 9269, 9274, 9286, 9290                             |
| \hwexam@tools            | 9270, 9275                                         |
| \hwexam@totalmin         | 9282, 9283                                         |
| \hwexam@totalpts         | 9281, 9290                                         |
| \if@bonuspoints          | 9285                                               |
| \if@rustex               | 305                                                |
| \itemize@inner           | 7998, 8004, 8013                                   |
| \itemize@label           | 8002, 8005, 8008                                   |
| \itemize@level           | 7996, 8001, 8004, 8013                             |
| \itemize@outer           | 7997, 8001                                         |
| \lstd@mrepos             | 2051, 2054, 2058                                   |
| \ltx@ifpackageloaded     | 140, 144, 2035, 2049, 2062, 7216, 8350, 8920, 9150 |
| \m@switch                | 5894                                               |
| \mdf@patchamsthm         | 8022                                               |
| \metakeys@show@keys      | 7999                                               |
| \notesslides@slidelabel  | 8025, 8040                                         |
| \ns@author               | 8212, 8213                                         |
| \ns@title                | 8221, 8222                                         |
| \pgf@temp                | 2080, 2081, 2082                                   |
| \pgfkeys@spdef           | 2080                                               |

```

\pgfutil@empty ..... 2082
\pgfutil@InputIfFileExists ... 2089
\prematurestop@endsfragment .....
..... 7931, 7934, 7940
\problem@kw@* ..... 8347
\problem@kw@correct .... 8928, 9006
\problem@kw@feedback .... 8847
\problem@kw@grading .... 8800
\problem@kw@hint .... 8698
\problem@kw@minutes . 8473, 8561, 8859
\problem@kw@note ..... 8747
\problem@kw@pts .... 8468, 8556, 8854
\problem@kw@solution .... 8637
\problem@kw@wrong .... 8930, 9008
\problem@restore .... 8500, 8504, 9229
\stex@backend ..... 140, 295
\symrefemph@uri .....
.... 102, 103, 5939, 5949, 5960, 6035
\varemph@uri .....
.... 103, 131, 5980, 5993, 6006, 6035
\texname ..... 24
\text ..... 20
\textbf ..... 19, 6077,
..... 7818, 8487, 8490, 9109, 9200, 9205
\textcolor ..... 9105
\textsymdecl .. 23, 24, 88, 129, 2984, 3905
\textwarning ..... 108
\textwidth ..... 8149, 8171, 9248
\the 250, 251, 253, 255, 257, 2083, 2084, 2085
\theassignment ..... 9193, 9200
\thechapter ... 1331, 1372, 1400, 1435,
..... 7844, 7849, 7853, 7857, 7861, 7865
\theframenumber ..... 8048
\theparagraph ..... 7861, 7865
\thepart .... 1327, 1369, 1396, 1429, 7839
\theplainsproblem ..... 8391, 8392
\thesection .....
.... 7849, 7853, 7857, 7861, 7865, 8392
\thesproblem . 8388, 8392, 8490, 8500, 9193
\thesubparagraph ..... 7865
\thesubsection ... 7853, 7857, 7861, 7865
\thesubsubsection .... 7857, 7861, 7865
\this ..... 57–59, 96, 97, 5223,
..... 6096, 6525, 6602, 6603, 6607, 6632
\thisarchive ..... 3289, 3330
\thisargs ..... 3736, 4506
\thiscopyname .....
.... 3481, 3501, 3525, 3598, 3618, 3639
\thisdeclname .... 3733, 3964, 4178, 4503
\thisdecluri .....
.... 3732, 3963, 4179, 4182, 4502, 4510
\thisdefiniens ..... 3735, 4505
\thisfor ..... 6892
\thismodulename 133, 2535, 3066, 3291, 3332
\thismoduleuri ..... 133, 2534, 3065, 3290, 3331,
..... 3482, 3502, 3526, 3599, 3619, 3640
\thisname ..... 6891
\thisnotation 4180, 4490, 4509, 4798, 4993
\thisnotationvariant .....
..... 4177, 4489, 4508, 4797, 4992
\ThisStyle ..... 5894
\thisstyle ..... 133, 458, 461,
..... 462, 463, 509, 512, 513, 514, 515,
..... 523, 526, 527, 3067, 3292, 3333,
..... 3737, 3965, 4488, 4507, 4796, 4991
\thistitle 103, 133, 2516, 2517, 2518,
..... 6890, 8440, 8486, 8487, 8491, 8492
\thistype ..... 3734, 4504
\thisvarname ..... 4487, 4491, 4795, 4799, 4990, 4994
\throwaway ..... 135
\tikzinput ... 118, 2074, 9320, 9325, 9351
tikzinput commands:
\c_tikzinput_image_bool 34, 9309, 9316
\tiny ..... 8026, 8040
\titl ..... 73, 116
\titl ..... 8221
tl commands:
\tl_clear:N ..... 399, 458, 509,
..... 771, 1343, 1722, 2173, 2292, 2498,
..... 2762, 3067, 3292, 3333, 3545, 3701,
..... 3702, 3703, 3737, 3885, 3908, 3909,
..... 3945, 3965, 4199, 4208, 4304, 4447,
..... 4488, 4507, 4546, 4741, 4796, 4956,
..... 4979, 4991, 5221, 5231, 5328, 5341,
..... 5539, 5926, 5927, 5928, 6025, 6116,
..... 6597, 6625, 6666, 6806, 6807, 6808,
..... 7109, 7110, 7112, 7113, 7114, 7115,
..... 7392, 7393, 8267, 8811, 8907, 8909,
..... 8910, 9051, 9168, 9169, 9170, 9240,
..... 9241, 9242, 9267, 9268, 9269, 9270
\tl_const:Nn ..... 5866, 5867
\tl_count:N ..... 5483, 5489
\tl_count:n ..... 5614, 5625
\tl_gclear:N ..... 2318
\tl_gput_left:Nn ..... 371, 373, 374
\tl_gput_right:Nn ..... 251, 2119,
..... 2123, 2142, 2295, 2559, 3300, 8253
\tl_gset:Nn ..... 253, 366, 367,
..... 1023, 1242, 1249, 2143, 2147, 4215,
..... 5184, 5188, 8539, 8542, 9232, 9233
\tl_gset_eq:NN ..... 46, 2794
\tl_head:N .. 886, 940, 4004, 6547, 6553
\tl_head:n ..... 553, 594, 597, 4692, 5615, 5626
\tl_if_empty:NTF ..... 523, 787, 1025, 1260, 1270, 1313,

```

1749, 2381, 2517, 2527, 3350, 3575,  
3764, 3798, 3801, 3804, 3933, 4108,  
4121, 4193, 4202, 4210, 4380, 4469,  
4545, 4858, 4861, 4864, 4872, 4894,  
4977, 5051, 5054, 5057, 5065, 5295,  
5307, 5398, 5418, 5503, 5642, 5755,  
5768, 5810, 5839, 6098, 7222, 7242,  
7245, 7633, 8152, 8274, 8433, 8486,  
8491, 8533, 8637, 8698, 8747, 8928,  
8930, 8932, 9006, 9008, 9010, 9106,  
9188, 9201, 9206, 9209, 9259, 9286  
\tl\_if\_empty:NTF . . . . . 368, 552,  
568, 584, 593, 751, 760, 882, 883,  
945, 952, 1098, 1568, 1726, 1748,  
1906, 1928, 1959, 2017, 2108, 2215,  
2581, 2596, 2648, 2708, 2733, 2803,  
2818, 2917, 2976, 3077, 3111, 3126,  
3140, 3159, 3190, 3217, 3406, 4002,  
4115, 4128, 4593, 4643, 4950, 5446,  
5467, 5543, 5566, 5969, 6170, 6294,  
6343, 6346, 6391, 6401, 6490, 6492,  
6586, 6595, 6690, 6700, 6764, 6909,  
7031, 7084, 7100, 7596, 8214, 8223  
\tl\_if\_empty\_p:N . . . . . 777  
\tl\_if\_eq:NNTF . . . . .  
..... 1140, 6188, 6547, 6553, 9068  
\tl\_if\_eq:NnTF . . . . .  
..... 8300, 8446, 8449, 8467, 8472  
\tl\_if\_eq:nnTF . . . . .  
.. 2450, 4717, 5111, 5626, 5671, 5709  
\tl\_if\_exist:NTF . . . . . 209,  
250, 265, 304, 462, 514, 526, 1286,  
1295, 1530, 1690, 2551, 6255, 7870  
\tl\_if\_in:NnTF . . . . . 2230, 2248, 2252  
\tl\_item:nn . . . . . 5617  
\tl\_map\_inline:Nn . . . . . 2183  
\tl\_map\_inline:nn . . . . . 216, 220  
\tl\_new:N . . . . .  
965, 1035, 1240, 1565, 2115, 2116,  
2139, 2150, 2468, 4762, 5177, 5181  
\tl\_put\_left:Nn . . . . .  
..... 4632, 4633, 5372, 8414, 9181  
\tl\_put\_right:Nn . . . . .  
1216, 2369, 2744, 2748, 3887, 3888,  
4310, 4316, 4326, 4553, 4558, 4561,  
4664, 4672, 4677, 4685, 4748, 5209,  
5233, 5243, 5353, 6596, 6692, 6739,  
6750, 6765, 9074, 9244, 9245, 9246  
\tl\_range:nnn . . . . . 570, 586  
\tl\_set:Nn . . . . . 210, 247, 453,  
472, 474, 493, 494, 499, 500, 502,  
503, 750, 784, 864, 958, 1022, 1278,  
1319, 1454, 1464, 1580, 1746, 2037,  
2040, 2535, 2644, 2646, 2649, 2864,  
3045, 3046, 3047, 3048, 3049, 3289,  
3330, 3481, 3501, 3525, 3555, 3561,  
3565, 3598, 3618, 3639, 3732, 3808,  
3963, 4024, 4025, 4026, 4027, 4028,  
4065, 4066, 4067, 4068, 4069, 4097,  
4098, 4099, 4100, 4101, 4179, 4194,  
4203, 4228, 4245, 4248, 4257, 4260,  
4266, 4273, 4292, 4502, 4550, 4563,  
4581, 4602, 4617, 4618, 4630, 4669,  
4682, 4701, 4714, 4740, 4823, 4868,  
4909, 4910, 4911, 4912, 4913, 5018,  
5061, 5145, 5178, 5183, 5185, 5192,  
5222, 5228, 5229, 5230, 5342, 5363,  
5483, 5489, 5502, 5508, 5511, 5514,  
5557, 5643, 5675, 5677, 5727, 5870,  
5871, 5915, 5916, 6155, 6167, 6236,  
6309, 6321, 6357, 6358, 6362, 6365,  
6372, 6408, 6447, 6521, 6527, 6537,  
6600, 6602, 6627, 6628, 6634, 6643,  
6691, 6693, 6735, 6740, 6752, 6755,  
6759, 6762, 6767, 6770, 6774, 6777,  
7215, 7221, 7839, 7840, 7844, 7845,  
7849, 7850, 7853, 7854, 7857, 7858,  
7861, 7862, 7865, 7866, 7881, 8369,  
8370, 8484, 8515, 8516, 8576, 8865,  
8919, 8926, 8951, 8973, 9004, 9062,  
9063, 9220, 9226, 9227, 9283, 9289  
\tl\_set\_eq>NN . . . . . 349, 370, 372,  
377, 1571, 2516, 3065, 3066, 3290,  
3291, 3331, 3332, 3482, 3502, 3526,  
3599, 3619, 3640, 3733, 3734, 3735,  
3736, 3964, 4182, 4250, 4262, 4286,  
4434, 4442, 4487, 4503, 4504, 4505,  
4506, 4549, 4786, 4795, 4886, 4891,  
4895, 4975, 4990, 5422, 5562, 5603,  
5682, 6453, 6525, 6607, 6632, 6890,  
7111, 8440, 8444, 8445, 9281, 9282  
\tl\_set\_rescan:Nnn . . . . . 138, 189  
\tl\_tail:n . . . . .  
..... 553, 554, 599, 2215, 5588, 6417  
\tl\_to\_str:n .. 66, 69, 90, 93, 109,  
121, 217, 221, 222, 246, 349, 571,  
587, 683, 706, 713, 812, 830, 832,  
835, 854, 860, 911, 981, 1073, 1176,  
1248, 1263, 1501, 1788, 1813, 1814,  
1842, 1843, 1849, 1898, 1946, 2106,  
2127, 2143, 2146, 2147, 2184, 2187,  
2229, 2231, 2247, 2249, 2253, 2271,  
2280, 2424, 2426, 2427, 2430, 2431,  
2461, 2462, 2520, 2588, 2589, 2590,  
2612, 2845, 2999, 3074, 3349, 3358,  
3547, 4289, 4601, 4615, 4694, 4739,  
5528, 6165, 6198, 6248, 6315, 6335,  
6339, 6498, 6626, 6681, 6711, 6725,

|                                                                                          |                                    |
|------------------------------------------------------------------------------------------|------------------------------------|
| \texttt{6749}, \texttt{6787}, \texttt{6823}, \texttt{7740}, \texttt{8351}, \texttt{9151} |                                    |
| \texttt{\_tl\_trim\_spaces:N} . . . . .                                                  | 49                                 |
| \texttt{\_l\_tmpa\_tl} . . . . .                                                         | 45, 46, 48,                        |
| 4320, \texttt{4741}, \texttt{4748}, \texttt{4755}, \texttt{5172}, \texttt{5470},         |                                    |
| 5476, \texttt{5481}, \texttt{5483}, \texttt{5487}, \texttt{5489}, \texttt{5492},         |                                    |
| 5501, \texttt{5503}, \texttt{5508}, \texttt{5511}, \texttt{5514}, \texttt{5729},         |                                    |
| 6405, \texttt{8926}, \texttt{8942}, \texttt{8945}, \texttt{9004}, \texttt{9020},         |                                    |
| 9023, \texttt{9062}, \texttt{9068}, \texttt{9240}, \texttt{9244}, \texttt{9252}          |                                    |
| \texttt{\_l\_tmpb\_tl} . . . . .                                                         |                                    |
| ... 5470, 5476, 5478, 5492, 5502,                                                        |                                    |
| 5506, 9063, 9068, 9241, 9245, 9253                                                       |                                    |
| \texttt{tmpc} commands:                                                                  |                                    |
| \texttt{\_l\_tmpc\_tl} . . . . .                                                         | 9242, 9246, 9254                   |
| \texttt{to} . . . . .                                                                    | 7541, 7544, 7554, 7555             |
| \texttt{today} . . . . .                                                                 | 8242                               |
| \texttt{TODO} . . . . .                                                                  | 5030, 5031, 6392, 8298             |
| todo internal commands:                                                                  |                                    |
| \texttt{\_l\_todo\_mmt\_module\_str} 7621, 7626,                                         |                                    |
| 7639, 7642, 7645, 7649, 7664, 7680                                                       |                                    |
| \texttt{\_l\_todo\_newlabel:n} . . . . .                                                 | 7739, 7746                         |
| \texttt{\_l\_todo\_old\_metagroup_cd} 7665, 7677                                         |                                    |
| \texttt{\_l\_todo\_old\_newlabel:} . . . . .                                             | 7740, 7745                         |
| \texttt{\_l\_todo\_stex\_module\_str} . . . . .                                          |                                    |
| 7625, 7626, 7651, 7678                                                                   |                                    |
| token commands:                                                                          |                                    |
| \texttt{c\_math\_subscript\_token} . . . . .                                             |                                    |
| ... 49, 87, 4416, 4417, 4420,                                                            |                                    |
| 4421, 4425, 6537, 7546, 7557, 7559                                                       |                                    |
| \texttt{topsep} . . . . .                                                                | 7231, 8550, 8826, 8882, 8984       |
| \texttt{trueFfalseT} . . . . .                                                           | 9045                               |
| \texttt{trueTfalseF} . . . . .                                                           | 9040                               |
| \texttt{type} . . . . .                                                                  | 73, 116                            |
| <b>U</b>                                                                                 |                                    |
| \texttt{undefined} . . . . .                                                             | 134, 41                            |
| \texttt{univ} . . . . .                                                                  | 63                                 |
| \texttt{unless} . . . . .                                                                | 7932                               |
| \texttt{uppercase} . . . . .                                                             | 689, 5963                          |
| \texttt{uri} . . . . .                                                                   | 138, 139                           |
| use commands:                                                                            |                                    |
| \texttt{use:N} . . . . .                                                                 | 248, 381,                          |
| 459, 463, 465, 510, 515, 517, 524,                                                       |                                    |
| 527, 529, 1314, 1316, 1725, 1748,                                                        |                                    |
| 1846, 1851, 2184, 2187, 2662, 3935,                                                      |                                    |
| 5087, 5297, 5323, 6455, 7871, 7877                                                       |                                    |
| \texttt{use:n} . . . . .                                                                 | 1834,                              |
| 2027, 2035, 2049, 2062, 3805, 4865,                                                      |                                    |
| 5058, 5253, 5343, 5375, 6507, 6614                                                       |                                    |
| \texttt{use:nn} . . . . .                                                                | 55, 197, 200, 206, 546,            |
| 1316, 2459, 2483, 2814, 2859, 3015,                                                      |                                    |
| 3020, 3759, 3929, 4183, 4334, 4373,                                                      |                                    |
| 4492, 4511, 4800, 4995, 5142, 5164,                                                      |                                    |
| 5169, 5385, 5588, 5635, 5720, 5895,                                                      |                                    |
| 6136, 6168, 6573, 6640, 6682, 6844,                                                      |                                    |
| 7102, 7430, 7667, 8278, 8578, 9222                                                       |                                    |
| \texttt{use_i:nn} . . . . .                                                              | 6614, 9245                         |
| \texttt{use_ii:nn} . . . . .                                                             | 1838, 2936, 2951, 6654             |
| \texttt{use_none:nnnnn} . . . . .                                                        | 3034                               |
| \texttt{usebox} . . . . .                                                                | 8103                               |
| \texttt{usemodule} . . . . .                                                             | 15, 18, 22, 23, 26,                |
| 37, 49, 93, 94, 96, 212, 422, 429, 3053                                                  |                                    |
| \texttt{usepackage} . . . . .                                                            | 7, 21, 2027, 7946                  |
| \texttt{useSVar} . . . . .                                                               | 109, 8291                          |
| \texttt{usestructure} . . . . .                                                          | 96, 6783                           |
| \texttt{usettheme} . . . . .                                                             | 7946                               |
| \texttt{usetikzlibrary} . . . . .                                                        | 80, 2105, 9319                     |
| <b>V</b>                                                                                 |                                    |
| \texttt{varbind} . . . . .                                                               | 47,                                |
| 52, 95, 100, 6937, 6960, 7071, 7079                                                      |                                    |
| \texttt{vardef} . . . . .                                                                | 35,                                |
| 47, 52, 95, 124, 125, 4762, 7102, 7430                                                   |                                    |
| \texttt{varempf} . . . . .                                                               | 103, 6035                          |
| \texttt{Varname} . . . . .                                                               | 5923                               |
| \texttt{varname} . . . . .                                                               | 5923                               |
| \texttt{varnotation} . . . . .                                                           | 95, 4476                           |
| \texttt{varref} . . . . .                                                                | 5923                               |
| \texttt{varseq} . . . . .                                                                | 43, 95, 4964                       |
| \texttt{vbox} . . . . .                                                                  | 135, 7217, 8020, 8109,             |
| 8124, 8618, 8679, 8728, 8780, 8921                                                       |                                    |
| vbox commands:                                                                           |                                    |
| \texttt{vbox\_set:Nn} . . . . .                                                          | 2152                               |
| \texttt{vdash} . . . . .                                                                 | 7569                               |
| \texttt{vfill} . . . . .                                                                 | 7875, 9126                         |
| \texttt{vM} . . . . .                                                                    | 54                                 |
| \texttt{vMs} . . . . .                                                                   | 55                                 |
| \texttt{vn} . . . . .                                                                    | 43                                 |
| \texttt{vop} . . . . .                                                                   | 45, 47                             |
| \texttt{vrule} . . . . .                                                                 | 7217, 8171, 8921                   |
| \texttt{vskip} . . . . .                                                                 | 7217, 8170, 8172, 8921             |
| \texttt{vspace} . . . . .                                                                | 9128                               |
| <b>W</b>                                                                                 |                                    |
| \texttt{wff} . . . . .                                                                   | 20                                 |
| \texttt{withbrackets} . . . . .                                                          | 92, 5913, 5921                     |
| <b>X</b>                                                                                 |                                    |
| \texttt{xspace} .                                                                        | 1286, 1295, 3979, 3980, 3986, 3987 |
| <b>Y</b>                                                                                 |                                    |
| \texttt{yesFnoT} . . . . .                                                               | 9035                               |
| \texttt{yesTnoF} . . . . .                                                               | 9030                               |
| \texttt{yield} . . . . .                                                                 | 7285, 7476, 7479                   |