

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

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

2023-06-15

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction & Setup | 2 |
| 1.1 | What is $\text{\textcolor{blue}{S}\textcolor{red}{T}\textcolor{blue}{E}\textcolor{red}{X}}$? | 2 |
| 1.2 | The $\text{\textcolor{teal}{S}\textcolor{blue}{T}\textcolor{red}{E}\textcolor{blue}{X}}$ package | 3 |
| 1.3 | What is MMT? | 3 |
| 1.4 | Math archives and the MathHub Directory | 3 |
| 1.5 | Setting Up the $\text{\textcolor{blue}{S}\textcolor{red}{T}\textcolor{blue}{E}\textcolor{red}{X}}$ IDE | 4 |
| I | Tutorial | 6 |
| 2 | The Basics | 7 |
| 2.1 | Text symbols | 9 |
| 2.1.1 | Using Modules & Search in the IDE | 10 |
| 2.2 | Symbol References | 13 |
| 2.3 | Modules and Simple Symbol Declarations | 16 |
| 2.4 | Documenting Symbols | 19 |
| 2.5 | Sectioning and Reusing Document Fragments | 21 |
| 2.6 | Building and Exporting HTML | 23 |
| 3 | Mathematical Concepts | 26 |
| 3.1 | Simple Symbol Declarations | 26 |
| 3.1.1 | Semantic Macros and Notations | 26 |
| 3.1.2 | Types and Variables | 29 |
| 3.1.3 | Flexary Macros and Argument Modes | 34 |
| 3.1.4 | Precedences | 36 |
| 3.1.5 | Implicit Arguments | 36 |
| 3.1.6 | Finishing Equality | 38 |
| 3.1.7 | Variable Sequences | 38 |
| 3.2 | Statements | 39 |
| 3.2.1 | Definitions | 39 |
| | Semantic Macros in Text Mode | 41 |

*Version 3.3.0 (last revised 2023-06-15)

| | |
|---|-----------|
| Definientia | 42 |
| Using Symbols Without Semantic Macros and Exporting Code in Modules | 43 |
| 3.2.2 Assertions | 44 |
| 3.2.3 Proofs | 47 |
| 3.3 Mathematical Structures | 47 |
| 3.3.1 Declaring and Using Structures | 47 |
| Instantiating Structures | 48 |
| 3.3.2 Extending Structures and Axioms | 50 |
| Conservative Extensions | 50 |
| 3.3.3 Nesting Structures and \this | 52 |
| 3.4 Complex Inheritance and Theory Morphisms | 54 |
| 3.4.1 Glueing Structures Together | 56 |
| 3.4.2 Realizations | 58 |
| 4 Extensions for Education | 60 |
| 4.1 Slides and Course Notes | 60 |
| 4.2 Problems and Exercises | 60 |
| 4.2.1 Simple Problems | 61 |
| 4.2.2 Multiple Choice Blocks | 62 |
| 4.2.3 Filling-In Concrete Solutions | 63 |
| 4.3 Including Problems | 64 |
| 4.4 Testing and Spacing | 64 |
| 4.5 Homework Assignments and Exams | 65 |
| II User Manual | 66 |
| 5 Basics | 67 |
| 5.1 Package and Class Options | 67 |
| 5.2 Math Archives and the MathHub Directory | 68 |
| 5.2.1 The Structure of Math Archives | 69 |
| 5.2.2 MANIFEST.MF-Files | 69 |
| 5.3 The lib-Directory | 70 |
| 5.4 Basic Macros | 71 |
| 6 Document Features | 72 |
| 6.1 Document Fragments | 72 |
| 6.2 Using and Referencing Document Fragments | 73 |
| 6.3 Cross-Document References | 73 |

| | |
|---|-----------|
| 7 Modules and Symbols | 76 |
| 7.1 Modules | 76 |
| 7.1.1 Signature Modules, Languages, and Multilinguality | 77 |
| 7.2 Symbol Declarations | 77 |
| 7.2.1 Returns | 78 |
| 7.3 Referencing Symbols | 78 |
| 7.4 Notations and Semantic Macros | 80 |
| 7.4.1 Precedences and Bracketing | 81 |
| 7.4.2 Notations for Argument Sequences | 82 |
| 7.4.3 Semantic Macros | 82 |
| 7.5 Simple Inheritance | 83 |
| 7.6 Variables and Sequences | 85 |
| 7.7 Structures | 86 |
| 7.7.1 Semantic Macros for Structures | 86 |
| 8 Statements | 89 |
| 8.1 More on Definitions | 90 |
| 8.2 More on Assertions | 91 |
| 9 Customizing Typesetting | 92 |
| 9.1 Highlighting Symbol References | 92 |
| 9.2 Styling Environments and Macros | 93 |
| 9.3 Custom CSS for Environments | 94 |
| 10 Additional Packages | 95 |
| 10.1 NotesSlides Manual | 95 |
| 10.1.1 Introduction | 95 |
| 10.1.2 Package Options | 95 |
| 10.1.3 Notes and Slides | 96 |
| 10.1.4 Customizing Header and Footer Lines | 97 |
| 10.1.5 Frame Images | 97 |
| 10.1.6 Ending Documents Prematurely | 98 |
| 10.1.7 Global Document Variables | 98 |
| 10.1.8 Excursions | 99 |
| 10.2 Problem Manual | 100 |
| 10.2.1 Introduction | 100 |
| 10.2.2 Problems and Solutions | 100 |
| 10.2.3 Markup for Added-Value Services | 101 |
| Multiple Choice Blocks | 102 |
| Filling-In Concrete Solutions | 103 |
| 10.2.4 Including Problems | 105 |
| 10.2.5 Testing and Spacing | 105 |
| 10.3 HWExam Manual | 105 |
| 10.3.1 Introduction | 105 |
| 10.3.2 Package Options | 106 |
| 10.3.3 Assignments | 106 |
| 10.3.4 Including Assignments | 106 |
| 10.3.5 Typesetting Exams | 106 |
| 10.4 Tikzinput Manual | 107 |

| | |
|--|------------|
| III Documentation | 109 |
| 11 S^TE_X Developer Manual | 110 |
| 11.1 Documents | 111 |
| 11.2 Modules | 111 |
| 11.3 Symbols | 113 |
| 11.4 Notations | 115 |
| 11.5 Structural Features | 118 |
| 11.6 Imports and Morphisms | 120 |
| 11.7 Expressions and Semantic Macros | 121 |
| 11.8 Optional (Key-Value) Argument Handling | 122 |
| 11.9 Stylistable Commands and Environments | 123 |
| 11.10 Math Archives | 124 |
| 11.11 SMS-Mode | 125 |
| 11.11.1 Second Pass | 125 |
| 11.11.2 First Pass | 126 |
| 11.12 Strings, File Paths, URIs | 126 |
| 11.12.1 File Paths | 127 |
| File Path Constants and Variables | 128 |
| 11.12.2 URIs | 128 |
| URI Constants and Variables | 129 |
| 11.13 Language Handling | 129 |
| 11.14 Inserting Annotations | 130 |
| 11.14.1 Backend macros | 130 |
| 11.15 Persisting Content from Math Archives in sms-Files | 131 |
| 11.16 Utility Methods | 131 |
| 11.16.1 Group-like Behaviours | 132 |
| 12 Additional Packages | 134 |
| 12.1 NotesSlides Documentation | 134 |
| 12.2 Problem Documentation | 134 |
| 12.3 HWExam Documentation | 134 |
| 12.4 Tikzinput Documentation | 134 |
| IV Implementation | 135 |
| 13 The S^TE_X Implementation | 136 |
| 13.1 Setting up | 136 |
| 13.2 Utilities | 137 |
| 13.2.1 Calling kpsewhich and Environment Variables | 137 |
| 13.2.2 Logging | 138 |
| 13.2.3 Languages | 139 |
| 13.2.4 Group-like Behaviours | 141 |
| 13.2.5 HTML Annotations | 142 |
| 13.2.6 Auxiliary Methods | 144 |
| 13.2.7 Persistence | 150 |
| 13.2.8 Files, Paths and URIs | 151 |
| 13.2.9 File Hooks | 159 |
| 13.3 Math Archives | 160 |

| | | |
|----------------------------------|--|-----|
| 13.4 | Documents | 164 |
| 13.4.1 | Title | 164 |
| 13.4.2 | Sectioning | 165 |
| 13.4.3 | References | 169 |
| 13.4.4 | Inputs | 177 |
| 13.5 | SMS Mode | 182 |
| 13.6 | Modules | 186 |
| 13.6.1 | The smodule-environment | 186 |
| 13.6.2 | Structural Features | 197 |
| 13.7 | Inheritance | 202 |
| 13.7.1 | \importmodule/\usemodule | 202 |
| 13.7.2 | Theory Morphisms | 208 |
| 13.8 | Symbols | 213 |
| 13.8.1 | Declarations | 213 |
| 13.8.2 | Notations | 222 |
| a/B-mode argument handling | | 234 |
| 13.8.3 | Variables | 235 |
| 13.8.4 | Sequences | 240 |
| 13.8.5 | Expressions | 244 |
| Invoking Semantic Macros | | 245 |
| Argument Handling and Annotating | | 251 |
| Term HTML Annotations | | 255 |
| Automated Bracketing | | 258 |
| Symname and Variants | | 259 |
| Highlighting | | 261 |
| 13.9 | Mathematical Structures | 263 |
| 13.10 | Statements | 276 |
| 13.11 | Proofs | 282 |
| 13.12 | Metatheory | 290 |
| 13.13 | MMT Interfaces | 292 |
| 14 | Additional Packages | 295 |
| 14.1 | Implementation: The notesslides Package | 295 |
| 14.1.1 | Class and Package Options | 295 |
| 14.1.2 | Notes and Slides | 299 |
| 14.1.3 | Environment and Macro Patches | 302 |
| 14.1.4 | Styling Across Notes/Slides | 304 |
| 14.1.5 | Beamer Compatibility | 304 |
| 14.1.6 | TODO Excursions | 305 |
| 14.2 | Implementation: The problem Package | 306 |
| 14.2.1 | Package Options | 306 |
| 14.2.2 | Problems and Solutions | 307 |
| 14.3 | Implementation: The hwexam Package | 317 |
| 14.3.1 | Package Options | 317 |
| 14.3.2 | Assignments | 317 |
| 14.3.3 | Leftovers | 320 |
| 14.4 | Tikzinput Implementation | 321 |



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

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

If you have questions or problems with `sTeX`, you can talk to us directly at <https://matrix.to/#/#stex:fau.de>.

Chapter 1

Introduction & Setup

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

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

At its core is the [\$\text{\textcolor{teal}{S}\textcolor{blue}{T}\textcolor{teal}{E}\textcolor{blue}{X}}\$ 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 $\text{\textcolor{blue}{S}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}}$ -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 [\LaTeX](#) 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 [\$\text{\textcolor{teal}{S}\textcolor{blue}{T}\textcolor{teal}{E}\textcolor{blue}{X}}\$ package](#),
- the [RusTeX](#) system for converting [\LaTeX](#) 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 [\text{\textcolor{blue}{S}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}} IDE](#): A [VS Code](#) extension that, besides the usual [IDE](#) functionalities like syntax highlighting, integrates [RusTeX](#) and [MMT](#) and allows for accessing the public [math archives](#) on [mathhub.info](#) to make the entire toolchain easily accessible to authors.

If [PDF](#) is all you are interested in, the [\$\text{\textcolor{teal}{S}\textcolor{blue}{T}\textcolor{teal}{E}\textcolor{blue}{X}}\$ package](#) is all you *need*. Either way, however, we recommend using the [\text{\textcolor{blue}{S}\textcolor{teal}{T}\textcolor{blue}{E}\textcolor{red}{X}} IDE](#) – it very much helps with semantically annotating your [\LaTeX](#) documents.

1.2 The **STEX** package

The **STEX** package extends **LATEX** with:

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

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

1.3 What is **MMT**?

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

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

1.4 Math archives and the MathHub Directory

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

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

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

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

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

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



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

1.5 Setting Up the [sTeX IDE](#)

[sTeX](#) is based on [L^AT_EX](#), 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

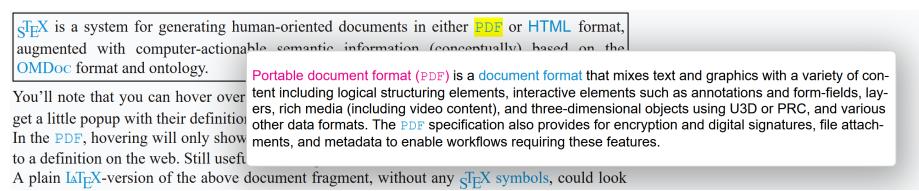


Figure 3: Definition on Hover

```

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:

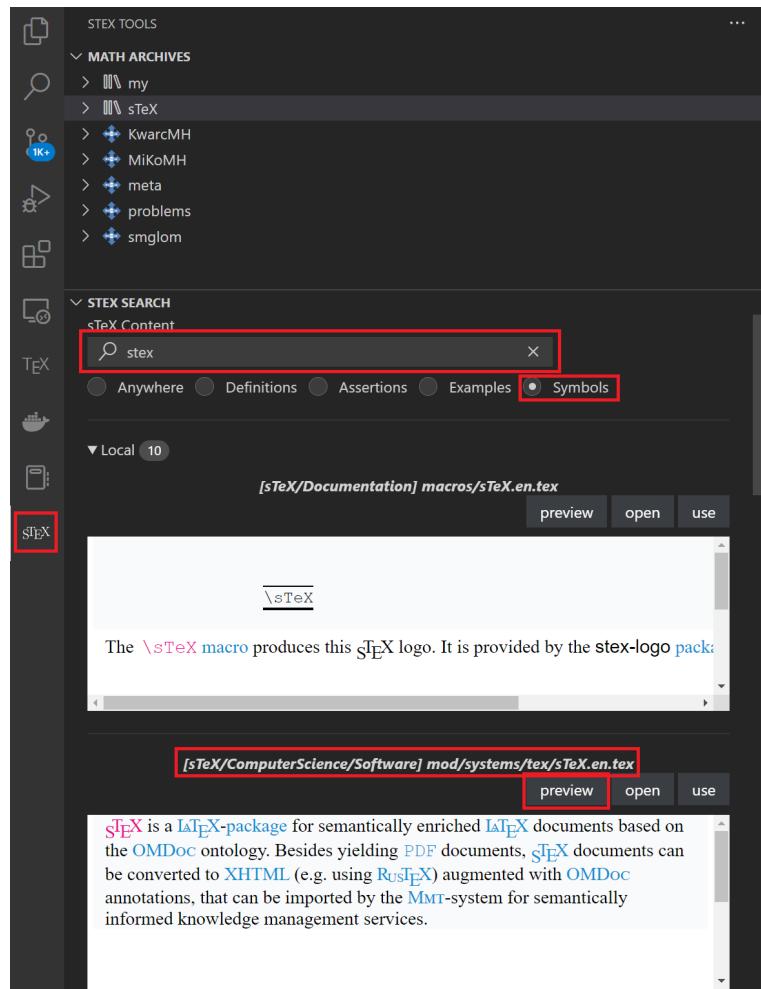


Figure 5: Search in the STeX IDE



Figure 6: OMDoc Preview

Example 2

Input:

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

Output:

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

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

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 `\ifinputref` conditional: by wrapping the header in an `\ifinputref \else... \fi`, it will only be processed if the file is *not* being loaded using `\inputref`. `\ifinputref` is a “classic” **TeX** conditional and is treated as such in both `PDF` and `HTML` compilation. A smarter `macro` to use is `\IfInputref`, which takes two arguments for the *true* and *false* cases, respectively.

Additionally, when compiling to **HTML**, *both* arguments to **\IfInputref** will be processed, and the backend will decide which of the two to present when serving a document.

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



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

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

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

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

2.6 Building and Exporting **HTML**

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

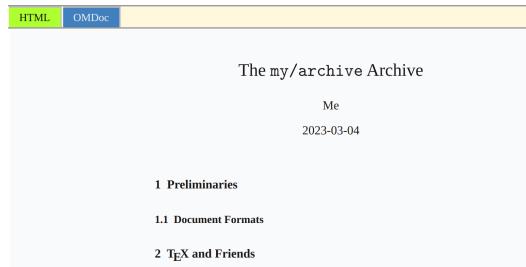


Figure 13: Missing Fragments in the **HTML** Preview

Well, they don't exist yet as **HTML**. The **HTML** Preview window in the **IDE** is really just that: A *preview*. But when using **\inputref**, it has to find the **HTML** of the **\inputref**ed fragment *somewhere*. Meaning: we have to compile all of the fragments we used to **HTML** first. Individually, we can compile the currently open file in **VS Code** using the button in **Figure 14**.

This will do the following:

1. Run **pdflatex** over the file three times.
2. Store the resulting **.pdf** in [**archive**]/**export/pdf/<filepath>.pdf**.



Figure 14: The Build PDF/XHTML/OMDoc Button

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

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



Figure 15: Building Archives in the IDE

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

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

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

In the dialog box that opens now, select an **empty** directory and [MMT](#) will dump a standalone version of our `all.en.tex` document there. You will still not be able to



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

open it in the browser directly, because most browser forbid javascript modules on the `file://` protocol, but opening the file via `http` will yield the desired result, and you can now upload the directory's content to wherever you might want to use it.

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

Chapter 3

Mathematical Concepts

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

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

Also **sTeX** provides specialized functionality for mathematical `statements`: the text fragments marked as Definition, Theorem, Proof that are iconic to mathematical documents.

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

3.1 Simple Symbol Declarations

We will start with `symbols` and `semantic macros` for mathematical concepts and objects and their contribution to mathematical formulae.

3.1.1 Semantic Macros and Notations

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

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

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

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

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

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

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

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

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

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

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

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

and indeed, we can now use the `\eq` **macro** in math mode to invoke our new **notation**: `$\eq{a}{b}$` 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

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

invalid unit:
http://mathhub.info/sTeX/MathTutorial/mod/Equality1/Equality?en?term 1?definition: Judgment |-- (implicit bind [a:/I/1, b:(/I/2 a)] (apply (apply equal a) b)) :: /omitted_type (Invalid Unit)
```

View Problem (Alt+F8) No quick fixes available

Figure 17: Type Checking Warning

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



Figure 18: Type Checking Proof Tree

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

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

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



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

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

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



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

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



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

Types are particularly useful for *variables*:

A **variable** represents a *generic* or *unspecified* object.
STEX **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** \mathbb{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.
Let's change that by giving the `variables` the `type` \mathbb{N} :

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

Example 7

Input:

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

Output:

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

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

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

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

| Type | $(A : \text{SET}, B : \text{SET}) \rightarrow \text{SET}$ | |
|------------|---|---|
| Notations | 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](#).

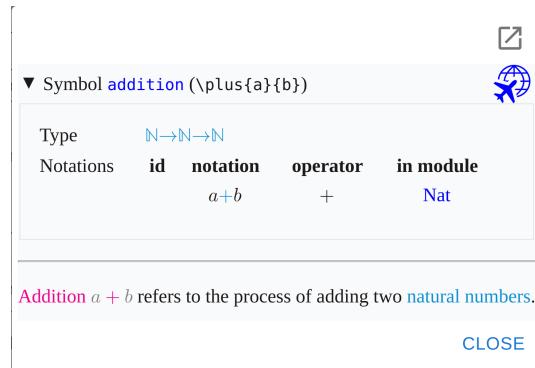


Figure 20: On-Click Popup in the `HTML`

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**

▼ Document Elements

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

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

Figure 21: Inferred Type

block collects all semantically annotated expressions in a `module` or document; including `variables` and the `\plus{\va}{\vb}`. Here, it tells us that it has checked the expression `a + b` (in the context of `a : N` and `b : 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.
3. It knows, that this means that if the two arguments `a` and `b` both have `type N`, then the full expression has `type N`.

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

3.1.3 Flexary Macros and Argument Modes

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

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

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

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

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

Example 8

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

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 $N \times N \rightarrow N$ in this special case, but would also allow `a` and `b` to appear in the range type in more complex situations; see ?? for details.

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

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

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

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

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

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

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

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

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

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

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

Exercise

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

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 | $a \equiv b$ | ≡ | 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". Inside this, a conjunction of equalities is shown: $\{n: \mathbb{N}, m: \mathbb{N}, p: \mathbb{N}\} \frac{n=m \wedge m=p}{\mathbb{N}}$. Below this, the "Inferred Type" is given as $\{n: \mathbb{N}, m: \mathbb{N}, p: \mathbb{N}\} \text{Prop}$.

Figure 23: Conjunction of Equalities

Now $\text{\seqm}{i}{j}$ produces m_i^j , and $\text{\seqm}!$ produces m_1^1, \dots, m_ℓ^k .

Of course, we can manually change the way $\text{\seqn}!$ 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 $\text{\seqn}!$ 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})\{ \mathit{vA}, \mathit{vA} \} \mathit{vA} $ is called
18  \definename{associative}, if
19  \definiens{$ \mathit{foral} \{ \mathit{arg}[2] \{ \mathit{eq}(
20    \mathit{vop} \{ (\mathit{vA}, \mathit{vb}) \}, \mathit{vc} \},
21    \mathit{vop} \{ \mathit{vA}, (\mathit{vop} \{ \mathit{vb}, \mathit{vc} \}) \}
22  } \} \mathit{comp} \{ \mathit{for all} \} \{ \mathit{arg}[1] \{ \mathit{inset} \{ \mathit{va}, \mathit{vb}, \mathit{vc} \} \mathit{vA} \} \} \}.
23 \end{sdefinition}
24 %

```

Output:

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

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



Figure 24: Type Inferred from Definiens

Using Symbols Without Semantic Macros and Exporting Code in Modules

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

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

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

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

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

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

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

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

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

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



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

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

Exercise

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

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:

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

⁵TODO: intent?

which yields

Addition is associative

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

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

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

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

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

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

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

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

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

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

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

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

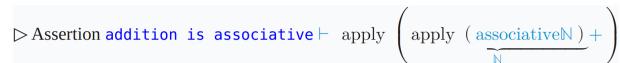


Figure 25: Assertion Statement in `OMDOC`

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

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

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

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

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

```
\begin{assertion}[style=axiom, name=induction axiom]
  Let $\varphi(n)$ a property on $\mathbb{N}$. If
  \begin{enumerate}
    \item $\varphi(0)$ and
    \item if $\varphi(m)$ holds for some $m$, then
      $\varphi(\textcolor{blue}{m+1})$ also holds,
  \end{enumerate}
  then $\varphi(n)$ holds for all $\textcolor{red}{n} \in \mathbb{N}$.
\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 `\implies` (**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 \defname{magma} is a \sr{mathstruct}{structure}  $\$ \magma ! \$$ ,
15   where  $\$ \univ \$$  is a \sn{collection} and  $\$ \op ! \$$ 
16   a binary operation  $\$ \funspace{\univ,\univ}\univ \$$ .
17 \end{sdefinition}
18 \end{mathstructure}
```

Output:

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

Instantiating Structures

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

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

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

| Syntax | Result |
|------------------------|----------------------------|
| $\$\\vM!$$ | M |
| $\$\\vM{}$$ | $\langle U_M, o_M \rangle$ |
| $\$\\vM{univ}{}$$ | U_M |
| $\$\\vM{op}!$$ | o_M |
| $\$\\vM{op}{a,b,c}{}$$ | $a o_M b o_M c$ |

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

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

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

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

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

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

This allows us to do things like:

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

yielding

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

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

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



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

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

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

```
\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 magmas...

Note that in the above, it seems that using `#1` in the `return` argument is allowed. Indeed, it is – the `return` statement takes the same arguments as the `semantic macro` itself does and is appropriately instantiated. Since the first (and only) argument to the

sequence `\vMs` is the index, when doing `\vMs{i}...` the `#1` in the `return`-statement will be replaced by `i`.

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



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

3.3.2 Extending Structures and Axioms

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

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

Example 12

Input:

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

Output:

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

Conservative Extensions

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



Figure 27: Axioms in OMDoc

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

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

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

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

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

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

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

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

Consider the following conservative extension:

Example 13

Input:

```

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

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

```

Output:

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

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

```

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

```

...producing a^2 .

3.3.3 Nesting Structures and `\this`

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

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

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

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

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

```

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

```

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

```

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

```

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

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

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

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

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

Example 14

Input:

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

Output:

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

Now if we instantiate our `magma homomorphism`:

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

Here is a list of what we can do now:

| Syntax | Result |
|---------------------------|-------------------------------|
| $\$\\vh!$$ | H |
| $\$\\vh{}$$ | $\langle M_1, M_2, H \rangle$ |
| $\$\\vh{f}!$$ | H |
| $\$\\vh{f}{a}$$ | $H(a)$ |
| $\$\\vh{dom}!$$ | M_1 |
| $\$\\vh{cod}{}$$ | $\langle U_2, o_2 \rangle$ |
| $\$\\vh{cod}{univ}$$ | U_2 |
| $\$\\vh{dom}{op}!$$ | o_1 |
| $\$\\vh{cod}{op}{a,b,c}$$ | $a o_2 b o_2 c$ |

Note how – as one would expect – we can treat $\backslash vh\{dom\}$ and $\backslash vh\{cod\}$ like any other instance of `magma`.

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

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

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

3.4 Complex Inheritance and Theory Morphisms

We are starting to approach seriously experimental territory.

While the theory behind all the following is relatively well understood, and their implementation in `MMT` is mature, the same can not be said out the implementation in `sTeX`.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

Exercise

Try to implement all of the above yourself!

3.4.1 Glueing Structures Together

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

Definition 3.4.20. A lattice is a structure $\langle U, \wedge, \vee \rangle$ such that $\langle U, \wedge \rangle$ and $\langle U, \vee \rangle$ are semilattices, and \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 **S_TE_X**. 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 **S_TE_X** ecosystem, we will cover them here. In fact, educational documents have been one of the initial document categories **S_TE_X** 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 4.5](#)).

4.1 Slides and Course Notes

TODO⁶

4.2 Problems and Exercises

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. They often 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.

The **problem** package provides functionality for marking up problems and exercises as semantic sources from which various presentations can be generated: documents without solutions for paper or online exams, and the corresponding exams with master solutions

⁶**TODO:** `notesslides.sty`

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{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.

4.2.1 Simple Problems

The main environment provided by the `problem` package is (surprise surprise) the `problem` 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{sproblem}[id=prob.elefants,pts=10,min=2,title=Fitting Elefants]
2 How many Elefants can you fit into a Volkswagen beetle?
3 \begin{hint}
4   Think positively, this is simple!
5 \end{hint}
6 \begin{exnote}
7   Justify your answer
8 \end{exnote}
9 \begin{gnote}
10  if they do not give the justification deduct 5 pts
11 \end{gnote}
12 \begin{solution}
13   Four, two in the front seats, and two in the back.
14 \end{solution}
15 \end{sproblem}
```

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 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.

4.2.2 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 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 17

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}
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 – Korrekt
that is for C and C++
- function – Falsch
that is for Standard ML
- fun – Falsch
that is for Java
- public static void – Noooooooooooo
that is for Java

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

Example 18

Input:

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

Output:

Exercise (Functions)

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

- def
- function
- fun
- public static void

4.2.3 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 19

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 20

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.⁷

4.3 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.

4.4 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

⁷For 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.

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` TODO⁸

4.5 Homework Assignments and Exams

TODO⁹

⁸TODO: check what is still undescribed problem.sty and make examples for it.

⁹TODO: hwexam.sty

Part II

User Manual

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

Chapter 5

Basics

5.1 Package and Class Options

- `debug=(prefixes)`: (see Developer Manual)
- `lang=(languages)`: If set, **S_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 `STEX`3 features,
- `smgloM/*` – a vast collection of multilingual `modules` of concepts in mathematics and computer science. The SMGloM predates `STEX`3 and is thus largely “underannotated” with respect to (formal) semantics,
- `MiKoMH/*` – a vast collection of lecture slides and notes in computer science for courses held by Michael Kohlhase. They largely make use of SMGloM `modules`.

5.2.1 The Structure of Math Archives

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

Each such `archive` needs two subdirectories:

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

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

5.2.2 MANIFEST.MF-Files

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

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

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

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

`source-base` or

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

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

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

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

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

5.3 The lib-Directory

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

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

`\libinput` `\libinput` {`some/file`} searches for a file `some/file[.tex]` in

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

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

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

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

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

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

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

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

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

A good practice is to have individual `sTEX` fragments follow basically this document frame:

```
\documentclass{sTEX}
\libinput{preamble}
\setsectionlevel{<your preference>}
\begin{document}
\IfInputref{}{
...
\maketitle
\ifstexhtml \else \tableofcontents \fi
}
...
\IfInputref{}{\libinput{postamble}}
\end{document}
```

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

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

5.4 Basic Macros

`\sTeX` The `\sTeX` macro produces this S^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` is a `TEX` conditional for whether the current file is currently processed via `\IfInputref` `\inputref`.

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

`\mhgraphics` `\mhgraphics` If the `graphicx` package is loaded, `\mhgraphics` takes the same arguments as `\includegraphics`, with the additional optional key `archive`. It then resolves the file path in

`\mhgraphics[archive=some/archive]{some/image}` relative to the `source`-folder of the `some/archive` `archive`. If no `archive` is provided, the file path `some/image` is resolved relative to the current `archive` (if existent).

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

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

6.3 Cross-Document References

If we take features like `\inputref` and `\mhinput` (and the `sfragment` environment) seriously and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputrefs` a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also `\inputrefed` in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex`

it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

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

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

- The *label* of the reference target (e.g. `some_definition`),
- (optionally) the *file/document* containing the reference target (e.g. `section2`). This is not strictly necessary if the reference target occurs in the *same* document, but if not, we need to know where to find the label,
- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

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

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

This `macro` references *label* (declared in *file1*) in `math archive <archive1>`). If the object (section, figure, etc.) with that label occurs (eventually) in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the `hyperref` package is not included.

If the referenced object does *not* occur in the current document however, `\sref` will refer to it by the object’s name as it occurs in the file *file2* in `archive <archive2>`, followed by the title.

In `HTML` mode, the reference additionally links to the `HTML` of the *file1*.¹⁰

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
↳ `?<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 ≤ 9 ,

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

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

return the `symbol`'s *return code* (see below), most commonly the **semantic macro** of a **mathematical structure**,
assoc how to resolve arguments of `mode` a or B; may be `pre`, `bin`, `binl`, `binr` or `conj`,
reorder how to reorder the arguments in **OMDoc** (*advanced*),
role `symbols` with certain roles are treated in particular ways in **MMT/OMDoc** (*advanced*),
argtypes `TODO`¹¹.

`\textsymdecl` `\textsymdecl{<mname>}[<options>]{<code>}`

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

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

`\symdef` `\symdef{<mname>}[<options>]{<notation>}`

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

7.2.1 Returns

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

1. **STEX** will replace `#1` and `#2` in `<code>` by `a` and `b`, yielding `<retcode>`.
2. **STEX** will insert `<retcode>\foo{a}{b}!` in the input token stream.

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

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

7.3 Referencing Symbols

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

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

¹¹TODO: 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 21

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 `TODO12`

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
 M → as OMA-terms with head `<OMS name="symbol" />`, or `<OMBIND name="symbol" />`,
 ~T → depending on the absence or presence of argument mode `b` or `B` arguments.
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`.

`\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.)`

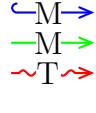
`\begin{extstructure}{\langle mname \rangle}[\langle name \rangle, this=\langle code \rangle]{\langle structs \rangle}` opens a new mathematical structure extending the structures given in $\langle structs \rangle$ (a comma-separated list of names).

`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]):

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

7.7.1 Semantic Macros for Structures

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

Example 22

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

```

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

```

- If `\struct` has no `notations`, then `\struct!` produces $\langle a, b, c \rangle$. Otherwise, it produces the notation, i.e. `StRuCt`. In both cases, `\struct{}{}` produces $\langle a, b, c \rangle$ (for reasons that will become clearer in a moment).
- `\struct{}{}` (or `\struct!` in the case where no `notations` are around) can be modified in the following ways:
 - `\struct{}{{}}[\langle fieldname \rangle, ...]` lets you pick, which precise fields to show, so e.g. `\struct{}{{}}[fielda, fieldb]` produces $\langle a, b \rangle$. By default, `\struct{}{}` shows exactly the fields that have `semantic macros` (which are also used to access the fields in `\struct{{}}{\langle fieldname \rangle}`).
 - `\struct{}{{}}[\langle id \rangle]` lets you pick the `notation` of the “`mathematical structure`” symbol to use to typeset the `structure`; e.g. `\struct{}{{}}[parens]` yields $\langle a, b, c \rangle$, and (combining both) `\struct{}{{}}[fielda, fieldb][angle]` yields $\langle a, b \rangle$.
- The two arguments in `\struct{first}{second}` represent 1. the term that is to be treated as an instance of `\struct`, and 2. the precise field to invoke. If the first is empty, then there is no instance. If the second is empty, `StTeX` will present all of them (that are not assertions). Hence, `\struct{S}{{}}` yields $\langle a_S, b_S, c_S \rangle$, `\struct{S}{fielda}` produces a_S , `\struct{}{fielda}` produces a , and `\struct{S}{fieldb}{x}{y}` produces $x b_S y$.
- For the sake of completion, `\struct{first}!` simply produces the given argument; e.g. `\struct{S}!` simply produces S .

More precisely: `\struct{\langle code \rangle}` acts like a “type coercion” of $\langle code \rangle$ to be an instance of `\struct`.

Of course, it is (occasionally, but) rarely useful to use the `semantic macro` `\struct` by giving it two arguments *manually*; but this is what `StTeX` does when using `\struct` in the `return` of a `symbol` (or `variable`).

Continuing:

- `\struct[comp=\langle code \rangle]{...}{...}` applies $\langle code \rangle$ (using #1) to every occurrence of `\maincomp` in the `notations` of the fields, as a replacement for the default modification `\#1_{\this}`. For example, `\struct[comp=\#1^{\langle Foo \rangle}]{S}{{}` produces $\langle a^{F\!oo}, b^{F\!oo}, c^{F\!oo} \rangle$, and `\struct[comp=\#1^{\langle \this \rangle}]{S}{fieldb}{x}{y}` yields $x b_S y$.
- `\struct[this=\langle code \rangle]{...}{...}` modifies the way `\this` is being typeset; i.e. the presentation of the first `{...}` argument. For example `\struct[this=T]{S}{{}}` produces $\langle a, b, c \rangle$ – since `\this` is not being used in the `notations` of the fields. Note that the `this=` and `comp=` variants are (as of yet) mutually incompatible.
- Finally, `\struct[\langle fieldname \rangle=\langle code \rangle]{...}{...}` assigns the field $\langle fieldname \rangle$ to the term $\langle code \rangle$. $\langle code \rangle$ is subsequently used when using `{...}{}`, but not in fields directly. For example, `\struct[fielda=A]{S}{{}}` produces $\langle A!, b_S, c_S \rangle$, but `\struct[fielda=A]{S}{fielda}` produces a_S .

Note the insertion of ! behind the A – this is to make sure that assignments to [semantic macros](#) that takes arguments don't accidentally eat more than they should.

Also note that multiple assignments can be done in the same pair of [], or chained – i.e. both `$\struct{fielda=A,fieldb=B}...` and `$\struct{fielda=A}[fieldb=B]...` are valid and equivalent. `$\struct{fielda=A,fielda=B}...` however is not – every field may be assigned at most once.

Chapter 8

Statements

STEX provides four environments to semantically annotate various kinds of statements:

sdefinition (*env.*) The **sdefinition environment** represents (primarily mathematical) *definitions*; in particular for *symbols*. The contents of the environment will be used as *documentation* for any *symbol* that either occurs as a **\definiendum** (or **\definename**) within the **sdefinition**, or that is listed in the optional **for=** argument of the **environment**.

If a **\definiens** occurs, this will be used by **MMT** as the formal definiens for the respective *symbol*.

sassertion (*env.*) The **sassertion environment** represents *assertions*, i.e. *propositions* such as *theorems, lemmata, axioms*, etc. If a **\conclusion** occurs within the **sassertion**, its argument will be used as the formal *statement* of the assertion.

sexample (*env.*) The **sexample environment** represents examples (or counterexamples).

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

All of these take the same arguments:

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

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

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

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

\inlineass The macros `\inlineass`, `\inlinedef` and `\inlineex` behave like the `sassertion`, `sdefinition` and `sexample` environments respectively, but take the text to annotate as an argument, rather than as the body of an environment, and do not break paragraphs.

The same macros available in the environments are also available in the argument of the `\inline*` macros.

\varbind `\varbind{\langle cls \rangle}`

retroactively attaches the `bind` option to every `variable` provided (as a comma-separated list).

8.1 More on Definitions

In `sdefinition` (and `sparagraph` with `style=symdoc`), the following additional macros are available:

\definiendum The `\definiendum` macro behaves largely like `\symref`, but it uses a dedicated highlighting for `definienda` and adds the referenced `symbol` to the `for=` list of the environment.
\defname `\defname` is to `\definiendum` as `\symname` is to `\symref`. Analogously, `\Defname` behaves like `\Symname`.

`\defnotation` can be used in math mode to apply the `\definiendum` highlighting to `notations`.

\definiens The `\definiens` macro can be used to semantically annotate the `definiens` in a `sdefinition`.

If the `sdefinition` environment has several elements in its `for` list, an optional argument `\definiens[\langle symbol \rangle]{...}` can be used to tell `STEX` which `symbol`'s definiens this is. By default, the *first* `symbol` in the `for` list is used.

Here is how `MMT` will treat the fragment marked up with `\definiens`:

Firstly, it will attempt to translate its contents into an `MMT/OMDoc` term. This succeeds easily if `\definiens` is some semantic macro (applied to arguments).

Secondly, it will check for all `variables` currently in scope, that were defined with the optional argument `bind`. It then will check, whether a `symbol` is in scope, that has `role=lambda`. If so, it will use that `lambda symbol` to bind all these variables (in the order in which they were defined) in the term. If no `lambda symbol` is found, it will use the `bind symbol` that ships with `STEX`.

The final term will be attached as definiens to the corresponding `MMT` constant, if it was declared in the same `module` as the `\definiens` occurrence.

8.2 More on Assertions

\premise
\conclusion

The `\conclusion` macro can be used to mark up the actual statement within an `sassertion`. The `\premise` macro can be used to additionally mark up *premises*.

If the `sassertion` environment has several elements in its `for` list, an optional argument `\conclusion[⟨symbol⟩]{...}` can be used to tell `STEX` which `symbol`'s statement this is. By default, the *first symbol* in the `for` list is used.

Here is how `MMT` will treat the fragments marked up with `\conclusion` and `\premise`:

Firstly, it will attempt to translate the contents of `\conclusion` into an `MMT/OM-Doc` term c . This succeeds easily if the `\conclusion` is some semantic macro (applied to arguments).

Secondly, it will collect all fragments marked up with `\premise` and do the same to them (p_1, \dots, p_n).

It will then check, whether a `symbol` is in scope, that has `role=implication`. If so, it will use that `implication symbol` to attach all the premises to the conclusion, resulting in $t := \text{imply}(p_1, \dots, p_n, c)$.

Next, it will check for all `variables` currently in scope, that were defined with the optional argument `bind`. It then will check, whether a `symbol` is in scope, that has `role forall`. If so, it will use that `forall symbol` to bind all these variables (in the order in which they were defined) in the term t .

Finally, it will check, whether a `symbol` is in scope, that has `role=judgment`. If so, it will set $t := \text{judgment}(t)$.

If no `forall symbol` is found, it will first apply the `judgment symbol` (if existent) and then use the `bind symbol` that ships with `STEX` to bind the `variables`.

The final term will be attached as `type` to the corresponding `MMT constant`, if it was declared in the same `module` as the `\definiens` occurrence.

`sproof (env.)`

TODO¹³

¹³TODO: proofs

Chapter 9

Customizing Typesetting

There are two kinds of typesetting that can be customized in **STEX**: **symbol** references (**notation** components, `\symref`, **variables**, etc.) are highlighted using a small set of **macros** that can be simply redefined by authors.

Other **macros** and **environments** usually have more complicated “typesetting rules” associated with them – often in the form of other already existing **environments** that should be used.

Lastly, in **HTML** we can provide custom CSS rules in **math archives** that determine the styling of certain **environments**, so that the actual presentation depends on the document in which the fragments are included (e.g. via `\inputref`), rather than the file the fragment is implemented in.

It is generally recommended to implement these customizations in a preamble in the `lib` directory (see [section 5.3](#))

9.1 Highlighting Symbol References

`\symrefemph`
`\symrefemph@uri`

`\symrefemph` governs how references via `\symref` (or `\symname`, or their short variants) are highlighted;

Doing `\symref{<symbol>}{<text>}` ultimately expands to `\symrefemph@uri{<text>}{{<symbol URI>}}`, the default implementation of which is just `\symrefemph{<text>}`. The default implementation of `\symrefemph`, in turn, is just `\emph{<text>}`.

If you only want to change e.g. the color of `\symrefs`, you only need to redefine `\symrefemph`, e.g. using

```
\renewcommand\symrefemph[1]{\textcolor{red}{#1}}
```

the `@uri` variant is useful if you want to link somewhere, or show the URI in a tooltip. The **stex-highlighting package** does both, using:

```
\usepackage{pdfcomment}
\protected\def\symrefemph@uri#1#2{%
  \pdftooltip{%
    \srefsymuri{#2}{\symrefemph{#1}}%
  }{%
    URI:~\detokenize{#2}%
  }%
```

```

}
\def\symrefemph#1{%
  \ifcsname textcolor\endcsname
    \textcolor{teal}{#1}%
  \else#1\fi
}

```

`\compemph` Like `\symrefemph` and `\symrefemph@uri`, but governs the highlighting of components
`\compemph@uri` (marked with `\comp` or `\maincomp`) in `notations`.

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

`\varemph` Like `\compemph` and `\compemph@uri`, but governs the highlighting of components (marked
`\varemph@uri` with `\comp` or `\maincomp`) in the `notations` of `variables`.
The second argument to `\varemph@uri` is the *name* of the `variable`.

9.2 Styling Environments and Macros

A variety of `environments` and `macros` provided by `STEX` are *stylistable* using the `macros` `\stextstyle{name}[\langle style \rangle]{...}`. These *stylistable environments* and `macros` bind various of their parameters to `macros` `\this{parameter}`, which can then be used in the styles.

For example, if we have a `definition environment` that we would want to use to style our `sdefinitions`, we can do (in the simplest case)

```
\stextstyledefinition{\begin{definition}}{\end{definition}}
```

This tells `STEX` to insert `\begin{definition}` at the beginning of every `sdefinition environment`, and `\end{definition}` at the end.

If we have a `environments theorem` and `lemma`, we probably want the `sassertion environment` to use those for theorems and lemma. We can achieve that by doing

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

Now if we do `\begin{sassertion}[style=theorem]`, it will wrap the `environment` with `\begin{theorem}... \end{theorem}`.

Of course, many such statements might have a title, as e.g. in

Theorem 9.2.1 (Gödel's First Incompleteness Theorem). ...

In `sassertion`, we can provide that title as optional argument using `title=....` Before calling the styling provided, `sassertion` will store that title in the macro `\thistitle`, which we can use in the styling. For example, we might prefer to pass it on to the `theorem environment`:

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

```
\stexstylemodule           TODO14
\stexstylecopymodule
\stexstyleinterpretmodule
\stexstylerealization
\stexstylemathstructure
\stexstyleextstructure
\stexstyledefinition
\stexstyleassertion
\stexstyleexample
\stexstyleparagraph
\stexstyleproof
\stexstylesubproof
```

Additionally, we can style certain [macros](#), if we want them to produce output. For example, we might (for debuggin or documentation purposes) `\symdecl` to give a short summary of the symbol.

We can achieve that by doing, for example:

```
\stexstylesymdecl[debug]{
    Symbol \thisdeclname~(with arity \thisargs) of type \$\thistype$.
```

in which case

```
\symdecl{foo}[args=2,type=\mathbb{N},style=debug]
```

will yield

```
Symbol foo (with arity ii) of type N.
```

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

9.3 Custom CSS for Environments

Chapter 10

Additional Packages

10.1 NotesSlides Manual

10.1.1 Introduction

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

The `notesslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the `STEX` and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `notesslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

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

10.1.2 Package Options

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

`slides` The options `slides` and `notes` switch between slides mode and notes mode (see [subsection 10.1.3](#)).

`sectocframes` If the option `sectocframes` is given, then for the `sfragments`, special frames with the `sfragment` title (and number) are generated.

`frameimages` If the option `frameimages` is set, then slide mode also shows the `\frameimage`-generated `fiboxed` frames (see [??](#)). If also the `fiboxed` option is given, the slides are surrounded by a box.

10.1.3 Notes and Slides

frame (*env.*) Slides are represented with the **frame** environment just like in the **beamer** class, see [Tantau:ugbc] for details.

note (*env.*) The **notesslides** class adds the **note** environment for encapsulating the course note fragments.



Note that it is essential to start and end the **notes** environment at the start of the line – in particular, there may not be leading blanks – else L^AT_EX 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^TE_X 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^TE_X 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 23

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}[for=elefants]
13 Four, two in the front seats, and two in the back.
14 \begin{gnote}
15 if they do not give the justification deduct 5 pts
16 \end{gnote}
17 \end{solution}
18 \end{sproblem}
19 \end{document}

```

Output:

Exercise (Fitting Elefants)

How many Elefants can you fit into a Volkswagen beetle?

solution (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.

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

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

10.2.3 Markup for Added-Value Services

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

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

Multiple Choice Blocks

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

`\mcc` `\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 24

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 – Korrekt
- function – Falsch
that is for C and C++
- fun – Falsch
that is for Standard ML
- public static void – Nooooooooooooo
that is for Java

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

Example 25

Input:

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

Output:

Exercise (Functions)

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

- def
- function
- fun
- public static void

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

Filling-In Concrete Solutions

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

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

Example 26

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 27

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. ¹⁶

¹⁶For the moment we only assume a single concrete value as correct. In the future we will almost certainly want to extend the functionality to multiple answer classes that allow different feedback like in MCQ. This still needs a bit of design. Also we want to make the formatting of the answer in solutions/test mode configurable.

10.2.4 Including Problems

\includeproblem

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

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

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

10.2.5 Testing and Spacing

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

```
\testspace      \testspace takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: \testsmallspace, \testmediumspace, \testlargepage \testsmallspace give small (1cm), medium (2cm), and big (3cm) vertical space.  
\testsmallspace \testnewpage makes a new page in test mode, and \testemptypage generates an empty page with the cautionary message that this page was intentionally left empty.  
\testemptypage Local Variables: mode: latex TeX-master: ../stex-manual End:  
  LocalWords: solutions,notes,hints,gnotes,pts,min,boxed,test gnotes elefants,pts gnote  
  LocalWords: 2,title exnote hint,exnote,gnote ifsolutions mcb keyvals Ttext Ftext Local-  
  Words: Functions,name F,feedback Noooooooooo,feedback 2,title includeproblem Local-  
  Words: elefants.fillin,title
```

10.3 HWExam Manual

10.3.1 Introduction

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

10.3.2 Package Options

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

gnotes

pts

min

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

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

10.3.3 Assignments

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

10.3.4 Including Assignments

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

10.3.5 Typesetting Exams

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

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

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2023-06-15

You have one hour (sharp) for the test;

Write the solutions to the sheet.

The estimated time for solving this exam is 20 minutes, leaving you 40 minutes for revising your exam.

You can reach 20 points if you solve all problems. You will only need 27 points for a perfect score, i.e. -7 points are bonus points.

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

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

| | To be used for grading, do not write here | | | | | | | | | |
|---------|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| prob. | 4.1 | 1.1 | 1.2 | 2.1 | 2.2 | 2.3 | 2.4 | 2.5 | 2.5 | Sum |
| total | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 20 |
| reached | | | | | | | | | | |

good luck

¹⁷

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

LocalWords: hwexam solutions,notes,hints,gnotes,pts,min gnotes testemptytypepage reqpts LocalWords: inputassignment reqpts hour,min 60,reqpts

10.4 Tikzinput Manual

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

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

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

```

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

The `standalone` class is a minimal L^AT_EX 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^AT_EX over it separately, e.g. for generating an image file from it.

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

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

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

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

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

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

Part III

Documentation

Chapter 11

sTeX Developer Manual



Keeping the implementation properly up-to-date is pretty much incompatible with the kinds of workflows systemically enforced in academia. Any of the following may be out of date.

* indicates fully expandable functions, which can be used within an x-type argument (in plain TeX terms, inside an `\edef`), as well as within an f-type argument. ☆ indicates restricted expandable functions, which can be used within an x-type argument but cannot be fully expanded within an f-type argument. *TF* indicates conditional (if) functions whose variants with T, F and TF argument specifiers expect different “true”/“false” branches.

`\stex_debug:nn \stex_debug:nn {<prefix>} {<msg>}`

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

`_stex_do_deprecation:n` TODO `_stex_do_deprecation:n`

11.1 Documents

`\l_stex_docheader_sect` integer register keeping track of the current sectioning level:

- 0 part
- 1 chapter
- 2 section
- 3 subsection
- 4 subsubsection
- 5 paragraph
- > 5 subparagraph

`\setsectionlevel` sets `\l_stex_docheader_sect` to the corresponding integer value.

`\stex_ref_new_doc_target:n` internal variant of `\sreflabel`. If the argument is empty, the label is determined to be `REF<counter>` and `<counter>` is increased.

`\stex_ref_new_symbol:n` registers a new link target for the symbol with the given full uri (as string), using the `url-base`-field of the current archive's manifest file to link the symbol to `<url-base>/symbol?<uri>`.

`\stex_ref_new_sym_target:n` sets a new label for the symbol with the given full uri (as string). If called in sms-mode, defers to `\stex_ref_new_symbol:n`, if not already registered. Otherwise, sets a `\label` for the symbol.

`\stex_ref_new_sym_target:nn` `\stex_ref_new_sym_target:nn {<symbol>} {<target>}`

redirects links for `<symbol>` to the one for the symbol `<target>`. Useful for e.g. symbols elaborated from structural features. Note that this acts as a *default*, in that previous or subsequent calls of `\stex_ref_new_sym_target:n{<symbol>}` are prioritized.

Requires that either `\stex_ref_new_sym_target:n{<target>}` or `\stex_ref_new_sym_target:nn{<target>} {<other-target>}` have been called previously.

11.2 Modules

The contents of a module with full URI `<uri>` are represented as four macros:

- `\c_stex_module_<uri>_code`: code to be executed every time the module is activated; e.g. the contents of `\STEXexport`, defines semantic macros and macros for notations, activates dependency modules, etc.

- `\c_stex_module_<uri>_morphisms_prop`: property list containing all module dependencies of this module (see `\stex_module_add_morphism:nnn`).
- `\c_stex_module_<uri>symbols_prop`: property list containing all declarations in this module (see `\stex_module_add_symbol:nnnnnnN`).
- `\c_stex_module_<uri>_notations_prop`: property list containing all notations introduced in this module (see `\stex_add_module_notation:nnnnn`).

`\l_stex_current_module_str` contains the full URI of the current module.

`\l_stex_all_modules_seq` contains the full URIs of all modules currently in scope.

`\stex_module_setup:n \stex_module_setup:n {<name>}`

Computes the full URI of a new module with name `<name>` in the current namespace, initializes the four macros above and sets `\l_stex_current_module_str` accordingly. Also takes care of correct naming for nested modules, activates the meta theory and loads the signature module if `sig=` was provided (according to `\l_stex_key_sig_str`).

`\stex_close_module:` closes the current module; checks whether we are currently in sms mode, and if so, calls `\stex_persist:n` to write the module contents to the sms-file.

`\stex_every_module:n \stex_every_module:n {<code>}`

executes `<code>` every time a new module is opened.

`\stex_if_in_module_p: *` tests whether we are currently in a module.
`\stex_if_in_module:TF *`

`\stex_if_module_exists_p:n *`
`\stex_if_module_exists:nTF *`

tests whether a module with the given full URI exists, in the sense of *has been parsed at some point in the current document*.

`\stex_activate_module:n` activates the module with the given full URI *if and only if* it has not already been activated in the current scope.
`\stex_activate_module:(o|x)`

`\stex_execute_in_module:n` executes the provided code, adds it to the current module activation code, and makes sure the macros defined in it are valid in the current module TeX group level.
`\stex_execute_in_module:x`

This macro is a combination of the following two macros:

`\stex_do_up_to_module:n` executes the provided code such that all definitions in it are valid in the current module
`\stex_do_up_to_module:x` regardless of TeX group level (using `\stex_metagroup_do_in:nn`).

`\stex_module_add_code:n` adds the provided code to the module's `\c_stex_module_<uri>_code`-macro.
`\stex_module_add_code:x`

11.3 Symbols

A symbol in **STEX** is represented as a tuple of several components:

`\stex_module_add_symbol:nnnnnnnN`

#1 : `<id>`: An *identifier* (possibly empty) that determines its semantic macro name, or e.g. in `mathstructure` its accessor-identifier (if empty, the name is used for that),
#2 : `<name>` a unique *name*, which in combination with the containing module determines its URI as `<module-URI>?<name>`,
#3 : `<arity>` a numeric string in the range 0..9,
#4 : `<args>` a list of argument specifiers of the form `<i><mode>{<argname>}` (the length of `<args>` must be 3·`<arity>`),
#5 : `<definiens>` (or empty),
#6 : `<type>` (or empty),
#7 : `<return code>` (or empty), and
#8 : `<\invocation_macro>`.

the arguments are stored in the property list `\c_stex_module_<\l_stex_current_module>_symbols_prop` under key `<name>`.

If the identifier `<id>` is non-empty, the macro `\id` is defined as `{\stex_invoke_symbol:nnnnnnnN}` with the arguments described there.

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

`\stex_iterate_symbols:n`
`\stex_iterate_break:`

`\stex_iterate_symbols:n {<code>}`

`\stex_iterate_break:n`

iterates over all symbols currently in scope and calls `<code>` for all of them with arguments `{<module>} {<name>} {<arity>} {<args>} {<definiens>} {<type>} {<return code>} {<\invocation_macro>}`.

Iteration can be stopped prematurely with `\stex_iterate_break:` and can stop and return code with `\stex_iterate_break:n`.

`\stex_iterate_symbols:nn`

`\stex_iterate_symbols:nn {<csl>} {<code>}`

iterates over all symbols in the provided `<csl>` of full module URIs. `<code>` receives the same arguments as `\stex_iterate_symbols:n`, but iteration can not be stopped early.

```
\stex_get_symbol:n  
\l_stex_get_symbol_mod_str  
\l_stex_get_symbol_name_str  
\l_stex_get_symbol_arity_int  
\l_stex_get_symbol_args_tl  
\l_stex_get_symbol_def_tl  
\l_stex_get_symbol_type_tl  
\l_stex_get_symbol_return_tl  
\l_stex_get_symbol_invoke_cs
```

`\stex_get_symbol:n` attempts to find a symbol with the given name or id that is currently in scope. A name may be prefixed with a module name/path separated by `?`.

Throws an error if no such symbol is found; otherwise, sets the listed `\l_stex_get_symbol_...` macros to the components of the found symbol.

```
\stex_if_check_terms_p: * whether to typeset declaration components (notations, types, definientia etc.) in a throw-  
\stex_if_check_terms:TF * away box. Is true iff the backend is pdflatex and either the STEX_CHECKTERMS environment variable or checkterms package option is set.
```

`\stex_check_term:n` typesets the argument in a throwaway box in math mode iff `\stex_if_check_terms:` is true.

Is deactivated in sms-mode.

```
\stex_symdecl_do:  
\l_stex_key_name_str  
\l_stex_key_args_str  
\l_stex_key_argnames_clist  
\l_stex_assoc_args_count  
\l_stex_argnames_seq
```

`\stex_symdecl_do:` processes the shared (mandatory and optional) arguments of e.g. `\symdecl`, `\symdef`, `\vardef` etc.

Requires the following macros to be set

- `\l_stex_key_name_str`: the name of the symbol,
- `\l_stex_key_args_str`: the `args`-string (e.g. `3` or `ai`)
- `\l_stex_key_argnames_clist`: a list of *names* for the arguments, the length of which should be \leq the arity of the symbol

and will generate the following macros:

- `\l_stex_get_symbol_arity_int`: the arity of the symbol,
- `\l_stex_key_args_str`: the args string as a definite sequence of argument-mode characters, whose length is the arity of the symbol; e.g. `3` is turned into `iii`,
- `\l_stex_assoc_args_count`: the number of sequence arguments (i.e. `a` or `B` mode),
- `\l_stex_argnames_seq`: the full sequence of argument names; those not provided by `\l_stex_key_argnames_clist` are set to be `$j`, where `j` is the index of the argument,
- `\l_stex_get_symbol_args_tl`: a token list of triples `j{argname}`, where `j` is the index and `m` the respective argument mode character (i.e. `i`, `a`, `b` or `B`).

`_stex_symdecl_check_terms:`

calls `\stex_check_term:n` for the type and definiens stored in `\l_stex_key_type_tl` and `\l_stex_key_def_tl`

`\stex_symdecl_top:n \stex_symdecl_top:n {{maybename}}`

checks whether `\l_stex_key_name_str` is empty, and if so, sets it to be `<maybename>`. Then calls `\stex_symdecl_do:` and `_stex_symdecl_check_terms:`, writes the components to the HTML (if applicable) and adds the symbol to the current module with invocation macro `\stex_invoke_symbol:` and id/macroname `\l_stex_mroname_str`.

Variables work very similar to symbols, except that their declarations are local to the current TeX-group rather than the current module, and they are not exported in modules.

`\l_stex_variables_prop` stores all variables currently in scope as a property list with key `<name>` and value `{{id}}{{name}}{{arity}}{{args}}{{definiens}}{{type}}{{return code}}{{\invokation_macro}}`.
The invocation macro for “normal” variables declared with `\vardef` is `\stex_invoke_symbol:`.

`_stex_vardecl_notation_macro:`

generates the notation macro for a variable, based on the values of the `\l_stex_key-` macros und `\l_stex_notation_macrocode_cs`.

`\stex_get_var:n` like `\stex_get_symbol:n`, but attempts to retrieve a variables and throws an error if none is found.

`\stex_get_symbol_or_var:n` like `\stex_get_symbol:n`, but first attempts to find a *variable*, and if none is found, defers to `\stex_get_symbol:n`.

11.4 Notations

`\stex_module_add_notation:nnnnn \stex_module_add_notation:eoexo \stex_module_add_notation:nnnnn`
 `{{symboluri}}{{id}}{{arity}}{{code}}{{op code}}`

stores the arguments in the property list `\c_stex_module_{\l_stex_current_module}_notations_prop` under key `<symboluri>!<id>` and calls `\stex_set_notation_macro:nnnnn`.

```
\stex_set_notation_macro:nnnnn \stex_set_notation_macro:nnnnn
\stex_set_notation_macro:eoexo {symboluri}{id}{arity}{code}{op code}
```

Declares the following macros:

An active notation for a symbol with uri $\langle symboluri \rangle$ and id $\langle id \rangle$ is represented as a macro $\backslash l_stex_notation_{\langle symboluri \rangle}_{\langle id \rangle}_cs$, that takes $\langle arity \rangle$ many argument and expands to $\langle code \rangle$, and (if nonempty) an *operator* notation as a macro $\backslash l_stex_notation_{\langle symboluri \rangle}_op_{\langle id \rangle}_cs$ that expands to $\langle op code \rangle$.

The default notation is represented as the *empty* id. If the correponding macros $\backslash l_stex_notation_{\langle symboluri \rangle}_cs$, and $\backslash l_stex_notation_{\langle symboluri \rangle}_op__cs$ do not yet exist, they are now defined as $\langle id \rangle$.

```
\stex_iterate_notations:nn \stex_iterate_notations:nn {csl}{code}
```

iterates over all notations in the provided $\langle csl \rangle$ of full module URIs and calls $\langle code \rangle$ on each of them with arguments $\{symboluri\}{id}{arity}{code}{op code}$.

```
\stex_notation_parse_and_then:nw \stex_notation_parse_and_then:nw {code}{notations}
\l_stex_key_prec_str
\l_stex_key_variant_str
\l_stex_notation_macrocode_cs
```

parses a notation (which may consist of multiple braced components, depending on the argument modes) and subsequently calls $\langle code \rangle$.

Requires the following macros to be set:

- $\backslash l_stex_get_symbol_arity_int$,
- $\backslash l_stex_get_symbol_args_tl$,
- $\backslash l_stex_key_prec_str$: The precedence string as provided by a user in the optional `precs`-argument,
- $\backslash l_stex_key_variant_str$: the id of the notation variant.

Stores the final notation in the macro $\backslash l_stex_notation_macrocode_cs$ taking $\backslash l_stex_get_symbol_arity_int$ many arguments.

Some thing to consider when we generate a notation macro:

- The notation macro generated by the `\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^AT_EX 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^AT_EX 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 28

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_EX parameter token (#2) the notation body ((#1 -- ##1 split ##2 -- #3)), and finally the argument mode (a). Additionally, the markers ##1 and ##2 are replaced by `\STEXInternalAssocArgMarkerI` and `\STEXInternalAssocArgMarkerII`, respectively.

Subsequently, the non-sequence parameter tokens are wrapped in `\STEXInternalTermMathArgiii` with arguments mj (where m is the mode und j the index), the precedence (20 or 40 respectively), and the parameter token.

Finally, a `\STEXInternalSymbolAfterInvocationTL` is inserted.

The final expansion of `\l_stex_notation_macrocode_cs` is thus:

```
1 \STEXInternalTermMathOMAiii{foo}{10}{
2   First: \STEXInternalTermMathArgiii{i1}{20}{#1};
3   Second: \STEXInternalTermMathAssocArgiiii{2}{30}{#2}{
4     (\STEXInternalTermMathArgiii{i1}{20}{##1} --
```

```

5      \STEXInternalAssocArgMarkerI split
6      \STEXInternalAssocArgMarkerII --
7      \STEXInternalTermMathArgiii{i3}{40}{##3}
8  }{a};
9  Third: \STEXInternalTermMathArgiii {i3}{40}{#3}; End
10 }\STEXInternalSymbolAfterInvocationTL

```

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

calls `\stex_notation_parse_and_then:nw{<code>}` and, adds the notation for the symbol with URI `<symboluri>` to the current module and exports it to the HTML (if applicable).

11.5 Structural Features

`\stex_structural_feature_module:nn` `\stex_structural_feature_module:nn {<name>}{<typeid>}`
`\stex_structural_feature_module_end:` `\g_stex_last_feature_str`

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

Its body behaves like a nested module with name `<modulename>/<name>`, the full URI of which is stored in `\g_stex_last_feature_str` for subsequent elaboration.

```
\stex_structural_feature_morphism:nnnnn
\stex_structural_feature_morphism_end: \stex_structural_feature_morphism:nnnnn
{\morphisname}{\typeid}{\archive}{\domain}{\annotations}
\l_stex_current_domain_str
\l_stex_feature_name_str
\l_stex_morphism_symbols_prop
\l_stex_morphism_renames_prop
\l_stex_morphism_morphisms_seq
```

opens a new morphism-like structural feature of type $\langle typeid \rangle$ with name $\langle morphisname \rangle$ and the module $[\langle archive \rangle] \{ \langle domain \rangle \}$ as domain, which needs to be closed with `\stex_structural_feature_morphism_end:`.

Deactivates `\symdecl`, `\textsymdecl`, `\symdef`, `\notation` and `smodule`, and activates `\assign`, `\assignMorphism` and `\renamedecl`.

Defines the following macros:

- `\l_stex_feature_name_str ={\langle name \rangle}.`
- `\l_stex_current_domain_str` = the full uri of $\langle domain \rangle$.
- `\l_stex_morphism_symbols_prop`: This property list is initialized as follows: For every symbol transitively included in $\langle domain \rangle$ with data $\langle module \rangle$, $\langle name \rangle$, $\langle id \rangle$, $\langle arity \rangle$, $\langle args \rangle$, $\langle definiens \rangle$, $\langle type \rangle$, and $\langle return code \rangle$, the property list contains an entry with key $[\langle module \rangle]/[\langle name \rangle]$ and value $\{\langle id \rangle\}\{\langle arity \rangle\}\{\langle args \rangle\}\{\langle definiens \rangle\}\{\langle type \rangle\}\{\langle return code \rangle\}$.
- `\l_stex_morphism_renames_prop`: An initially empty property list.
- `\l_stex_morphism_morphisms_seq`: TODO

At `\stex_structural_feature_morphism_end:`, the elaboration is computed from the above data thusly:

For every entry in `\l_stex_morphism_symbols_prop`, a new symbol is created with the values $\langle arity \rangle$, $\langle args \rangle$, $\langle definiens \rangle$, $\langle type \rangle$ and $\langle return code \rangle$ from that property list, and either:

- if `\l_stex_morphism_renames_prop` does not contain an entry with key $\langle module \rangle? \langle name \rangle$, then the elaborated name is $\langle morphisname \rangle / \langle name \rangle$ and its $\langle id \rangle$ is empty (no semantic macro is generated), or
- if `\l_stex_morphism_renames_prop` contains an entry with key $\langle module \rangle? \langle name \rangle$, then its value needs to be of the form $\{\langle id \rangle\} \{\langle name \rangle\}$, which are used for the elaborated symbol.

All notations of the symbols transitively included in the domain are copied over to their elaborations.

11.6 Imports and Morphisms

```
\stex_module_add_morphism:nnnn
\stex_module_add_morphism:(nonn|ooox)
```

adds a new module morphism (i.e. inheritance, possibly with modification) to the current module

#1 : The name of the morphism (may be empty for e.g. `\importmodule`, but may be named for e.g. `copymodule`),

#2 : the URI of the module being “imported”,

#3 : the “type” of the morphism (e.g. `import` or `copymodule`),

#4 : a list of assignments as pairs $\{\langle source \rangle\} \{\langle target \rangle\}$ that signify that the symbol with full URI $\langle source \rangle$ is being mapped or elaborated to the new symbol with name $\langle target \rangle$ in the current module. May be empty for e.g. `\importmodule`.

The provided arguments are stored in `\c_stex_module_<uri>_morphisms_prop` with key #1 (if non-empty) or [#2] (if #1 is empty) and value {#1}{#2}{#3}{#4}

Import-like statements in STEX are usually given as pairs $[\langle archive \rangle] \{\langle path \rangle? \langle module \rangle\}$, which be relative to the current archive and/or file path. For persistence and sms-mode, these pairs first need to be resolved into an *absolute* specification:

```
\stex_import_module_uri:nn
\l_stex_import_archive_str
\l_stex_import_name_str
\l_stex_import_uri_str
\l_stex_import_path_str
```

`\stex_import_module_uri:nn` $\{\langle archive \rangle\} \{\langle pathstring \rangle\}$

(where $\langle archive \rangle$ may be empty) resolves the given arguments into:

- `\l_stex_import_archive_str` the given archive id (in which case `\stex_require_archive:n` is called), or the id of the current archive (if existent and $\langle archive \rangle$ empty), or empty,
- `\l_stex_import_uri_str` if an archive id is given, or we currently are in an archive, its corresponding namespace; otherwise `{file:}`,
- `\l_stex_import_path_str` the path of the URI relative to the given (or current) archive, or (if not existent) the absolute path of $\langle pathstring \rangle$ (without a module name) resolved relative to the current file’s parent directory, and
- `\l_stex_import_name_str` the name of the module.

If $\langle pathstring \rangle$ does not contain the character ?, the whole pathstring is assumed to be the name of the module and the path is empty (or the current file’s parent directory, depending on the above).

```
\stex_require_module:nnn
```

takes as arguments values `\l_stex_import_archive_str`, `\l_stex_import_path_str` and `\l_stex_import_name_str` as computed by `\stex_import_module_uri:nn` and (optionally loads and) activates the corresponding module (or throws an error if it does not exist).

Most complex morphisms (`copymodule` et al) are implemented as structural features using `\stex_structural_feature_morphism:nnnn`.

```
\stex_get_in_morphism:n  
\l_stex_get_symbol_macro_str
```

finds a symbol with the provided name or id in the domain of the current morphism. Sets the same macros as `\stex_get_symbol:n`, and additionally `\l_stex_get_symbol_macro_str` to the $\langle id \rangle$ of the symbol. Throws an error if no such symbol is found.

11.7 Expressions and Semantic Macros

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

is how a semantic macro is/should be defined. `_stex_invoke_symbol: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 ={\langle module \rangle?\langle name \rangle}`
- `\l_stex_current_arity_str ={\langle arity \rangle}`
- `\l_stex_current_args_tl ={\langle args \rangle}`
- `\l_stex_current_type_tl ={\langle type \rangle}`
- `\l_stex_current_return_tl ={\langle return code \rangle}`

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

```
\_stex_invoke_variable:nnnnnnN
```

analogous to `_stex_invoke_symbol: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=\langle return \rangle`.
 - * If $\langle return \rangle$ is empty, call the notation macro,
 - * otherwise, call $\langle return \rangle\{\l_stex_invoke_symbol!\}$.
- If text:

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

TODO

11.8 Optional (Key-Value) Argument Handling

LATEX3 is surprisingly weak when it comes to handling optional (key-val) arguments in such a manner that *only* the freshly set macros are defined, and to modularly build up sets of argument keys. The following macros attempt to fix that:

```
\stex_keys_define:nnnn  
\stex_keys_set:nn
```

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

Defines a set of keys and their allowed values with identifier **stex**/*⟨id⟩*, that inherits from the sets with identifiers in *⟨parents⟩*.

\stex_keys_set:nn {*⟨id⟩*}{{*CSL*}} first executes *⟨setup code⟩* (e.g. to empty the macros holding the values) and then sets the keys in set *⟨id⟩* with the values provided in *⟨CSL⟩*.

_stex_do_id: should be called whenever a macro or environment has a label id, i.e. calls **\stex_keys_set:nn**{*id*}{{...}}, after the title has been typeset. Sets a **\label** by calling **\stex_ref_new_doc_target:n**{*⟨id⟩*}.

Example 29

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 \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 \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 \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 30

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

Example 31

\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^AT_EX3 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^AT_EX3 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^AT_EX3 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^AT_EX3 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_annotation:nnn
```

```
\stex_annotation:nnn {⟨attr⟩}{⟨value⟩}{⟨code⟩}
```

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

\stex_annotation_invisible:n
\stex_annotation_invisible:n

\stex_annotation_invisible:n {⟨code⟩}

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

\stex_annotation_invisible:nnn combines \stex_annotation_invisible:n and \stex_annotation_invisible:n.

`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 32

```
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{expl3,13keys2e}
5  \ProvidesExplClass{stex}{2023/03/19}{3.3.0}{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{expl3,13keys2e,ltxcmds}
19 \ProvidesExplPackage{stex}{2023/03/19}{3.3.0}{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~3.3.0~*^^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   unknown     .code:n     = {}
36 }
37 \exp_args:NNo \clist_set:Nn \c_stex_debug_clist \c_stex_debug_clist
38 \ProcessKeysOptions { stex / package }

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

```

13.2 Utilities

40 \cs_set_eq:NN \stex_undefined: \undefined

13.2.1 Calling kpsewhich and Environment Variables

41 <@=stex_envs>

\stex_kpsewhich:Nn

```

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

```

(End of definition for \stex_kpsewhich:Nn. This function is documented on page 132.)

\stex_get_env:Nn

```

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

```

(End of definition for \stex_get_env:Nn. This function is documented on page 132.)

13.2.2 Logging

```

63  <@=stex_debug>

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

```

(End of definition for `\stex_debug:nn`. This function is documented on page 110.)

`\stex_fatal_error:n`
`\stex_fatal_error:nnn`
`\stex_fatal_error:nxx`

To avoid dead locks etc., we throw errors and make tex stop entirely:

```

80 \cs_new_protected:Nn \stex_fatal_error:n {
81   \msg_error:nn{stex}{#1}\input{Fatal-Error!!}
82 }
83 \cs_new_protected:Nn \stex_fatal_error:nnn {
84   \msg_error:nnn{stex}{#1}{#2}{#3}\input{Fatal-Error!!}
85 }
86 \cs_generate_variant:Nn \stex_fatal_error:nnn {nxx}

```

(End of definition for `\stex_fatal_error:n` and `\stex_fatal_error:nnn`. These functions are documented on page 132.)

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

```

87 \stex_get_env:Nn\__stex_debug_env_str{STEX_DEBUG}
88 \str_if_empty:NTF\__stex_debug_env_str {
89   \clist_set_eq:NN \l__stex_debug_cl \c_stex_debug_clist
90 }{
91   \clist_set:No \l__stex_debug_cl {\__stex_debug_env_str}
92 }
93 \clist_clear:N \c_stex_debug_clist
94 \clist_map_inline:Nn \l__stex_debug_cl {
95   \exp_args:NNo \clist_put_right:Nn \c_stex_debug_clist
96   { \tl_to_str:n{#1} }
97 }
98
99 \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist {\tl_to_str:n{all}} {
100   \msg_redirect_module:nnn{ stex }{ none }{ warning }
101   \stex_debug:nn{all}{Logging~everything!}
102 }{
103   \clist_map_inline:Nn \c_stex_debug_clist {
104     \msg_redirect_name:nnn{ stex }{ debug / #1 }{ warning }
105     \stex_debug:nn{#1}{Logging~#1}

```

```

106     }
107 }
```

13.2.3 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
121 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
122   english = en ,
123   ngerman = de ,
124   arabic = ar ,
125   bulgarian = bg ,
126   russian = ru ,
127   finnish = fi ,
128   romanian = ro ,
129   turkish = tr ,
130   french = fr
131 } }
132 % todo: chinese simplified (zhs)
133 %         chinese traditional (zht)
```

(End of definition for \c_stex_languages_prop and \c_stex_language_abbrevs_prop. These variables are documented on page 129.)

\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 129.)

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:N\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 130.)

`\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 130.)

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 }

```

13.2.4 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 132.)

\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{\__stex_groups_\tl_to_str:n{##1}}
223   }
224 }

```

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

\stex_metagroup_new:n List of all currently existing metagroup identifiers

\stex_metagroup_new:o 225 \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 133.)

```

\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:cng{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 133.)

13.2.5 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 130.)

`\stex_suppress_html:n` Temporarily disable HTML output

```

285 \cs_new_protected:Nn \stex_suppress_html:n {
286     \stex_pseudogroup:nn{
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 130.)

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:N\stex@backend{
305         \if@rustex
306             \def\stex@backend{rustex}
307         \else
308             \cs_if_exist:NTF\HCode{
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
320 \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 130.)

`_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 ??.)

13.2.6 Auxiliary Methods

`\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   \tl_set:Nn#1{
351     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#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 132.)

```

\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 132.)

`\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 122.)

`\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 122.)

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 110.)

\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 122.)

```

```

\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 123.)

`\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 126.)

`\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 126.)

`\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 131.)

`\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 131.)

13.2.7 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 131.)

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 }

```

13.2.8 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 127.)

```

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

```

```

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

(End of definition for `\stex_file_resolve:Nn`. This function is documented on page [127](#).)

```

\stex_if_file_absolute_p:N
\stex_if_file_absolute:NTF
745
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   \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
758     \seq_if_empty:NTF #1 \prg_return_false: {
759       \exp_args:Nx \tl_if_empty:nTF {\seq_item:Nn #1 1}
760         \prg_return_true: \prg_return_false:
761       }
762     }
763 }
```

(End of definition for `\stex_if_file_absolute:NTF`. This function is documented on page [127](#).)

```

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

(End of definition for `\stex_file_use:N`. This function is documented on page [127](#).)

```

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

```

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

```

(End of definition for `\stex_if_file_starts_with:NNTF`. This function is documented on page 127.)

`\stex_file_split_off_ext:NN`
`\stex_file_split_off_lang:NN`

```

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

```

(End of definition for `\stex_file_split_off_ext:NN` and `\stex_file_split_off_lang:NN`. These functions are documented on page 127.)

URIs:

`\stex_map_uri:Nnnnn`

```

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

```

```

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

```

(End of definition for `\stex_map_uri:Nnnnn`. This function is documented on page 128.)

```

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

```

```

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

```

(End of definition for `\stex_uri_set:Nn`. This function is documented on page 128.)

`\stex_uri_resolve:Nn`

```

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

```

(End of definition for `\stex_uri_resolve:Nn`. This function is documented on page 128.)

`\stex_uri_use:N`

```

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

```

(End of definition for `\stex_uri_use:N`. This function is documented on page 128.)

`\stex_uri_from_repo_file>NNNn`

`\stex_uri_from_repo_file_nolang:NNNn`

```

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

```

```

912     }
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
926     \stex_uri_set:Nx #1 { \l__stex_path_uri / \stex_file_use:N \l__stex_path_seq }
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 128.)

```

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

```

(End of definition for `\stex_uri_from_current_file:Nn` and `\stex_uri_from_current_file_nolang:Nn`.
These functions are documented on page 129.)

```

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

```

```

957         \tl_set:Nx #1 {#2}
958     }
959     }{
960         \msg_error:nnnn{stex}{error/invalid-uri}{#2}{Not~a~URI}
961     }
962 }
963 \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 129.)

`\l_stex_current_doc_uri`

```
964 \tl_new:N \l_stex_current_doc_uri
```

(End of definition for `\l_stex_current_doc_uri`. This variable is documented on page 129.)

`\stex_set_document_uri:`

```

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

```

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

`\stex_set_current_namespace:`

```

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

```

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

We determine the PWD

```

\c_stex_pwd_file
\c_stex_main_file
973 \sys_if_platform_windows:TF{
974     \stex_get_env:Nn\l_stex_path_str{CD}
975 }{
976     \stex_get_env:Nn\l_stex_path_str{PWD}
977 }
978 \stex_file_resolve:No \c_stex_pwd_file \l_stex_path_str
979 \seq_set_eq:NN \c_stex_main_file \c_stex_pwd_file
980 \seq_put_right:Nx \c_stex_main_file {\jobname\tl_to_str:n{.tex}}
981
982 \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 128.)

13.2.9 File Hooks

keeps track of file changes:

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

`\stex_filestack_push:n`

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

(End of definition for `\stex_filestack_push:n`. This function is documented on page [127](#).)

`\stex_filestack_pop:`

```
1007 \cs_new_protected:Nn \stex_filestack_pop: {
1008     \seq_if_empty:NF \g__stex_path_stack {
1009         \seq_gpop>NN \g__stex_path_stack \l__stex_path_str
1010     }
1011     \seq_if_empty:NTF \g__stex_path_stack {
1012         \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
1013     }{
1014         \seq_get:NN \g__stex_path_stack \l__stex_path_str
1015         \exp_args:NNo \stex_file_set:Nn \g_stex_current_file \l__stex_path_str
1016         \seq_gset_eq:NN \g_stex_current_file \g_stex_current_file
1017     }
1018     \_stex_every_file:
1019 }
```

(End of definition for `\stex_filestack_pop:`. This function is documented on page [127](#).)

Hooks for the current file:

```
1020 \cs_new_protected:Nn \stex_input_with_hooks:n {
1021     \tl_set:Nn \l__stex_path_do_hooks_pre_tl {
1022         \tl_gset:Nn \l__stex_path_do_hooks_pre_tl {}
1023         \stex_debug:nn{HERE}{Hook~for~#1^~J\meaning\CurrentFilePath^~J\CurrentFile}
1024         \tl_if_empty:NTF\CurrentFilePath{
```

```

1025     \exp_args:No \stex_filestack_push:n {\CurrentFile}
1026     }{
1027         \exp_args:Ne \stex_filestack_push:n { \CurrentFilePath / \CurrentFile }
1028     }
1029 }
1030 \input{#1}
1031 \stex_debug:nn{HERE}{Hook-end-for~#1}
1032 \stex_filestack_pop:
1033 }
1034 \tl_new:N \l__stex_path_do_hooks_pre_tl {}
1035
1036 \AddToHook{file/before}{
1037     \l__stex_path_do_hooks_pre_tl
1038 }
1039 \%AddToHook{file/after}{ \stex_filestack_pop: }

```

13.3 Math Archives

```

1040 <@=stex_mathhub>
\mathhub
\c_stex_home_file
\c_stex_mathhub_file
1041
1042 \sys_if_platform_windows:TF{
1043     \stex_get_env:Nn \l__stex_mathhub_str {homedrive\c_percent_str\c_percent_str homepath}
1044 }{
1045     \stex_get_env:Nn \l__stex_mathhub_str {HOME}
1046 }
1047 \stex_file_resolve:No \c_stex_home_file \l__stex_mathhub_str
1048
1049 \str_if_empty:NTF\mathhub{
1050     \stex_get_env:Nn \l__stex_mathhub_str {MATHHUB}
1051     \str_if_empty:NTF \l__stex_mathhub_str {
1052         \ior_open:NnTF \g_tmpa_iorf{\stex_file_use:N \c_stex_home_file/.stex/mathhub.path}{
1053             \group_begin:
1054                 \escapechar=-1\catcode`\\"=12
1055                 \ior_str_get:NN \g_tmpa_ior \l__stex_mathhub_str
1056                 \str_gset_eq:NN \l__stex_mathhub_str \l__stex_mathhub_str
1057             \group_end:
1058                 \ior_close:N \g_tmpa_ior
1059                 \stex_debug:nn{mathhub}{MathHub-directory-determined-from-home-directory}
1060             }{
1061                 \str_clear:N \l__stex_mathhub_str
1062             }
1063     }{
1064         \stex_debug:nn{mathhub}{MathHub-directory-determined-from-environment-variable}
1065     }
1066 }{
1067     \str_set_eq:NN \l__stex_mathhub_str \mathhub
1068 }
1069
1070 \str_if_empty:NTF \l__stex_mathhub_str {
1071     \msg_warning:nn{stex}{warning/nomathhub}

```

```

1072  \exp_args:NNe \stex_file_set:Nn \c_stex_mathhub_file {\stex_file_use:N \c_stex_home_file \
1073  \seq_clear:N \c_stex_mathhub_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 124.)

```

\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       \stex_fatal_error:n{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 124.)

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     \stex_fatal_error:nxx{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:NNNx
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:cN {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:
1238 }

```

(End of definition for `\c_stex_main_archive_prop` and `\l_stex_current_archive_prop`. These variables are documented on page 124.)

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{shtml: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 \thepart {
1430         \stex_annotation_invisible:nn{shtml: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{shtml:sectionlevel=1}{}}
1437       }{
1438         \int_incr:N \l_stex_docheader_sect
1439         \__stex_doc_check_topsect:
1440       }
1441     }{
1442       \stex_annotation_invisible:nn{shtml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
1443     }
1444   }
1445   \AtBeginDocument{\__stex_doc_check_topsect:}
1446 }
1447
1448 \AtBeginDocument{
1449   \cs_if_exist:NTF\frontmatter{
1450     \let\__stex_doc_orig_frontmatter\frontmatter
1451     \let\frontmatter\relax
1452   }{
1453     \tl_set:Nn\__stex_doc_orig_frontmatter{
1454       \clearpage
1455       %\mainmatterfalse
1456       \pagenumbering{roman}
1457     }
1458   }
1459   \cs_if_exist:NTF\backmatter{
1460     \let\__stex_doc_orig_backmatter\backmatter
1461     \let\backmatter\relax
1462   }{
1463     \tl_set:Nn\__stex_doc_orig_backmatter{

```

```

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

```

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

13.4.3 References

```
1492 <@=stex_refs>
```

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

```

1493 \iow_new:N \c__stex_refs_iow
1494 \AtBeginDocument{\iow_open:Nn \c__stex_refs_iow {\jobname.sref}}
1495 \AtEndDocument{\iow_close:N \c__stex_refs_iow}

```

The following macros are written to the `.aux`-file, and hence use L^AT_EX2e character code scheme:

```

1496 \cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}
1497
1498 \cs_new_protected:Npn \STEXInternalSetSrefSymURL #1 #2 {
1499     \str_gset:cn{g_stex_sref_sym_\tl_to_str:n{#1}_target}{#2}
1500 }
1501

```

```

\stex_ref_new_doc_target:n
\srflabel
1502 \seq_new:N \g__stex_refs_files_seq
1503 \int_new:N \l__stex_refs_unnamed_counter_int
1504
1505 \cs_new_protected:Nn \_stex_ref_new_id:n {

```

```

1506 \str_if_empty:nTF {#1}{
1507   \int_gincr:N \l__stex_refs_unnamed_counter_int
1508   \str_set:Nx \l__stex_refs_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1509 }{
1510   \str_set:Nn \l__stex_refs_str {#1}
1511 }
1512 \str_set:Nx \l_stex_ref_url_str {\stex_uri_use:N \l_stex_current_doc_uri ? \l__stex_refs_s
1513 }
1514
1515 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1516   \stex_ref_new_id:n{#1}
1517   \%stex_uri_add_module:NNo \l__stex_refs_uri \l_stex_current_doc_uri \l__stex_refs_str
1518   \%stex_debug:nn{sref}{New-document-target:~\stex_uri_use:N \l__stex_refs_uri}
1519   \__stex_refs_add_doc_ref:xo {\stex_uri_use:N \l_stex_current_doc_uri} \l__stex_refs_str
1520   \stex_if_smemode:F {
1521     \iow_now:Nx \c__stex_refs_iow {
1522       \STEXInternalSrefRestoreTarget
1523       {\stex_uri_use:N \l_stex_current_doc_uri}
1524       {\l__stex_refs_str}
1525       {\@currentcounter}
1526       {\@currentlabel}
1527       {
1528         \tl_if_exist:NT\@currentlabelname{
1529           \exp_args:No\exp_not:n\@currentlabelname
1530         }
1531       }
1532     }
1533     \exp_args:Nx \label {sref@\l_stex_ref_url_str}
1534     \stex_if_do_html:T {
1535       \pdfdest name "sref@\l_stex_ref_url_str" xyz\relax
1536     }
1537   }
1538 }
1539 \NewDocumentCommand \sreflabel {m} {\stex_ref_new_doc_target:n {#1}}
1540
1541 \cs_new_protected:Nn \__stex_refs_add_doc_ref:nn {
1542   \seq_if_in:NnTF \g__stex_refs_files_seq {#1} {
1543     \seq_if_in:cnF {g__stex_refs_#1_seq}{#2}{
1544       \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1545     }
1546   }{
1547     \seq_gput_right:Nn \g__stex_refs_files_seq {#1}
1548     \seq_new:c{g__stex_refs_#1_seq}
1549     \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1550   }
1551 }
1552 \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 [111](#).)

\sref Optional arguments:

\extref

```

1553 \stex_keys_define:nnnn{sref / 1}{}{
1554   % TODO get rid of this

```

```

1555     fallback .code:n = {},
1556     pre .code:n = {},
1557     post .code:n = {}
1558 }{archive file}
1559 \stex_keys_define:nnnn{sref / 2}{}{}{archive file, title}
1560
1561 \str_new:N \l__stex_refs_default_archive_str
1562 \str_new:N \l__stex_refs_default_file_str
1563 \tl_new:N \l__stex_refs_default_title_tl
1564
1565 \cs_set_protected:Nn \__stex_refs_set_keys_b:n {
1566     \tl_if_empty:nTF{#1}{%
1567         \str_set_eq:NN \l_stex_key_archive_str \l__stex_refs_default_archive_str
1568         \str_set_eq:NN \l_stex_key_file_str \l__stex_refs_default_file_str
1569         \tl_set_eq:NN \l_stex_key_title_tl \l__stex_refs_default_title_tl
1570     }{%
1571         \stex_keys_set:nn{ sref / 2 }{ #1 }
1572     }
1573 }
1574
1575 \newcommand\srefsetin[3][]{%
1576     \str_set:Nx \l__stex_refs_default_archive_str {#1}
1577     \str_set:Nx \l__stex_refs_default_file_str {#2}
1578     \tl_set:Nn \l__stex_refs_default_title_tl {#3}
1579 }
1580

```

Auxiliary methods:

```

1581 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1582     \str_clear:N \l__stex_refs_uri_str
1583     \stex_debug:nn{sref}{%
1584         File:~\l_stex_key_file_str^^J
1585         Repo:\l_stex_key_archive_str
1586     }
1587     \str_if_empty:NTF \l_stex_key_file_str {%
1588         \stex_debug:nnf{sref}{Empty.~Checking~current~file~for~#1}
1589         \seq_if_exist:cT{\g__stex_refs_\stex_uri_use:N \l_stex_current_doc_uri _seq}{%
1590             \exp_args:Nnx \__stex_refs_find_uri_in_file:nnn{#1}{%
1591                 {\stex_uri_use:N \l_stex_current_doc_uri}\seq_map_break:
1592             }
1593             \str_if_empty:NT \l__stex_refs_uri_str {%
1594                 \seq_map_inline:Nn \g__stex_refs_files_seq {%
1595                     \__stex_refs_find_uri_in_file:nnn{#1}{##1}{\seq_map_break:n{\seq_map_break:}}
1596                 }
1597             }
1598 }{%
1599     \str_if_empty:NTF \l_stex_key_archive_str {%
1600         \prop_if_exist:NTF \l_stex_current_archive_prop {%
1601             \__stex_refs_find_uri_in_prop_file:N \l_stex_current_archive_prop
1602         }{%
1603             \stex_file_resolve:Nx \l__stex_refs_file
1604             { \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str }
1605             \str_set:Nx \l__stex_refs_uri_str { file:/ \stex_file_use:N \l__stex_refs_file }
1606         }
1607     }
1608 }
```

```

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

```

Doing the actual referencing:

```

1637 \cs_new_protected:Nn \__stex_refs_do_autoref:n {
1638     \cs_if_exist:cTF{autoref}{%
1639         \exp_args:Nx\autoref{sref@#1}
1640     }{
1641         \exp_args:Nx\ref{sref@#1}
1642     }
1643 }
1644
1645 \cs_new_protected:Nn \__stex_refs_do_sref:nn {
1646     \str_if_empty:NTF \l__stex_refs_uri_str {
1647         \str_if_empty:NTF \l_stex_key_file_str {
1648             \stex_debug:nn{sref}{autoref~on~\stex_uri_use:N \l_stex_current_doc_uri?#1}
1649             \exp_args:Ne \__stex_refs_do_autoref:n{\stex_uri_use:N \l_stex_current_doc_uri ? #1}
1650         }{
1651             \stex_debug:nn{sref}{srefin~on~#1}
1652             \__stex_refs_set_keys_b:n{ #2 }
1653             \__stex_refs_do_sref_in:n{#1}
1654         }
1655     }{
1656         \exp_args:NNo \seq_if_in:NnTF \g__stex_refs_files_seq \l__stex_refs_uri_str {
1657             \stex_debug:nn{sref}{Using~ref~file~\l__stex_refs_uri_str}
1658             \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref}\l__stex_refs_uri_str _seq}{\detokenize{#1}}
1659     }
1660 }

```

```

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

```

```

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

```

```

1767     }
1768   }
1769 }
1770 }
```

The actual macros:

```

1771 \NewDocumentCommand \sref { O{} m O{} }{
1772   \stex_keys_set:nn { sref / 1 }{ #1 }
1773   \__stex_refs_find_uri:n { #2 }
1774   \__stex_refs_do_sref:nn{#2}{#3}
1775 }
1776 \NewDocumentCommand \extref { O{} m m }{
1777   \stex_keys_set:nn { sref / 1 }{ #1 }
1778   \__stex_refs_find_uri:n { #2 }
1779   \__stex_refs_set_keys_b:n{#3}
1780   \str_if_empty:NT \l_stex_key_file_str {
1781     \msg_error:nn{stex}{error/extrefmissing}
1782   }
1783   \__stex_refs_do_sref_in:n{#2}
1784 }
```

(End of definition for `\sref` and `\extref`. These functions are documented on page 74.)

`\stex_ref_new_sym_target:n`

```

1785 \cs_new_protected:Nn \stex_ref_new_symbol:n {
1786   \cs_if_exist:cF{r@sref@sym@\tl_to_str:n{#1}}{
1787     \__stex_refs_new_symbol:n{#1}
1788   }
1789 }
1790
1791 \cs_new_protected:Nn \__stex_sref_do_aux:n {
1792   #1 \iow_now:Nn \auxout {#1}
1793 }
1794
1795 \cs_new_protected:Nn \__stex_refs_new_symbol:n {
1796   \prop_if_exist:NTF \l_stex_current_archive_prop {
1797     \prop_get:NnNTF \l_stex_current_archive_prop {docurl} \l__stex_refs_str {
1798       \exp_args:Ne \__stex_sref_do_aux:n {
1799         \STEXInternalSetSrefSymURL{#1}{\l__stex_refs_str / symbol? #1}
1800       }
1801     }{
1802       \__stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
1803     }
1804   }{
1805     \__stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
1806   }
1807 }
1808
1809 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1810   \cs_if_exist:NT \hypertarget{
1811     \exp_args:Ne \hypertarget{\tl_to_str:n{sref@sym@ #1}}{}
1812     \str_gset:cx{\tl_to_str:n{r@sref@sym@ #1}}{\tl_to_str:n{sref@sym@ #1}}
1813   }
1814 }
```

```

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

```

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

\srefsym

```

1823 \NewDocumentCommand \srefsym { m m }{
1824   \stex_get_symbol:n { #1 }
1825   \exp_args:Ne
1826   \__stex_refs_sym_aux:nn{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
1827 }
1828
1829 \cs_new_protected:Npn \__stex_refs_do_internal_link:nn #1 {
1830   \cs_if_exist:NTF \hyperlink {
1831     \hyperlink{#1}
1832   }\use:n
1833 }
1834
1835 \cs_new_protected:Npn \__stex_refs_do_url_link:nn {
1836   \cs_if_exist:NTF \href \href \use_i:nn
1837 }
1838
1839 \cs_new_protected:Npn \__stex_refs_sym_aux:nn #1 {
1840   \cs_if_exist:cTF{\tl_to_str:n{r@sref@sym@#1}}{
1841     \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1842   }{
1843     \str_if_exist:cTF{g_stex_sref_sym_#1_label}{
1844       \exp_args:Ne \__stex_refs_sym_aux:nn{\use:c{g_stex_sref_sym_#1_label}}
1845     }{
1846       \str_if_empty:cTF{g_stex_sref_sym_#1_target}{
1847         \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1848       }{
1849         \exp_args:Ne \__stex_refs_do_url_link:nn{\use:c{g_stex_sref_sym_#1_target}}
1850       }
1851     }
1852   }
1853 }

```

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

\srefsymuri

```

1854 \cs_new_protected:Npn \srefsymuri #1 {
1855   \__stex_refs_sym_aux:nn{#1}
1856 }

```

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

13.4.4 Inputs

```

1857 <@=stex_inputs>

\stex_resolve_path_pair:Nnn
\stex_resolve_path_pair:Nxx
1858 \cs_new_protected:Nn \stex_resolve_path_pair:Nnn {
1859   \stex_debug:nn{resolving-path}{#3-in-[#2]}
1860   \str_if_empty:nTF{#2} {
1861     \prop_if_exist:NTF \l_stex_current_archive_prop {
1862       \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1863       \prop_item:Nn \l_stex_current_archive_prop { id } / source /
1864       #3}
1865       \stex_debug:nn{resolving-path}{In-current-archive-
1866       \prop_item:Nn \l_stex_current_archive_prop { id }
1867       ;~result:~#1}
1868     }{
1869       \str_set:Nx #1 {\stex_file_use:N \c_stex_pwd_file / .. / #3 }
1870       \stex_debug:nn{resolving-path}{No-current-archive;~result:~#1}
1871     }
1872   }{
1873     \stex_require_archive:n { #2 }
1874     \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1875     \prop_item:cn {c_stex_mathhub_#2_manifest_prop} { id } / source /
1876     #3}
1877     \stex_debug:nn{resolving-path}{result:~#1}
1878   }
1879 }
1880 \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 124.)

```

\inputref
\mhinput
\ifinputref
1881 \newif \ifinputref \inputreffalse
1882
1883 \cs_new_protected:Nn \__stex_inputs_mhinput:nn {
1884   \stex_in_archive:nn {#1} {
1885     \ifinputref
1886       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1887     \else
1888       \inputreftrue
1889       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1890       \inputreffalse
1891     \fi
1892   }
1893 }
1894
1895 \NewDocumentCommand \mhinput { O{} m }{
1896   \exp_args:NNx \exp_args:Nnx \__stex_inputs_mhinput:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2} }
1897 }
1898
1899 \cs_new_protected:Nn \__stex_inputs_inputref_html:nn {
1900   \str_clear:N \l_tmpa_str
1901   \prop_get:NnNF \l_stex_current_archive_prop { narr } \l_tmpa_str {
1902     \prop_get:NnNF \l_stex_current_archive_prop { ns } \l_tmpa_str {}
1903   }

```

```

1904 \tl_if_empty:nTF{ #1 }{
1905   \IfFileExists{#2}{
1906     \ifvmode\noindent\fi\stex_annotation_invisible:nn{shtml:inputref=}
1907     \l_tmpa_str / #2
1908   }{}}
1909 }{
1910   \stex_input_with_hooks:n{#2}
1911 }
1912 }{
1913 \IfFileExists{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }{
1914   \ifvmode\noindent\fi\stex_annotation_invisible:nn{shtml:inputref=}
1915   \l_tmpa_str / #2
1916 }{}}
1917 }{
1918   \input{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1919 }
1920 }
1921 }
1922
1923 \cs_new_protected:Nn \__stex_inputs_inputref_pdf:nn {
1924   \begingroup
1925     \inputreftrue
1926     \tl_if_empty:nTF{ #1 }{
1927       \stex_input_with_hooks:n{#2}
1928     }{
1929       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1930     }
1931   \endgroup
1932 }
1933
1934 \cs_new_protected:Nn \__stex_inputs_inputref:nn {
1935   \stex_in_archive:nn {#1} {
1936     \stex_if_html_backend:TF
1937       \__stex_inputs_inputref_html:nn
1938       \__stex_inputs_inputref_pdf:nn
1939       {##1}{#2}
1940   }
1941 }
1942
1943 \NewDocumentCommand \inputref { O{} m }{
1944   \exp_args:NNx \exp_args:Nnx \__stex_inputs_inputref:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2} }
1945 }

```

(End of definition for `\inputref`, `\mhinput`, and `\ifinputref`. These functions are documented on page 73.)

\addmhbibresource

```

1946 \cs_new_protected:Nn \__stex_inputs_bibresource:n {
1947   \__stex_inputs_up_archive:nn{#1}{bib}
1948   \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
1949     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\addmhbibresource}{#1.bib}
1950   }{
1951     \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
1952       \addbibresource{ ##1 }

```

```

1953     }
1954   }
1955 }
1956 \newcommand\addmhbibresource[2][]{%
1957   \tl_if_empty:nTF{#1}{%
1958     \__stex_inputs_bibresource:n{#2}%
1959   }{%
1960     \stex_in_archive:nn{#1}{\__stex_inputs_bibresource:n{#2}}%
1961   }%
1962 }

```

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

\IfInputref

```

1963 \stex_if_html_backend:TF{%
1964   \newcommand \IfInputref[2]{%
1965     \stex_annotate:nn{shtml:ifinputref=true}{#1}%
1966     \stex_annotate:nn{shtml:ifinputref=false}{#2}%
1967   }%
1968 }{%
1969   \newcommand \IfInputref[2]{%
1970     \ifinputref #1 \else #2 \fi%
1971   }%
1972 }

```

(End of definition for `\IfInputref`. This function is documented on page 73.)

\libinput

```

1973 \cs_new_protected:Nn \__stex_inputs_up_archive:nn {%
1974   \prop_if_exist:NF \l_stex_current_archive_prop {%
1975     \msg_error:nnn{stex}{error/notinarchive}\libinput%
1976   }%
1977   \prop_get:NnNF \l_stex_current_archive_prop {id} \l__stex_inputs_id_str {%
1978     \msg_error:nnn{stex}{error/notinarchive}\libinput%
1979   }%
1980   \seq_clear:N \l__stex_inputs_libinput_files_seq%
1981   \seq_set_eq:NN \l__stex_inputs_path_seq \c_stex_mathhub_file%
1982   \seq_set_split:NnV \l__stex_inputs_id_seq / \l__stex_inputs_id_str%
1983 %
1984   \bool_while_do:nn { ! \seq_if_empty_p:N \l_stex_inputs_id_seq }{%
1985     \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / meta-i}%
1986     \IfFileExists{ \l__stex_inputs_path_str }{%
1987       \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_s%
1988         \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str%
1989       }%
1990     }{}%
1991     \seq_pop_left:NN \l__stex_inputs_id_seq \l__stex_inputs_path_str%
1992     \seq_put_right:No \l__stex_inputs_path_seq \l__stex_inputs_path_str%
1993   }%
1994 %
1995   \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / lib / #1}%
1996   \IfFileExists{ \l__stex_inputs_path_str }{%
1997     \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_s%
1998       \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str%
1999     }%

```

```

2000     }{}
2001 }
2002
2003 \cs_new_protected:Nn \__stex_inputs_libinput:n {
2004     \__stex_inputs_up_archive:nn{#1}{tex}
2005     \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
2006         \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
2007     }{
2008         \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
2009             \input{ ##1 }
2010         }
2011     }
2012 }
2013
2014 \newcommand \libinput [2] [] {
2015     \tl_if_empty:nTF{#1} {
2016         \__stex_inputs_libinput:n{#2}
2017     }{
2018         \stex_in_archive:nn{#1}{\__stex_inputs_libinput:n{#2}}
2019     }
2020 }

```

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

\libusepackage

```

2021 \newcommand\libusepackage[2] [] {
2022     \__stex_inputs_up_archive:nn{#2}{sty}
2023     \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2024         \str_set:Nx \l__stex_inputs_tmp_str {\seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
2025         \exp_args:Nne \use:n {\usepackage[#1]} {
2026             \str_range:Nnn\l__stex_inputs_tmp_str 1 {-5}
2027         }
2028     }{
2029         \stex_fatal_error:nnn{error/nofile}{\libusepackage}{#1.sty}
2030     }
2031 }

```

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

```

\mhgraphics
\cmhgraphics
\lstinputmhlisting
\clstinputmhlisting
\mhtikzinput
\cmhtikzinput
2032 \str_new:N \l__stex_inputs_gin_repo_str
2033 \ltx@ifpackageloaded{graphicx}{\use:n}{\AtEndOfPackageFile{graphicx}}{
2034     \define@key{Gin}{archive}{
2035         \tl_set:Nx\Gin@mrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2036     }
2037     \providecommand\mhgraphics[2][]{%
2038         \tl_set:Nx\Gin@mrepos{
2039             \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id}
2040         }
2041         \setkeys{Gin}{#1}
2042         \includegraphics[#1]{ \Gin@mrepos #2 }
2043     }
2044     \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
2045 }
2046

```

```

2047 \ltx@ifpackageloaded{listings}{\use:n}{\AtEndOfPackageFile{listings}}{
2048   \define@key{lst}{archive}{
2049     \def\lst@mrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2050   }
2051 \newcommand\lstinputmhlisting[2][]{%
2052   \def\lst@mrepos{
2053     \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id}
2054   }
2055   \setkeys{lst}{#1}%
2056   \lstinputlisting[#1]{\lst@mrepos #2}}
2057 \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
2058 }
2059
2060 \ltx@ifpackageloaded{tikzinput}{\use:n}{\AtEndOfPackageFile{tikzinput}}{
2061   \define@key{Gin}{archive}{
2062     \str_set:Nn \l__stex_inputs_gin_repo_str {#1}
2063     \def\Gin@mrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2064   }
2065 \newcommand\mhtikzinput[2][]{%
2066   \str_clear:N \l__stex_inputs_gin_repo_str
2067   \def\Gin@mrepos{
2068     \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id}
2069   }
2070   \setkeys{Gin}{#1}%
2071   \exp_args:No \stex_in_archive:nn \l__stex_inputs_gin_repo_str {
2072     \tikzinput[#1]{\Gin@mrepos #2}
2073   }
2074 }
2075 \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
2076 }

```

(End of definition for `\mhgraphics` and others. These functions are documented on page 73.)

`\libusetikzlibrary`

```

2077 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary_i:nn {
2078   \pgfkeys@spdef\pgf@temp{#1}
2079   \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
2080   \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfutil@empty
2081   \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode`@}
2082   \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode`\|}%
2083   \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode`\$}
2084   \catcode`\@=11
2085   \catcode`\|=12
2086   \catcode`\$=3
2087   \pgfutil@InputIfFileExists{#2}{}{%
2088     \catcode`\@=\csname tikz@library@#1@atcode\endcsname
2089     \catcode`\|= \csname tikz@library@#1@barcode\endcsname
2090     \catcode`\$= \csname tikz@library@#1@dollarcode\endcsname
2091   }
2092
2093 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary:n{
2094   \__stex_inputs_up_archive:nn{tikzlibrary#1}{code.tex}
2095   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2096     \exp_args:Nne \__stex_inputs_usetikzlibrary_i:nn{#1}{\seq_item:Nn \l__stex_inputs_libin

```

```

2097 }{
2098   \stex_fatal_error:n{error/nofile}{\libusetikzlibrary}{tikzlibrary#1.code.tex}
2099 }
2100 }
2101 \newcommand \libusetikzlibrary [2] [] {
2102   \cs_if_exist:N \usetikzlibrary {
2103     \msg_error:nnx{stex}{error/notikz}{\tl_to_str:n{\libusetikzlibrary}}
2104   }
2105   \tl_if_empty:nTF{#1}{
2106     \__stex_inputs_usetikzlibrary:n{#2}
2107   }{
2108     \stex_in_archive:nn{#1}{\__stex_inputs_usetikzlibrary:n{#2}}
2109   }
2110 }
2111 }

```

(End of definition for `\libusetikzlibrary`. This function is documented on page 108.)

13.5 SMS Mode

```
2112 <@=stex_smsmode>
```

Macros and environments allowed in sms mode:

```
2113 \tl_new:N \g__stex_smsmode_allowed_tl
2114 \tl_new:N \g__stex_smsmode_allowed_escape_tl
2115 \seq_new:N \g__stex_smsmode_allowedenvs_seq
```

```
\stex_sms_allow:N
```

```
\stex_sms_allow_escape:N
```

```
\stex_sms_allow_env:n
```

```
2116 \cs_new_protected:Nn \stex_sms_allow:N {
2117   \tl_gput_right:Nn \g__stex_smsmode_allowed_tl {#1}
2118 }
2119 \cs_new_protected:Nn \stex_sms_allow_escape:N {
2120   \tl_gput_right:Nn \g__stex_smsmode_allowed_escape_tl {#1}
2121 }
2122 \cs_new_protected:Nn \stex_sms_allow_env:n {
2123   \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowedenvs_seq {\tl_to_str:n{#1}}
2124 }
```

(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 125.)

Some initial allowed macros:

```
2127 \stex_sms_allow:N \makeatletter
2128 \stex_sms_allow:N \makeatother
2129 \stex_sms_allow:N \ExplSyntaxOn
2130 \stex_sms_allow:N \ExplSyntaxOff
2131 \stex_sms_allow:N \rustexBREAK
```

```
\stex_if_smsmode_p:
```

```
\stex_if_smsmode:TF
```

```
2132 \bool_new:N \g__stex_smsmode_bool
2133 \bool_set_false:N \g__stex_smsmode_bool
2134 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2135   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2136 }
```

(End of definition for \stex_if_smsmode:TF. This function is documented on page 125.)

```
\stex_sms_allow_import:Nn
  \stex_sms_allow_import_env:nn
    2137  \tl_new:N \g__stex_smsmode_allowed_import_tl
    2138  \seq_new:N \g__stex_smsmode_allowed_import_env_seq
    2139  \cs_new_protected:Nn \stex_sms_allow_import:Nn {
    2140    \tl_gput_right:Nn \g__stex_smsmode_allowed_import_tl {#1}
    2141    \tl_gset:cn{\tl_to_str:n{#1}~~~smsmode} {#2}
    2142  }
    2143  \cs_new_protected:Nn \stex_sms_allow_import_env:nn {
    2144    \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowed_import_env_seq {\tl_to_str:n{#1}
    2145      \tl_gset:cn{\tl_to_str:n{#1}~~~env~~~smsmode} {#2}
    2146  }
    2147
    2148  \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 126.)

```
\stex_file_in_smsmode:nn
\stex_file_in_smsmode:on
  2149  \cs_new_protected:Nn \__stex_smsmode_in_smsmode:n { \stex_suppress_html:n {
  2150    \vbox_set:Nn \l_tmpa_box {
  2151      \bool_set_true:N \g__stex_smsmode_bool
  2152      \bool_set_false:N \stex_html_do_output_bool
  2153      #1
  2154    }
  2155    \%box_clear:N \l_tmpa_box
  2156  } }
  2157
  2158  \quark_new:N \q__stex_smsmode_break
  2159
  2160  \cs_new_protected:Nn \__stex_smsmode_start_smsmode:n {
  2161    \everyeof{\q__stex_smsmode_break\exp_not:N}
  2162    \let\stex_smsmode_do:\__stex_smsmode_smsmode_do:
  2163    \exp_after:wN \exp_after:wN \exp_after:wN
  2164    \stex_smsmode_do:
  2165    \cs:w @ input\cs_end: "#1" \relax
  2166  }
  2167
  2168  \cs_new_protected:Nn \stex_file_in_smsmode:nn {
  2169    \seq_gclear:N \l__stex_smsmode_importmodules_seq
  2170    \seq_gclear:N \l__stex_smsmode_sigmodules_seq
  2171    \tl_clear:N \g_stex_sms_import_code
  2172    \group_begin:
  2173      \let \l_stex_metatheory_uri \c_stex_default_metatheory
  2174      \cs_set:Npn \stex_check_term:n {#1 {}}
  2175      \seq_clear:N \l_stex_all_modules_seq
  2176      \str_clear:N \l_stex_current_module_str
  2177      #2
  2178      \stex_filestack_push:n{#1}
  2179      \__stex_smsmode_in_smsmode:n {
  2180        \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_imports:N
  2181        \tl_map_inline:Nn \g__stex_smsmode_allowed_import_tl {
  2182          \use:c{\tl_to_str:n{##1}~~~smsmode}
```

```

2183     }
2184     \seq_map_inline:Nn \g__stex_smsmode_allowed_import_env_seq {
2185         \use:c{\tl_to_str:n{##1}~env~~~smsmode}
2186     }
2187     \__stex_smsmode_start_smsmode:n{#1}
2188 }
2189 \__stex_smsmode_in_smsmode:n \g_stex_sms_import_code
2190 \__stex_smsmode_in_smsmode:n {
2191     \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_normal:N
2192     \__stex_smsmode_start_smsmode:n{#1}
2193 }
2194 \stex_filestack_pop:
2195 \group_end:
2196 }
2197 \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 125.)

`\stex_smsmode_do:`

```

2198 \cs_new_protected:Nn \__stex_smsmode_smsmode_do: {
2199     \% \stex_if_smsmode:T {
2200         \__stex_smsmode_do:w
2201     %}
2202 }
2203 \let\stex_smsmode_do:\relax
2204
2205
2206 \cs_new:Nn \__stex_smsmode_check_cs:NNn {
2207     \exp_after:wN\if\exp_after:wN\relax\exp_not:N#3
2208     \exp_after:wN#1\exp_after:wN#3\else
2209     \exp_after:wN#2\fi
2210 }
2211
2212 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2213     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2214         \__stex_smsmode_check_cs:NNn \__stex_smsmode_do_aux:N \__stex_smsmode_do:w { #1 }
2215     }{
2216         \__stex_smsmode_do:w
2217     }
2218 }
2219
2220 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2221     \cs_if_eq:NNF #1 \q__stex_smsmode_break {
2222         \__stex_smsmode_do_aux_curr:N #1
2223     }
2224 }
2225
2226 \cs_new_protected:Nn \__stex_smsmode_do_aux_imports:N {
2227     \% \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~import}
2228     \tl_if_in:NnTF \g__stex_smsmode_allowed_import_tl {#1} {
2229         \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}~in~import}
2230         #1
2231     }{
2232         \cs_if_eq:NNTF \begin{ #1 {

```

```

2233     \__stex_smsemode_check_begin:Nn \g__stex_smsemode_allowed_import_env_seq
2234     }{
2235         \cs_if_eq:NNTF \end #1 {
2236             \__stex_smsemode_check_end:Nn \g__stex_smsemode_allowed_import_env_seq
2237         }{
2238             \__stex_smsemode_do:w
2239         }
2240     }
2241 }
2242 }
2243
2244 \cs_new_protected:Nn \__stex_smsemode_do_aux_normal:N {
2245     % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}-in-sms-mode}
2246     \tl_if_in:NnTF \g__stex_smsemode_allowed_tl {#1} {
2247         \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}}
2248         #1\__stex_smsemode_do:w
2249     }{
2250         \tl_if_in:NnTF \g__stex_smsemode_allowed_escape_tl {#1} {
2251             \stex_debug:nn{sms}{Executing~escaped~\tl_to_str:n{#1}}
2252             #1
2253         }{
2254             \cs_if_eq:NNTF \begin #1 {
2255                 \__stex_smsemode_check_begin:Nn \g__stex_smsemode_allowedenvs_seq
2256             }{
2257                 \cs_if_eq:NNTF \end #1 {
2258                     \__stex_smsemode_check_end:Nn \g__stex_smsemode_allowedenvs_seq
2259                 }{
2260                     \__stex_smsemode_do:w
2261                 }
2262             }
2263         }
2264     }
2265 }
2266
2267 \cs_new_protected:Nn \__stex_smsemode_check_begin:Nn {
2268     % \stex_debug:nn{sms}{Checking~environment~#2}
2269     \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2270         \stex_debug:nn{sms}{Environment~#2}
2271         \begin{#2}
2272     }{
2273         \__stex_smsemode_do:w
2274     }
2275 }
2276 \cs_new_protected:Nn \__stex_smsemode_check_end:Nn {
2277     % \stex_debug:nn{sms}{Checking~end~environment~#2}
2278     \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2279         \stex_debug:nn{sms}{End-Environment~#2}
2280         \end{#2}\__stex_smsemode_do:w
2281     }{
2282         \%str_if_eq:nnTF{#2}{document} \endinput
2283         \__stex_smsemode_do:w
2284     }
2285 }

```

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

13.6 Modules

13.6.1 The smodule-environment

2286 `<@@=stex_modules>`

`\l_stex_current_module_str` The current module:

2287 `\str_new:N \l_stex_current_module_str`

(End of definition for `\l_stex_current_module_str`. This variable is documented on page 112.)

`\l_stex_all_modules_seq` Stores all modules currently in scope

2288 `\seq_new:N \l_stex_all_modules_seq`

(End of definition for `\l_stex_all_modules_seq`. This variable is documented on page 112.)

`\stex_every_module:n`

2289 `<@@=stex_module_setup>`
2290 `\tl_clear:N \g_stex_every_module_tl {`
2291 `}`
2292 `\cs_new_protected:Nn \stex_every_module:n {`
2293 `\tl_gput_right:Nn \g_stex_every_module_tl { #1 }`
2294 `}`

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

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

2295 `\cs_new_protected:Npn \stex_module_setup:n {`
2296 `\stex_if_in_module:TF __stex_module_setup_nested:n __stex_module_setup_top:n`
2297 `}`
2298 `\cs_new_protected:Nn __stex_module_setup_top:n {`
2299 `__stex_module_setup_get_uri_str:n{#1}`
2300 `\stex_debug:nn{module}{Module~URI:~\l__stex_module_setup_ns_str?#1}`
2301 `\str_if_empty:NTF \l_stex_key_sig_str`
2302 `\stex_module_setup_top_nosig:n __stex_module_setup_top_sig:n {\l__stex_module_setu`
2303 `\stex_metagroup_new:o \l_stex_current_module_str`
2304 `\g_stex_every_module_tl`
2305 `\stex_execute_in_module:x {`
2306 `\stex_do_deprecation:n{#1}`
2307 `}`
2308 `__stex_module_setup_load_meta:`
2309 `}`
2310 `__stex_module_setup_top_nosig:n {`
2311 `\stex_if_module_exists:nTF{#1}{`
2312 `\stex_debug:nn{modules}{(already exists)}`
2313 `}{`
2314 `\tl_gclear:c{\c_stex_module_ #1 _code}`
2315 `\prop_gclear:c{\c_stex_module_ #1 _morphisms_prop }`
2316 `\prop_gclear:c{\c_stex_module_ #1 _symbols_prop }`
2317 `\prop_gclear:c{\c_stex_module_ #1 _notations_prop }`
2318 `}`
2319 `\str_set:Nx \l_stex_current_module_str {#1}`
2320 `\seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str`

```

2323 }
2324
2325 \cs_new_protected:Nn \__stex_module_setup_top_sig:n {
2326   \stex_if_module_exists:nTF{#1} {
2327     \stex_debug:nn{modules}{(already exists)}
2328   }{
2329     \stex_debug:nn{modules}{(needs loading)}
2330     \__stex_module_setup_load_sig:
2331   }
2332 \% \stex_if_smsmode:F { % WHY?
2333   \stex_activate_module:x {
2334     #1
2335   }
2336 %}
2337 \str_set:Nx\l_stex_current_module_str{#1}
2338 }
2339
2340 \cs_new_protected:Nn \__stex_module_setup_load_sig: {
2341   \stex_file_split_off_ext:NN \l__stex_module_setup_sigfile \g_stex_current_file
2342   \stex_file_split_off_lang:NN \l__stex_module_setup_sigfile \l__stex_module_setup_sigfile
2343   \exp_args:Ne \stex_file_in_smsmode:nn {
2344     \stex_file_use:N \l__stex_module_setup_sigfile . \l_stex_key_sig_str . tex
2345   }{}
2346 }
2347
2348 \cs_new_protected:Nn \__stex_module_setup_setup_nested:n {
2349   \exp_after:wN
2350     \__stex_module_setup_split_module:n \l_stex_current_module_str \__stex_module_setup_end:
2351   \stex_debug:nn{module}{Nested-Module~URI:~\l_stex_current_module_str}
2352   \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
2353   \stex_metagroup_new:o \l_stex_current_module_str
2354 }
2355
2356
2357 \cs_new_protected:Nn \__stex_module_setup_get_uri_str:n {
2358   \str_clear:N \l__stex_module_setup_ns_str
2359   \stex_map_uri:Nnnnn \l_stex_current_ns_uri {
2360     \str_set:Nx \l__stex_module_setup_ns_str{##1\c_colon_str/}
2361   }{
2362     \seq_set_split:Nnn \l__stex_module_setup_seq / {##1}
2363     \seq_pop_right:NN \l__stex_module_setup_seq \l__stex_module_setup_seg
2364     \exp_args:No \str_if_eq:nnF \l__stex_module_setup_seg {#1} {
2365       \seq_put_right:No \l__stex_module_setup_seq \l__stex_module_setup_seg
2366     }
2367     \tl_put_right:Nx \l__stex_module_setup_ns_str {\seq_use:Nn \l__stex_module_setup_seq /}
2368   }{}{}
2369 }
2370
2371 \cs_new_protected:Npn \__stex_module_setup_split_module:n #1#2 \__stex_module_setup_end: #3
2372   \stex_module_setup_top_nosig:n { #1 ? #2 / #3}
2373 }
2374
2375 \bool_new:N \l_stex_in_meta_bool
2376 \bool_set_false:N \l_stex_in_meta_bool

```

```

2377 \cs_new_protected:Nn \__stex_module_setup_load_meta: {
2378   \tl_if_empty:NF \l_stex_metatheory_uri {
2379     \stex_execute_in_module:x{
2380       \stex_pseudogroup_with:nn{\l_stex_in_meta_bool} {
2381         \stex_activate_module:n {\l_stex_uri_use:N \l_stex_metatheory_uri }
2382       }
2383     }
2384   }
2385 }
2386 }
2387
2388 (@@=stex_modules)

```

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

`\stex_close_module:`

```

2389 \cs_new:Nn \stex_close_module: {
2390   \bool_if:NT \c_stex_persist_write_mode_bool \__stex_modules_persist_module:
2391   \stex_debug:nn{module} {
2392     Closing~module~\l_stex_current_module_str^~J
2393     Code:~\expandafter\meaning\csname c_stex_module_\l_stex_current_module_str _code\endcsna
2394     Imports:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_str
2395     Declarations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_
2396     Notations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_
2397   }
2398 }
2399
2400 \cs_new_protected:Nn \__stex_modules_persist_module: {
2401   \stex_persist:e {
2402     \__stex_modules_restore_module:nnnn {\l_stex_current_module_str} {
2403       \exp_after:wN \prop_to_keyval:N \cs:w
2404         c_stex_module_\l_stex_current_module_str _morphisms_prop
2405       \cs_end:
2406     }{
2407       \exp_after:wN \prop_to_keyval:N \cs:w
2408         c_stex_module_\l_stex_current_module_str _symbols_prop
2409       \cs_end:
2410     }{
2411       \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
2412       \exp_after:wN \exp_after:wN \exp_after:wN
2413       { \cs:w c_stex_module_\l_stex_current_module_str _code \cs_end: }
2414     }{}
2415     \prop_map_function:cN{c_stex_module_\l_stex_current_module_str _notations_prop}
2416       \__stex_modules_persist_nots_i:nn
2417     \exp_not:N \STEXRestoreNotsEnd {}
2418   }
2419 }
2420
2421 \cs_new_protected:Nn \__stex_modules_restore_module:nnnn {
2422   \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_morphisms_prop}{#2}
2423   \cs_set:Npn \__stex_modules_tl {#3}
2424   \exp_args:Nno \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}\__stex_
2425   \prop_map_inline:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}{%
2426     \stex_ref_new_symbol:n{#1?##1}

```

```

2427 }
2428 \cs_gset:cpn{c_stex_module_ \tl_to_str:n{#1}_code}{#4}
2429 \prop_gclear:c{c_stex_module_ \tl_to_str:n{#1} _notations_prop}
2430 \str_set:Nn \l_stex_modules_restore_mod_str {#1}
2431 \group_begin:
2432   \catcode`_=8\relax
2433   \catcode`:=12\relax
2434   \__stex_modules_restore_nots:n
2435 }
2436
2437 \cs_new:Nn \__stex_modules_persist_nots_i:nn {
2438   \exp_not:n{#2}
2439 }
2440
2441 \quark_new:N \STEXRestoreNotsEnd
2442
2443 \cs_new_protected:Nn \__stex_modules_restore_nots:n {
2444   \__stex_modules_restore_nots_i:n
2445 }
2446
2447 \cs_new_protected:Nn \__stex_modules_restore_nots_i:n {
2448   \tl_if_eq:nnTF{#1}{\STEXRestoreNotsEnd}{
2449     \group_end:
2450   }{
2451     \__stex_modules_restore_nots_ii:nnnn {#1}
2452   }
2453 }
2454
2455 \cs_new_protected:Nn \__stex_modules_restore_nots_ii:nnnn {
2456   \cs_set:Npn \l_stex_modules_tl {{#4}{#5}}
2457   \exp_args:NNe\use:nn\prop_gput:cnn{
2458     {c_stex_module_ \l_stex_modules_restore_mod_str _notations_prop}
2459     {\tl_to_str:n{#1!#2}}{
2460       {\tl_to_str:n{#1}}{\tl_to_str:n{#2}}{#3}
2461       \exp_args:No \exp_not:n \l_stex_modules_tl
2462     }
2463   }
2464   \__stex_modules_restore_nots_i:n
2465 }

```

(End of definition for `\stex_close_module`. This function is documented on page [112](#).)

`\l_stex_metatheory_uri`

```
2466 \tl_new:N \l_stex_metatheory_uri
```

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

`\setmetatheory`

```

2467 \cs_new_protected:Nn \__stex_modules_set_matatheory:nn {
2468   \group_begin:
2469   \stex_debug:nn{metatheory}{Setting-metatheory~[#1]#2}
2470   \stex_import_module_uri:nn { #1 } { #2 }
2471   \stex_debug:nn{metatheory}{Here:^^J
2472     \l_stex_import_archive_str^^J
2473     \l_stex_import_path_str^^J

```

```

2474     \l_stex_import_name_str^^J
2475   }
2476 \stex_import_require_module:ooo
2477   \l_stex_import_archive_str
2478   \l_stex_import_path_str
2479   \l_stex_import_name_str
2480 \stex_debug:nn{metatheory}{Found:~\l_stex_import_ns_str}
2481 \exp_args:Nne \use:nn {
2482   \group_end: \stex_uri_resolve:Nn \l_stex_metalanguage_uri
2483 }{\l_stex_import_ns_str}
2484 }
2485
2486 \NewDocumentCommand \setmetatheory {O{} m} {
2487   \__stex_modules_set_metalanguage:nn {#1}{#2}
2488   \stex_smsmode_do:
2489 }
2490 \stex_sms_allow_escape:N \setmetatheory

```

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

Keys and key handling:

```

2491 \stex_keys_define:nnnn{smodule}{
2492   \str_clear:N \l_stex_key_sig_str
2493 }{
2494   meta .code:n = {
2495     \str_if_empty:nTF {#1} {
2496       \tl_clear:N \l_stex_metalanguage_uri
2497     }{
2498       \stex_uri_resolve:Nx \l_stex_metalanguage_uri {#1}
2499     }
2500   },
2501   ns .code:n = {
2502     \stex_uri_resolve:Nx \l_stex_current_ns_uri {#1}
2503   },
2504   lang .code:n = {
2505     \stex_set_language:n {#1}
2506   },
2507   sig .str_set_x:N = \l_stex_key_sig_str ,
2508   creators .code:n = {} , % todo ?
2509   contributors .code:n = {} , % todo ?
2510   srccite .code:n = {} % todo ?
2511 }{id, title, style, deprecate}

```

smodule (env.)

```

2512 \stex_new_stylable_env:nnnnnnn {module} {O{} m} {
2513   \stex_keys_set:nn { smodule }{ #1 }
2514   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
2515   \tl_if_empty:NF \thistitle {
2516     \exp_args:No \stexdoctitle \thistitle
2517   }
2518   \exp_args:Nx \stex_module_setup:n { \tl_to_str:n{ #2 } }
2519
2520   \stex_if_do_html:T {
2521     \exp_args:Nne \begin{stex_annotation_env} {
2522       shtml:theory={\l_stex_current_module_str},

```

```

2523     shtml:language={ \l_stex_current_language_str},
2524     shtml:signature={\l_stex_key_sig_str}
2525     \tl_if_empty:NF \l_stex_metatheory_uri {
2526         shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
2527     }
2528 }
2529 \stex_annotation_invisible:n{}
2530 }
2531 \stex_if_smsmode:F {
2532     \str_set_eq:NN \thismoduleuri \l_stex_current_module_str
2533     \tl_set:Nn \thismodulename {#2}
2534     \stex_style_apply:
2535 }
2536 \stex_smsmode_do:
2537 }{
2538     \stex_close_module:
2539     \stex_if_smsmode:F \stex_style_apply:
2540     \stex_if_do_html:T{ \end{stex_annotation_env} }
2541 }{}{}{s}
2542
2543 \stex_sms_allow_env:n{smodule}

```

\stex_if_in_module:p: Are we currently in a module?

\stex_if_in_module:TF

```

2544 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
2545     \str_if_empty:NTF \l_stex_current_module_str
2546     \prg_return_false: \prg_return_true:
2547 }

```

(End of definition for `\stex_if_in_module:TF`. This function is documented on page 112.)

\stex_if_module_exists_p:n Does a module with this URI exist?

\stex_if_module_exists:nTF

```

2548 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
2549     \tl_if_exist:cTF { c_stex_module_#1_code }
2550     \prg_return_true: \prg_return_false:
2551 }

```

(End of definition for `\stex_if_module_exists:nTF`. This function is documented on page 112.)

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

```

2552 \cs_new_protected:Nn \stex_do_up_to_module:n {
2553     \exp_args:No \stex_metagroup_do_in:nn \l_stex_current_module_str {#1}
2554 }
2555 \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 113.)

\stex_module_add_code:n

\stex_module_add_code:x

```

2556 \cs_new_protected:Nn \stex_module_add_code:n {
2557     \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
2558 }
2559 \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 113.)

```

\stex_execute_in_module:n
\stex_execute_in_module:x 2560 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:TF {
2561   \stex_module_add_code:n { #1 }
2562   \stex_do_up_to_module:n { #1 }
2563 }{ #1 }
2564 \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 112.)

\STEXexport

```

2565 \NewDocumentCommand \STEXexport {} {
2566   \ExplSyntaxOn
2567   \__stex_modules_export:n
2568 }
2569 \cs_new_protected:Nn \__stex_modules_export:n {
2570   \stex_ignore_spaces_and_pars:#1\ExplSyntaxOff
2571   \stex_module_add_code:n { \stex_ignore_spaces_and_pars:#1}
2572   \stex_smsmode_do:
2573 }

```

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

```

2574 \stex_deactivate_macro:Nn \STEXexport {module~environments}
2575 \stex_sms_allow_escape:N \STEXexport
2576 \stex_every_module:n {\stex_reactivate_macro:N \STEXexport}

```

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

\stex_module_add_morphism:nnnn

```

\stex_module_add_morphism:nonn
\stex_module_add_morphism:ooox
2577 \cs_new_protected:Nn \stex_module_add_morphism:nnnn {
2578   \exp_args:Nne \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _morphisms_prop}{%
2579     \tl_if_empty:nTF{#1}{[#2]}{#1}
2580   }{#1}{#2}{#3}{#4}
2581 }
2582 \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 120.)

\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
2583 \cs_new_protected:Nn \stex_module_add_symbol:nnnnnnN {
2584   \stex_debug:nn{declaration}{New~declaration:\l_stex_current_module_str?#2^J
2585     Macro:#1^JArity:#3~(#4)^J
2586     Def:~\tl_to_str:n{#5}^J
2587     Type:~\tl_to_str:n{#6}^J
2588     Returns:~\tl_to_str:n{#7}
2589   }
2590   \%prop_gput:cnx{c_stex_module_\l_stex_current_module_str _symbols_prop}
2591   %{#2}{\exp_not:n{#1}{#2}{#3}{#4}{#5}{#6}{#7}\exp_not:n{#8}}

```

```

2592 \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
2593 {##2}{##1}{##2}{##3}{##4}{##5}{##6}{##7}{##8}}
2594 \tl_if_empty:nF{#1} {
2595   \stex_execute_in_module:n {
2596     \__stex_modules_activate_sym:n {#2}
2597   }
2598 }
2599 }
2600
2601 \cs_new_protected:Nn \__stex_modules_activate_sym:n {
2602   \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _symbols_prop}{%
2603     \str_if_eq:nnT{#1}{##1}{%
2604       \__stex_modules_activate_i:nnnnnnnn ##2
2605     }
2606   }
2607 }
2608 \cs_new_protected:Nn \__stex_modules_activate_i:nnnnnnnn {
2609   \stex_debug:nn{activating}{#1:\l_stex_current_module_str^~J}
2610   \tl_to_str:n{##2}{##3}{##4}{##5}{##6}{##7}{##8}
2611 }
2612 \cs_set:cp{#1} {
2613   \stex_invoke_symbol:nnnnnnnN
2614   {\l_stex_current_module_str}
2615   \exp_not:n{##2}{##3}{##4}{##5}{##6}{##7}{##8}
2616 }
2617 \stex_debug:nn{activating}{done}
2618 \prop_map_break:
2619 }

```

(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

2620 \cs_new_protected:Nn \stex_module_add_notation:nnnnn {
2621   \stex_debug:nn{notations}{Adding~notation:~^~J}
2622   #1~\c_hash_str#2~#3~^~J
2623   to~\l_stex_current_module_str
2624 }
2625 \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _notations_prop}
2626 {##1!##2}{##1}{##2}{##3}{##4}{##5}
2627 \stex_execute_in_module:n {
2628   \__stex_modules_activate_not:nn{##1}{##2}
2629 }
2630 }
2631 \cs_generate_variant:Nn \stex_module_add_notation:nnnnn {eoeoo}
2632
2633
2634 \cs_new_protected:Nn \__stex_modules_activate_not:nn {
2635   \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _notations_prop}{%
2636     \str_if_eq:nnT{##1!##2}{##1}{%
2637       \prop_map_break:n{\stex_set_notation_macro:nnnnn ##2 } }}
```

```

2638     }
2639   }
2640 }
```

(End of definition for \stex_module_add_notation:nnnnn and \stex_set_notation_macro:nnnnn. These functions are documented on page 115.)

```
\stex_set_notation_macro:nnnnn
\stex_set_notation_macro:eoexo
2641 \cs_new_protected:Nn \stex_set_notation_macro:nnnnn {
2642   \tl_set:cn {l_stex_notation_#1#2_cs}{#4}
2643   \cs_if_exist:cF{l_stex_notation_#1__cs} {
2644     \tl_set:cn {l_stex_notation_#1__cs}{#4}
2645   }
2646   \tl_if_empty:nF{#5} {
2647     \tl_set:cn{l_stex_notation_#1_op_#2_cs}{#5}
2648     \cs_if_exist:cF{l_stex_notation_#1_op__cs} {
2649       \cs_set_eq:cc {l_stex_notation_#1_op__cs}{l_stex_notation_#1_op_#2_cs}
2650     }
2651   }
2652 }
2653 \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 116.)

```
\stex_activate_module:n
\stex_activate_module:o
\stex_activate_module:x
2654 \cs_new_protected:Nn \stex_activate_module:n {
2655   \seq_if_in:Nnf \l_stex_all_modules_seq { #1 } {
2656     \stex_debug:nnf{modules}{Activating-module~#1^J\expandafter\meaning\csname c_stex_module
2657     \seq_put_right:Nn \l_stex_all_modules_seq { #1 }
2658     \stex_pseudogroup:nn{
2659       \str_set:Nn \l_stex_current_module_str {#1}
2660       \use:c{ c_stex_module_#1_code }
2661     }{
2662       \stex_pseudogroup_restore:N \l_stex_current_module_str
2663     }
2664   }
2665 }
2666 \cs_generate_variant:Nn \stex_activate_module:n {o,x}
```

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

Iterating:

```
2667 <@=stex_iterate>
```

\stex_iterate_symbols:n

```
2668 \cs_new_protected:Nn \stex_iterate_symbols:n {
2669   \stex_pseudogroup_with:nn{\_\stex_iterate_sym cs:nnnnnnnnN\stex_iterate_break:\stex_iterat
2670   \cs_set:Npn \_\stex_iterate_sym cs:nnnnnnnnN
2671   ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
2672   \cs_set:Npn \stex_iterate_break: {
2673     \prop_map_break:n{\seq_map_break:}
2674   }
2675   \cs_set:Npn \stex_iterate_break:n ##1 {
2676     \prop_map_break:n{\seq_map_break:n{##1}}
2677 }
```

```

2678     \seq_map_inline:Nn \l_stex_all_modules_seq {
2679         \prop_map_inline:cn{c_stex_module_##1_symbols_prop}{
2680             \__stex_iterate_sym_cs:nnnnnnnnN {##1} ####2
2681         }
2682     }
2683 }
2684 }
```

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

\stex_iterate_symbols:nn

```

2685 \cs_new_protected:Nn \stex_iterate_symbols:nn {
2686     \seq_clear:N \l_stex_iterate_mods_seq
2687     \stex_pseudogroup_with:nn{\__stex_iterate_sym_cs:nnnnnnnnN}{
2688         \cs_set:Npn \__stex_iterate_sym_cs:nnnnnnnnN
2689             ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #2 }
2690         \clist_map_function:nN {#1} \__stex_iterate_it_decl_i:n
2691     }
2692 }
2693
2694 \cs_new_protected:Nn \__stex_iterate_it_decl_i:n {
2695     \seq_if_in:NnF \l_stex_iterate_mods_seq {#1} {
2696         \seq_put_left:Nn \l_stex_iterate_mods_seq {#1}
2697         \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2698             \__stex_iterate_it_decl_check:nnnn ##2
2699         }
2700         \prop_map_inline:cn{c_stex_module_#1_symbols_prop}{
2701             \__stex_iterate_sym_cs:nnnnnnnnN {##1} ##2
2702         }
2703     }
2704 }
2705 \cs_new_protected:Nn \__stex_iterate_it_decl_check:nnnn {
2706     \tl_if_empty:nT{#1}{%
2707         \__stex_iterate_it_decl_i:n {#2}
2708     }
2709 }
```

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

\stex_iterate_notations:nn

```

2710 \cs_new_protected:Nn \stex_iterate_notations:nn {
2711     \seq_clear:N \l_stex_iterate_mods_seq
2712     \stex_pseudogroup_with:nn{\__stex_iterate_not_cs:nnnn}{%
2713         \cs_set:Npn \__stex_iterate_not_cs:nnnn
2714             ##1 ##2 ##3 ##4 ##5 { #2 }
2715         \clist_map_function:nN {#1} \__stex_iterate_it_not_i:n
2716     }
2717 }
2718
2719 \cs_new_protected:Nn \__stex_iterate_it_not_i:n {
2720     \seq_if_in:NnF \l_stex_iterate_mods_seq {#1} {
2721         \seq_put_left:Nn \l_stex_iterate_mods_seq {#1}
2722         \prop_map_inline:cn{c_stex_module_#1_notations_prop}{
2723             \__stex_iterate_not_cs:nnnn ##2
2724         }
2725 }
```

```

2725   \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2726     \__stex_iterate_it_not_check:nnnn ##2
2727   }
2728 }
2729 }
2730 \cs_new_protected:Nn \__stex_iterate_it_not_check:nnnn {
2731   \tl_if_empty:nT{#1}{%
2732     \__stex_iterate_it_not_i:n {#2}
2733   }
2734 }
```

(End of definition for `\stex_iterate_notations:nn`. This function is documented on page 116.)

`\stex_iterate_morphisms:nn`

```

2735 \cs_new_protected:Nn \stex_iterate_morphisms:nn {
2736   \seq_clear:N \l__stex_iterate_mods_seq
2737   \bool_set_true:N \l__stex_iterate_continue_bool
2738   \cs_set:Npn \__stex_iterate_morphism_cs:nnnn ##1 ##2 ##3 ##4 ##5 {
2739     #2
2740     \bool_if:NT \l__stex_iterate_continue_bool {
2741       \str_if_eq:nnTF{##1}{##2}{%
2742         \tl_put_right:Nn \l__stex_iterate_todo_tl {
2743           \__stex_iterate_iterate_morphism:nn{##5}{##2}
2744         }
2745       }{
2746         \tl_put_right:Nn \l__stex_iterate_todo_tl {
2747           \__stex_iterate_iterate_morphism:nn{##5 / ##1}{##2}
2748         }
2749       }
2750     }
2751   }
2752   \cs_set:Npn \stex_iterate_break:n ##1 {
2753     \bool_set_false:N \l__stex_iterate_continue_bool
2754     \prop_map_break:n{##1}
2755   }
2756   \__stex_iterate_iterate_morphism:nn{}{##1}
2757 }
2758
2759 \cs_new_protected:Nn \__stex_iterate_iterate_morphism:nn {
2760   \tl_clear:N \l__stex_iterate_todo_tl
2761   \seq_if_in:NnF \l__stex_iterate_mods_seq {##1 ##2}{%
2762     \seq_put_right:Nn \l__stex_iterate_mods_seq {##1 ##2}
2763     \prop_map_inline:cn{c_stex_module_#2_morphisms_prop}{
2764       \__stex_iterate_morphism_cs:nnnn ##2 {##1}
2765       % TODO
2766       % ##1: name or [mpath]
2767       % ##2 = {####1}{####2}{####3}{####4}
2768       % ####1 = name
2769       % ####2 = mpath
2770       % ####3 = type
2771       % ####4 = {origname}{newname}*
2772     }
2773     \bool_if:NT \l__stex_iterate_continue_bool \l__stex_iterate_todo_tl
2774   }
2775 }
```

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

13.6.2 Structural Features

2776 `<@@=stex_features>`

```
\stex_structural_feature_module:nn
\stex_structural_feature_module_end:
2777 \cs_new_protected:Nn \stex_structural_feature_module:nn {
2778   \stex_if_do_html:TF {
2779     \exp_args:Nne \begin{stex_annotation_env} {
2780       \shtml:feature-#2={
2781         \l_stex_current_module_str/#1}
2782       \str_if_empty:NF \l_stex_mroname_str {
2783         \shtml:macroname={\l_stex_mroname_str}
2784       }
2785     }
2786     \stex_annotation_invisible:n{}
2787   }\group_begin:
2788   \stex_module_setup:n {#1-module}
2789 }
2790
2791 \cs_new_protected:Nn \stex_structural_feature_module_end: {
2792   \tl_gset_eq:NN \g_stex_last_feature_str \l_stex_current_module_str
2793   \stex_close_module:
2794   \stex_if_do_html:TF{
2795     \end{stex_annotation_env}
2796   }\group_end:
2797 }
```

(End of definition for \stex_structural_feature_module:nn and \stex_structural_feature_module_end:. These functions are documented on page 118.)

\stex_structural_feature_morphism:nnn

\stex_structural_feature_morphism_end:

```
2798 \bool_new:N \l__stex_features_implicit_bool
2799 \cs_new_protected:Nn \stex_structural_feature_morphism:nnnnn {
2800   \str_clear:N \l_stex_current_domain_str
2801   \tl_if_empty:nT{#3} {
2802     \l_stex_get_mathstructure:n{#4}
2803     \str_set_eq:NN \l_stex_current_domain_str \l_stex_get_structure_module_str
2804   }
2805   \str_if_empty:NT \l_stex_current_domain_str {
2806     \stex_import_module_uri:nn { #3 }{ #4 }
2807     \group_begin:
2808     \stex_import_require_module:ooo
2809       \l_stex_import_archive_str
2810       \l_stex_import_path_str
2811       \l_stex_import_name_str
2812     \exp_args:Nnx \use:nn \group_end: {
2813       \str_set:Nn \exp_not:N\l_stex_current_domain_str {\l_stex_import_ns_str}
2814     }
2815   }
2816   \tl_if_empty:nTF{#1} {
2817     \bool_set_true:N \l__stex_features_implicit_bool
2818     \str_set:Nx \l_tmpa_str {[ \l_stex_current_domain_str ] }
2819   }
```

```

2820   \bool_set_false:N \l__stex_features_implicit_bool
2821   \str_set:Nn \l_tmpa_str {#1}
2822 }
2823
2824 \stex_if_do_html:TF {
2825   \begin{stex_annotation_env} {
2826     \shtml:feature-#2={\l_stex_current_module_str?\l_tmpa_str},
2827     \shtml:domain={\l_stex_current_domain_str}
2828     #5
2829   }
2830   \stex_annotation_invisible:n{}
2831 } \group_begin:
2832 \str_set:Nn \l__stex_features_feature_str {#2}
2833 \str_set_eq:NN \l_stex_feature_name_str \l_tmpa_str
2834 \__stex_features_setup:
2835 \__stex_features_reactivate:
2836 %^A\stex_activate_module:o \l_stex_current_domain_str
2837 \exp_args:Ne \stex_metagroup_new:n {\l_stex_current_module_str / \l_stex_feature_name_str}
2838 }
2839
2840 \cs_new_protected:Nn \__stex_features_do_for_list: {
2841   \seq_clear:N \l_stex_fors_seq
2842   \clist_map_inline:Nn \l_stex_key_for_clist {
2843     \exp_args:Ne \stex_get_in_morphism:n{\tl_to_str:n{##1}}
2844     \seq_put_right:Nx \l_stex_fors_seq
2845     {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
2846   }
2847 }
2848
2849 \cs_new_protected:Nn \__stex_features_add_definiens:nn {
2850   \__stex_features_set_definiens_macros: #1\__stex_features_break:
2851   \stex_assign_do:n{#2}
2852   #2
2853 }
2854 \cs_new_protected:Npn \__stex_features_set_definiens_macros: #1?#2?#3\__stex_features_break:
2855   \str_set:Nn \l_stex_get_symbol_mod_str {#1?#2}
2856   \str_set:Nn \l_stex_get_symbol_name_str {#3}
2857   \exp_args:Nne \use:nn{\__stex_features_set_definiens_macros_i:nnnnnnn}{%
2858     \prop_item:Nn \l_stex_morphism_symbols_prop {[#1?#2]/[#3]}}
2859   }
2860 }
2861 \cs_new_protected:Nn \__stex_features_set_definiens_macros_i:nnnnnnn {
2862   \tl_set:Nn \l_stex_get_symbol_def_tl{#4}
2863 }
2864
2865 \cs_new_protected:Nn \stex_structural_feature_morphism_end: {
2866   \str_gset_eq:NN \l_stex_feature_name_str \l_stex_feature_name_str
2867   \str_gset_eq:NN \l_stex_current_domain_str \l_stex_current_domain_str
2868   \seq_gset_eq:NN \l_stex_morphism_symbols_prop \l_stex_morphism_symbols_prop
2869   \seq_gset_eq:NN \l_stex_morphism_renames_prop \l_stex_morphism_renames_prop
2870   \seq_gset_eq:NN \l_stex_morphism_morphisms_seq \l_stex_morphism_morphisms_seq
2871   \__stex_features_do_elaboration:
2872   \stex_if_do_html:TF{
2873     \end{stex_annotation_env}

```

```

2874   }\group_end:
2875 }
2876
2877 \cs_new_protected:Nn \__stex_features_setup: {
2878   \prop_clear:N \l_stex_morphism_symbols_prop
2879   \prop_clear:N \l_stex_morphism_renames_prop
2880   \seq_clear:N \l_stex_morphism_morphisms_seq
2881   \__stex_features_do_decls:
2882   \exp_args:No \__stex_features_do_morphisms:n \l_stex_current_domain_str
2883 }
2884
2885 \cs_new_protected:Nn \__stex_features_rename_all: {
2886 }
2887
2888 \cs_new:Npn \__stex_features_clean:nnw [#1] / #2 \_stex_end: {
2889   [#1]/[#2]
2890 }
2891
2892 \cs_new_protected:Nn \__stex_features_do_decls: {
2893   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_domain_str {
2894     \stex_if_starts_with:nnTF{##3}[{
2895       \exp_args:NNe \prop_put:Nnn \l_stex_morphism_symbols_prop {
2896         \__stex_features_clean:nnw ##3 \_stex_end:
2897       }
2898     }{
2899       \prop_put:Nnn \l_stex_morphism_symbols_prop
2900         {[##1]/[##3]}
2901     }{
2902       {[##2}{##4}{##5}{##6}{##7}{##8}##9
2903     }
2904   }
2905 }
2906
2907 \cs_new_protected:Nn \stex_structural_feature_morphism_check_total: {
2908   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2909     \__stex_features_total_check: ##1 ##2
2910   }
2911 }
2912
2913 \cs_new_protected:Npn \__stex_features_total_check: [#1]/[#2] #3 #4 #5 #6 #7 #8 #9 {
2914   \tl_if_empty:nT{#6}{%
2915     \msg_error:nnxx{\stex}{error/needsdefiniens}{##1##2}{total-morphism}
2916   }
2917 }
2918
2919 \cs_new:Npn \__stex_features_split_qm:w #1 ? #2 ? #3 { #3 }
2920 \cs_new_protected:Nn \__stex_features_do_elaboration: {
2921   \stex_debug:nn{morphisms}{%
2922     Elaborating:^^J\prop_to_keyval:N \l_stex_morphism_symbols_prop
2923     ^^J
2924     Renamings:^^J
2925     \prop_to_keyval:N \l_stex_morphism_renames_prop
2926   }
2927 }
```

```

2928 \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2929   \__stex_features_elab_check: ##1 ##2
2930 }
2931 \exp_args:No\stex_iterate_notations:nn\l_stex_current_domain_str{
2932   \prop_get:NnNTF \l_stex_morphism_renames_prop {##1}\l__stex_features_tmp {
2933     \exp_args:Ne \stex_module_add_notation:nnnnn
2934     {\l_stex_current_module_str ? \exp_after:wN \use_i:nn \l__stex_features_tmp}
2935   }{
2936     \exp_args:Ne \stex_module_add_notation:nnnnn
2937     {\l_stex_current_module_str ? \l_stex_feature_name_str
2938       / \__stex_features_split_qm:w ##1}
2939   }{##2}{##3}{##4}{##5}
2940 }
2941 \stex_module_add_morphism:ooox
2942   \l_stex_feature_name_str
2943   \l_stex_current_domain_str
2944   \l_stex_features_feature_str
2945   {\prop_map_function:NN \l_stex_morphism_renames_prop \__stex_features_rename:nn}
2946 }
2947
2948 \cs_new:Nn \__stex_features_rename:nn{
2949   {#1}{\use_i:nn#2}
2950 }
2951
2952 \cs_new_protected:Npn \__stex_features_elab_check: [#1]/[#2] #3 {
2953   \prop_get:NnNTF \l_stex_morphism_renames_prop {#1##2} \l__stex_features_tmp {
2954     \stex_debug:nn{morphisms}{Generating~\l_stex_features_tmp}
2955     \exp_after:wN \stex_module_add_symbol:nnnnnnnnN \l_stex_features_tmp
2956   }{
2957     \bool_if:NTF \l_stex_features_implicit_bool {
2958       \stex_debug:nn{morphisms}{Generating~#3:~\l_stex_feature_name_str / #2}
2959       \exp_args:Nno \stex_module_add_symbol:nnnnnnnnN {#3}{\l_stex_feature_name_str / #2}
2960     }{
2961       \stex_debug:nn{morphisms}{Generating~\l_stex_feature_name_str / #2}
2962       \exp_args:Nno \stex_module_add_symbol:nnnnnnnnN {}{\l_stex_feature_name_str / #2}
2963     }
2964   }
2965 }
2966
2967 \cs_new_protected:Nn \__stex_features_do_morphisms:n {
2968   \prop_map_inline:cn {c_stex_module_#1_morphisms_prop} {
2969     \__stex_features_do_morph:nnnn ##2
2970   }
2971 }
2972
2973 \cs_new_protected:Nn \__stex_features_do_morph:nnnn {
2974   \tl_if_empty:nF{#3} {
2975     \seq_put_right:Nn \l_stex_morphism_morphisms_seq {{#1}{#2}{#3}}
2976   }
2977   \__stex_features_do_morphisms:n{#2}
2978 }
2979
2980 \cs_new_protected:Npn \__stex_features_reactivate: {
2981   \stex_deactivate_macro:Nn \symdecl {module~environments}

```

```

2982 \stex_deactivate_macro:Nn \textsymdecl {module-environments}
2983 \stex_deactivate_macro:Nn \symdef {module-environments}
2984 \stex_deactivate_macro:Nn \notation {module-environments}
2985 \stex_deactivate_macro:Nn \importmodule {module-environments}
2986 \stex_deactivate_macro:Nn \requiremodule {module-environments}
2987 \stex_deactivate_macro:Nn \smodule {outside-of-morphisms}
2988 \stex_reactivate_macro:N \assign
2989 \stex_reactivate_macro:N \assignMorphism
2990 \stex_reactivate_macro:N \renamedecl
2991 \cs_set_eq:NN \stex_do_for_list: \stex_features_do_for_list:
2992 \cs_set_eq:NN \stex_add_definiens:nn \stex_features_add_definiens:nn
2993 }

```

(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

```

2994 \cs_new_protected:Nn \stex_get_in_morphism:n {
2995   \str_clear:N \l_stex_get_symbol_name_str
2996   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2997     \exp_args:Nx \stex_features_get_check:nnnn{\tl_to_str:n{#1}}##1##2
2998   }
2999   \str_if_empty:NT \l_stex_get_symbol_name_str {
3000     \prop_map_inline:Nn \l_stex_morphism_renames_prop {
3001       \stex_features_renamed_check:nnnnn{#1}##1=##2
3002     }
3003     \str_if_empty:NT \l_stex_get_symbol_name_str {
3004       \msg_error:nnxx{\stex}{error/unknownsymbolin}{#1}{
3005         morphism-\l_stex_feature_name_str
3006       }
3007     }
3008   }
3009 }
3100
311 \cs_new_protected:Npn \stex_features_renamed_check:nnnnn #1##2##3##4##5##6 {
312   \str_if_eq:nnTF{#1}{#5} {
313     \exp_args:Nnx \use:nn{\stex_features_check_break:nnnnnnnn{#2##3}{#4}}{
314       \prop_item:Nn \l_stex_morphism_symbols_prop {[#2##3]/[#4]}
315     }
316   }{
317     \str_if_eq:nnT{#1}{#6} {
318       \exp_args:Nnx \use:nn{\stex_features_check_break:nnnnnnnn{#2##3}{#4}}{
319         \prop_item:Nn \l_stex_morphism_symbols_prop {[#2##3]/[#4]}
320       }
321     }
322   }
323 }
324
325 \cs_new_protected:Npn \stex_features_get_check:nnnn #1[#2]/[#3]##4 {
326   \str_if_eq:nnTF{#1}{#3} {
327     \stex_features_check_break:nnnnnnnn{#2}{#3}{#4}
328   }{
329     \str_if_eq:nnTF{#1}{#4} {
330       \stex_features_check_break:nnnnnnnn{#2}{#3}{#4}

```

```

3031     }{
3032         \use_none:nnnnnn
3033     }
3034 }
3035 }
3036
3037 \cs_new_protected:Nn \__stex_features_check_break:nnnnnnnn {
3038     \prop_map_break:n{
3039         \str_set:Nn \l_stex_get_symbol_mod_str{\#1}
3040         \str_set:Nn \l_stex_get_symbol_name_str{\#2}
3041         \str_set:Nn \l_stex_get_symbol_macro_str{\#3}
3042         \int_set:Nn \l_stex_get_symbol_arity_int {\#4}
3043         \tl_set:Nn \l_stex_get_symbol_args_tl {\#5}
3044         \tl_set:Nn \l_stex_get_symbol_def_tl {\#6}
3045         \tl_set:Nn \l_stex_get_symbol_type_tl {\#7}
3046         \tl_set:Nn \l_stex_get_symbol_return_tl {\#8}
3047         \tl_set:Nn \l_stex_get_symbol_invoke_cs {\#9}
3048     }
3049 }

```

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

13.7 Inheritance

13.7.1 `\importmodule/\usemodule`

```

3050 <@=stex_importmodule>
\usemodule
3051 \stex_new_styable_cmd:nnnn {usemodule} { 0{} m } {
3052     \stex_import_module_uri:nn { #1 }{ #2 }
3053     \stex_import_require_module:ooo
3054     \l_stex_import_archive_str
3055     \l_stex_import_path_str
3056     \l_stex_import_name_str
3057     \stex_if_do_html:T {
3058         \hbox{\stex_annotation_invisible:nn
3059             {shtml:usemodule=\l_stex_import_ns_str} {}}
3060     }
3061     \stex_if_smsmode:F{
3062         \group_begin:
3063         \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3064         \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3065         \tl_clear:N \thisstyle
3066         \stex_style_apply:
3067         \group_end:
3068     }
3069 }{}

```

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

```
\stex_import_module_uri:nn
3070 \cs_new_protected:Nn \stex_import_module_uri:nn {
3071     \stex_debug:nn{importmodule}{URI:~>#1<~>#2<}
```

```

3072 \exp_args:NNnx \seq_set_split:Nnn \__stex_importmodule_seq ? { \tl_to_str:n{ #2 } }
3073 \seq_pop_right:NN \__stex_importmodule_seq \l_stex_import_name_str
3074 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \__stex_importmodule_seq ? }
3075 \tl_if_empty:nTF { #1 } {
3076   \stex_debug:nn{importmodule}{No~archive}
3077   \prop_if_exist:NTF \l_stex_current_archive_prop {
3078     \stex_debug:nn{importmodule}{Picking~current~archive}
3079     \str_set:Nx \l_stex_import_archive_str {
3080       \prop_item:Nn \l_stex_current_archive_prop { id }
3081     }
3082   }{
3083     \str_clear:N \l_stex_import_archive_str
3084     \str_set:Nn \l_stex_import_uri_str {file:}
3085     \str_if_empty:NTF \l_stex_import_path_str {
3086       \stex_debug:nn{importmodule}{Empty~Path}
3087       \stex_file_split_off_ext:NN \l__stex_importmodule_path_seq \g_stex_current_file
3088       \stex_file_split_off_lang:NN \l__stex_importmodule_path_seq \l__stex_importmodule_pa
3089       \str_set:Nx \l_stex_import_path_str {
3090         \stex_file_use:N \l__stex_importmodule_path_seq
3091       }
3092     }{
3093       \stex_debug:nn{importmodule}{Resolving~path~\l_stex_import_path_str~relative~to~\ste
3094       \stex_file_resolve:Nx \l__stex_importmodule_seq { \stex_file_use:N \g_stex_current_f
3095       \str_set:Nx \l_stex_import_path_str {
3096         \stex_file_use:N \l__stex_importmodule_seq
3097       }
3098       \stex_debug:nn{importmodule}{...yields~\l_stex_import_path_str}
3099     }
3100   }
3101 }{
3102   \stex_debug:nn{importmodule}{Archive~#1}
3103   \str_set:Nx \l_stex_import_archive_str { #1 }
3104   \stex_require_archive:o \l_stex_import_archive_str
3105   \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ \l_stex_import_archi
3106 }
3107 }

```

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

```

\stex_import_require_module:nnn
\stex_import_require_module:ooo
3108 \cs_new_protected:Npn \stex_import_require_module:nnn #1 {
3109   \tl_if_empty:nTF { #1 } {
3110     \str_clear:N \l__stex_importmodule_archive_str
3111     \str_set:Nn \l_stex_import_uri_str {file:}
3112     \__stex_importmodule_get_module:nnn {}
3113   }{
3114     \stex_require_archive:n { #1 }
3115     \str_set:Nx \l__stex_importmodule_archive_str {#1}
3116     \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ #1 _manifest_prop}{}
3117     \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
3118     \exp_args:No \__stex_importmodule_get_module:nnn \l__stex_importmodule_str
3119   }
3120 }
3121 \cs_generate_variant:Nn \stex_import_require_module:nnn {ooo}

```

```

3122 \cs_new_protected:Npn \stex_import_require_module_safe:nnn #1 {
3123   \tl_if_empty:nTF { #1 } {
3124     \str_clear:N \l__stex_importmodule_archive_str
3125     \str_set:Nn \l_stex_import_uri_str {file:}
3126     \__stex_importmodule_get_module_safe:nnn {}
3127   }{
3128     \stex_require_archive:n { #1 }
3129     \str_set:Nx \l__stex_importmodule_archive_str {#1}
3130     \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ #1 _manifest_prop}{}
3131     \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
3132     \exp_args:No \__stex_importmodule_get_module_safe:nnn \l__stex_importmodule_str
3133   }
3134 }
3135 }
3136
3137 \cs_new_protected:Nn \__stex_importmodule_get_module_uri:nnn {
3138   \tl_if_empty:nF {#2} {
3139     \str_set:Nx \l_stex_import_uri_str {\l_stex_import_uri_str / #2}
3140   }
3141   \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
3142     Current~file:\stex_file_use:N \g_stex_current_file^^J
3143     Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3144   }
3145   \stex_if_module_exists:nTF {\l_stex_import_uri_str?#3} {
3146     \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3147   }{
3148     \stex_if_module_exists:nTF{\stex_uri_use:N \l_stex_current_ns_uri ? #3} {
3149       \str_set:Nx \l_stex_import_ns_str {\stex_uri_use:N \l_stex_current_ns_uri ? #3}
3150     }{
3151       \__stex_importmodule_get_from_file:nnn{#1}{#2}{#3}
3152       \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3153     }
3154   }
3155 }
3156 \cs_new_protected:Nn \__stex_importmodule_get_module_uri_safe:nnn {
3157   \tl_if_empty:nF {#2} {
3158     \str_set:Nx \l_stex_import_uri_str {\l_stex_import_uri_str / #2}
3159   }
3160   \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
3161     Current~file:\stex_file_use:N \g_stex_current_file^^J
3162     Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3163   }
3164   \stex_if_module_exists:nTF {\l_stex_import_uri_str?#3} {
3165     \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3166   }{
3167     \stex_if_module_exists:nTF{\stex_uri_use:N \l_stex_current_ns_uri ? #3} {
3168       \str_set:Nx \l_stex_import_ns_str {\stex_uri_use:N \l_stex_current_ns_uri ? #3}
3169     }{
3170       \__stex_importmodule_get_from_file_safe:nnn{#1}{#2}{#3}
3171       \str_set:Nx \l_stex_import_ns_str {\l_stex_import_uri_str?#3}
3172     }
3173   }
3174 }
3175

```

```

3176 \cs_new_protected:Nn \__stex_importmodule_get_module:nnn {
3177   \stex_debug:nn{importmodule}{Requiring~>[#1]#2?#3<}
3178   \__stex_importmodule_get_module_uri:nnn{#1}{#2}{#3}
3179   \stex_activate_module:o \l_stex_import_ns_str
3180 }
3181
3182 \cs_new_protected:Nn \__stex_importmodule_get_module_safe:nnn {
3183   \stex_debug:nn{importmodule}{Requiring~>[#1]#2?#3<}
3184   \__stex_importmodule_get_module_uri_safe:nnn{#1}{#2}{#3}
3185 }
3186
3187 \cs_new_protected:Nn \__stex_importmodule_get_from_file:nnn {
3188   \stex_file_resolve:Nx \l_stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3189   \str_set:Nx \l_stex_importmodule_str {\stex_file_use:N \l_stex_importmodule_seq}
3190   \stex_debug:nn{imports}{Looking-for~\l_stex_import_uri_str?#3...}
3191   \__stex_importmodule_check_file:nn{ /#3.tex }{
3192     \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}){
3193       \__stex_importmodule_check_file:nn{/#3.en.tex} {
3194         \__stex_importmodule_check_file:nn{.tex} {
3195           \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex} {
3196             \__stex_importmodule_check_file:nn{.en.tex} {
3197               \msg_error:nnx{\stex}{error/unknownmodule}{\l_stex_import_uri_str?#3}
3198             }
3199           }
3200         }
3201       }
3202     }
3203   }
3204 \stex_if_smsmode:TF{
3205   \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l_stex_importmodule_str}{\stex_file_use:N
3206     \stex_debug:nn{imports}{Skipping-current-file}
3207   }{
3208     \__stex_importmodule_load_file:n{#3}
3209   }
3210 }{
3211   \__stex_importmodule_load_file:n{#3}
3212 }
3213 }
3214 \cs_new_protected:Nn \__stex_importmodule_get_from_file_safe:nnn {
3215   \stex_file_resolve:Nx \l_stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3216   \str_set:Nx \l_stex_importmodule_str {\stex_file_use:N \l_stex_importmodule_seq}
3217   \stex_debug:nn{imports}{Looking-for~\l_stex_import_uri_str?#3...}
3218   \__stex_importmodule_check_file:nn{ /#3.tex }{
3219     \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
3220       \__stex_importmodule_check_file:nn{/#3.en.tex} {
3221         \__stex_importmodule_check_file:nn{.tex} {
3222           \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex} {
3223             \__stex_importmodule_check_file:nn{.en.tex} {
3224               }
3225             }
3226           }
3227         }
3228       }
3229     }

```

```

3230 \stex_if_smsmode:TF{
3231   \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
3232     \stex_debug:nn{imports}{Skipping-current-file}
3233   }{
3234     \IfFileExists{ \l__stex_importmodule_str }{
3235       \__stex_importmodule_load_file:n{#3}
3236     }{}
3237   }
3238 }{
3239   \IfFileExists{ \l__stex_importmodule_str }{
3240     \__stex_importmodule_load_file:n{#3}
3241   }{}
3242 }
3243 }
3244
3245 \cs_new_protected:Nn \__stex_importmodule_load_file:n {
3246   \stex_file_in_smsmode:on \l__stex_importmodule_str {
3247     \str_if_empty:NF \l__stex_importmodule_archive_str {
3248       \stex_set_current_archive:n \l__stex_importmodule_archive_str
3249     }
3250     \stex_debug:nn{modules}{Loading~\l__stex_importmodule_str}
3251   }
3252   \stex_if_module_exists:nF {\l_stex_import_uri_str?#1} {
3253     \msg_error:nnx{stex}{error/unknownmodule}{\l_stex_import_uri_str?#1}
3254   }
3255 }
3256
3257 \cs_new_protected:Npn \__stex_importmodule_check_file:nn #1 {
3258   \stex_debug:nn{imports}{Checking~\l__stex_importmodule_str #1}
3259   \IfFileExists{ \l__stex_importmodule_str #1 }{
3260     \stex_debug:nnf{imports}{Success}
3261     \str_set:Nx \l__stex_importmodule_str { \l__stex_importmodule_str #1 }
3262   }
3263 }

```

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

`\importmodule`

```

3264 \stex_new_stylable_cmd:nnnn{importmodule} { O{} m } {
3265   \__stex_importmodule_import_module:nn {#1}{#2}
3266   \stex_smsmode_do:
3267 }{}
3268 \stex_deactivate_macro:Nn \importmodule {module-environments}
3269
3270 \cs_new_protected:Nn \__stex_importmodule_import_module:nn {
3271   \stex_import_module_uri:nn { #1 }{ #2 }
3272   \stex_import_require_module:ooo
3273     \l_stex_import_archive_str
3274     \l_stex_import_path_str
3275     \l_stex_import_name_str
3276   \stex_execute_in_module:x{
3277     \stex_activate_module:n{\l_stex_import_ns_str}
3278   }
3279   \stex_module_add_morphism:nonn

```

```

3280     {}{\l_stex_import_ns_str}{import}{}}
3281 \stex_if_do_html:T {
3282   \stex_annotation_invisible:nn
3283   {shtml:import=\l_stex_import_ns_str} {}
3284 }
3285 \stex_if_smsmode:F{
3286   \group_begin:
3287   \tl_set:Nn \thisarchive {\#1}
3288   \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3289   \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3290   \tl_clear:N \thisstyle
3291   \stex_style_apply:
3292   \group_end:
3293 }
3294 }
3295
3296 \cs_new_protected:Nn \__stex_importmodule_import_module_presms:nn {
3297   \stex_import_module_uri:nn { #1 }{ #2 }
3298   \tl_gput_right:Nx \g_stex_sms_import_code {
3299     \stex_import_require_module_safe:nnn
3300     {\l_stex_import_archive_str}
3301     {\l_stex_import_path_str}
3302     {\l_stex_import_name_str}
3303   }
3304 }
3305
3306 \stex_sms_allow_escape:N \importmodule
3307 \stex_every_module:n {\stex_reactivate_macro:N \importmodule}
3308 \stex_sms_allow_import:Nn \importmodule {
3309   \stex_reactivate_macro:N \importmodule
3310   \let \__stex_importmodule_import_module:nn \__stex_importmodule_import_module_presms:nn
3311 }
3312
3313 \stex_new_stylable_cmd:nnnn[requiremodule] { 0{} m } {
3314   \stex_import_module_uri:nn { #1 }{ #2 }
3315   \stex_import_require_module:ooo
3316   \l_stex_import_archive_str
3317   \l_stex_import_path_str
3318   \l_stex_import_name_str
3319   \stex_do_up_to_module:x{
3320     \stex_activate_module:n{\l_stex_import_ns_str}
3321   }
3322   \stex_if_do_html:T {
3323     \stex_annotation_invisible:nn
3324     {shtml:import=\l_stex_import_ns_str} {}
3325   }
3326 \stex_if_smsmode:F{
3327   \group_begin:
3328   \tl_set:Nn \thisarchive {\#1}
3329   \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3330   \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3331   \tl_clear:N \thisstyle
3332   \stex_style_apply:
3333   \group_end:

```

```

3334   }
3335   \stex_smsmode_do:
3336 }{}
3337 \stex_deactivate_macro:Nn \requiremodule {module-environments}

```

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

13.7.2 Theory Morphisms

```

3338 <@=stex_morphisms>
3339 \stex_new_stylable_cmd:nnnn {assign} { m m }{
3340   \stex_get_in_morphism:n{#1}
3341   \stex_assign_do:n{#2}
3342   \stex_smsmode_do:
3343 }{}
3344 \stex_sms_allow_escape:N\assign
3345
3346 \cs_new_protected:Nn \stex_assign_do:n{
3347   \stex_debug:nn{assign}{Assigning~\l_stex_get_symbol_name_str~to~\tl_to_str:n{#1}}
3348   \tl_if_empty:NF \l_stex_get_symbol_def_tl {
3349     \%msg_error:nnxx{stex}{error/symbolalreadydefined}{\l_stex_get_symbol_name_str}{
3350       % morphism~\l_stex_feature_name_str
3351       %}
3352   }
3353   \stex_check_term:n{#1}
3354   \stex_debug:nn{HERE!}{
3355     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str^J
3356     \tl_to_str:n{#1}
3357   }
3358   \stex_if_do_html:T{
3359     \stex_annotation_invisible:nn{shtml:assign={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_
3360       \stex_annotation_force_break:n{
3361         \mode_if_math:T\hbox{\$ \stex_annotation:nn{shtml:definiens={}{}{#1} \$}
3362       }
3363     }
3364   }
3365   \exp_args:Ne \stex_metagroup_do_in:nx{
3366     \l_stex_current_module_str / \l_stex_feature_name_str
3367   }{
3368     \prop_put:Nnn \exp_not:N \l_stex_morphism_symbols_prop
3369     {[ \l_stex_get_symbol_mod_str ] / [ \l_stex_get_symbol_name_str ]}
3370   {
3371     {\l_stex_get_symbol_macro_str}
3372     {\int_use:N \l_stex_get_symbol_arity_int}
3373     {\l_stex_get_symbol_args_tl}
3374     {\exp_not:n{#1}}
3375     {\exp_args:No \exp_not:n \l_stex_get_symbol_type_tl}
3376     {\exp_args:No \exp_not:n \l_stex_get_symbol_return_tl}
3377     {\l_stex_get_symbol_invoke_cs}
3378   }
3379 }
3380 }
3381
3382

```

```

3383 \stex_new_stylable_cmd:nnnn {renamedecl} { m 0{} m }{
3384   \stex_get_in_morphism:n{#1}
3385   \_stex_renamedecl_do:nn{#2}{#3}
3386   \stex_smsmode_do:
3387 }{}
3388 \stex_sms_allow_escape:N\renamedecl
3389
3390 \cs_new_protected:Nn \_stex_renamedecl_do:nn {
3391   \stex_debug:nn{renamedecl}{Renaming~\l_stex_get_symbol_name_str~to~[#1]{#2}}
3392   \stex_if_do_html:T{
3393     \exp_args:Ne \stex_annotation_invisible:nn{
3394       shtml:rename={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str},
3395       shtml:macroname={#2}
3396       \str_if_empty:nF{#1}{ ,shtml:to={#1} }
3397     }{}
3398   }
3399   \exp_args:Ne \stex_metagroup_do_in:nx{
3400     \l_stex_current_module_str / \l_stex_feature_name_str
3401   }{
3402     \prop_put:Nnn \exp_not:N \l_stex_morphism_renames_prop
3403     {\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str}{##2}{%
3404       \tl_if_empty:nTF{#1}{\l_stex_feature_name_str/\l_stex_get_symbol_name_str}{#1}
3405     }%
3406   }
3407 }
3408
3409 \stex_new_stylable_cmd:nnnn {assignMorphism} { m m }{
3410   \str_clear:N \l__stex_morphisms_morphism_dom_str
3411   \stex_iterate_morphisms:nn\l_stex_current_domain_str{
3412     \stex_debug:nn{assignMorphism}{%
3413       Checking:~#1-vs:^^J##1^^J##2^^J##3^^J##4
3414     }
3415     \str_if_eq:nnTF{#1}{##1}{%
3416       \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3417     }{
3418       \stex_str_if_ends_with:nnT{##2}{#1}{%
3419         \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3420       }
3421     }
3422   }
3423   \str_if_empty:NT \l__stex_morphisms_morphism_dom_str {
3424     \msg_error:nnn{\stex}{error/nomorphism}{#1}
3425   }
3426   \bool_set_false:N \l_tmpa_bool
3427   \stex_iterate_morphisms:nn \l_stex_current_module_str {
3428     \stex_debug:nn{assignMorphism}{%
3429       Checking:~#2-vs:^^J##1^^J##2^^J##3^^J##4
3430     }
3431     \str_if_eq:nnTF{#2}{##1}{%
3432       \stex_debug:nn{assignMorphism}{match!}
3433       \stex_iterate_break:n{
3434         \stex_annotation_invisible:nn{
3435           shtml:assignMorphismFrom={\l__stex_morphisms_morphism_dom_str}
3436           ahtml:assignMorphismTo={\l_stex_current_module_str?##1}

```

```

3437     }{}
3438     \bool_set_true:N \l_tmpa_bool
3439   }
3440   }{
3441     \stex_if_ends_with:nnT{##2}{#2}{%
3442       \stex_debug:nn{assignMorphism}{match!}
3443       \stex_iterate_break:n{%
3444         \stex_annotation_invisible:nn{%
3445           shtml:assignMorphismFrom={\l_stex_morphisms_morphism_dom_str},
3446           shtml:assignMorphismTo={\l_stex_current_module_str?##1}
3447         }{}%
3448         \bool_set_true:N \l_tmpa_bool
3449       }%
3450     }%
3451   }%
3452 }
3453 \bool_if:NF \l_tmpa_bool {
3454   \msg_error:nnn{\stex}{error/nomorphism}{#2}
3455 }
3456 }{}%
3457 \cs_new_protected:Nn \__stex_morphisms_do_morph_assign:nnn {
3458   \stex_iterate_break:n{%
3459     \str_set:Nx \l_stex_morphisms_morphism_dom_str { \l_stex_current_domain_str ? #1 }
3460     \stex_debug:nn{assignMorphism}{match!}
3461     \stex_iterate_symbols:nn{#2}{%
3462       \stex_debug:nn{assignMorphism}{removing~##1?##3}
3463       % TODO: non-trivial assignments
3464       \prop_remove:NN \l_stex_morphism_symbols_prop {
3465         [##1]/[##3]
3466       }%
3467     }%
3468   }%
3469 }%
3470 \stex_deactivate_macro:Nn \assign {morphism~environments}
3472 \stex_deactivate_macro:Nn \renamedecl {morphism~environments}
3473 \stex_deactivate_macro:Nn \assignMorphism {morphism~environments}
3474 \stex_new_stylable_env:nnnnnnn {copymodule}{m 0{} m}{%
3475
3476   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3477
3478   \stex_if_smsmode:F {
3479     \tl_set:Nn \thiscopyname { #1 }
3480     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3481     \stex_style_apply:
3482   }%
3483   \stex_smsmode_do:
3484 }{%
3485     \stex_if_smsmode:F {
3486       \stex_style_apply:
3487     }%
3488     \stex_structural_feature_morphism_end:
3489 }{}{}{}%
3490 \stex_deactivate_macro:Nn \copymodule {module~environments}

```

```

3491 \stex_every_module:n {
3492   \stex_reactivate_macro:N \copymodule
3493 }
3494 \stex_sms_allow_env:n{copymodule}
3495
3496 \stex_new_stylable_env:nnnnnn {interpretmodule}{m 0{} m}{
3497   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3498   \stex_if_smsmode:F {
3499     \tl_set:Nn \thiscopyname { #1 }
3500     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3501     \stex_style_apply:
3502   }
3503   \stex_smsmode_do:
3504 }{
3505   \stex_structural_feature_morphism_check_total:
3506   \stex_if_smsmode:F {
3507     \stex_style_apply:
3508   }
3509   \stex_structural_feature_morphism_end:
3510 }{}{}{}
3511 \stex_deactivate_macro:Nn \interpretmodule {module~environments}
3512 \stex_every_module:n {
3513   \stex_reactivate_macro:N \interpretmodule
3514 }
3515 \stex_sms_allow_env:n{interpretmodule}
3516 \stex_new_stylable_env:nnnnnn {realization}{0{} m}{

3517   \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}
3518   \% \stex_execute_in_module:x{
3519   % \stex_activate_module:n{\l_stex_current_domain_str}
3520   %}
3521   \stex_if_smsmode:F {
3522     \tl_set:Nn \thiscopyname { #2 }
3523     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3524     \stex_style_apply:
3525   }
3526   \stex_smsmode_do:
3527 }{
3528   \stex_structural_feature_morphism_check_total:
3529   \stex_if_smsmode:F {
3530     \stex_style_apply:
3531   }
3532   \stex_structural_feature_morphism_end:
3533 }{}{}{}
3534 \stex_deactivate_macro:Nn \realization {module~environments}
3535 \stex_every_module:n {
3536   \stex_reactivate_macro:N \realization
3537 }
3538 \stex_sms_allow_env:n{realization}
3539
3540 \cs_new_protected:Nn \__stex_morphisms_parse_assign:n {
3541   \str_clear:N \l__stex_morphisms_name_str
3542   \str_clear:N \l__stex_morphisms_newname_str
3543   \tl_clear:N \l__stex_morphisms_ass_tl

```

```

3544 \exp_args:NNe \seq_set_split:Nnn \l__stex_morphisms_seq {\tl_to_str:n{0}} {#1}
3545 \int_compare:nNnTF {\seq_count:N \l__stex_morphisms_seq} = 1 {
3546   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_next_tl
3547 }
3548   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3549   \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3550   \tl_set:Nx \l__stex_morphisms_next_tl {\seq_use:Nn \l__stex_morphisms_seq 0}
3551 }
3552 \exp_args:NNNo \seq_set_split:Nnn \l__stex_morphisms_seq = \l__stex_morphisms_next_tl
3553 \str_if_empty:NTF \l__stex_morphisms_name_str {
3554   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3555   \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3556   \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3557 }
3558   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_newname_str
3559   \exp_args:NNo \str_set:Nn \l__stex_morphisms_newname_str \l__stex_morphisms_newname_str
3560   \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3561 }
3562 \__stex_morphisms_do_parsed_assign:
3563 }
3564
3565 \cs_new_protected:Nn \__stex_morphisms_do_parsed_assign: {
3566   \exp_args:No \stex_get_in_morphism:n \l__stex_morphisms_name_str
3567   \str_if_empty:NF \l__stex_morphisms_newname_str {
3568     \exp_after:wN \__stex_morphisms_do_parsed_newname: \l__stex_morphisms_newname_str \__ste
3569   }
3570   \tl_if_empty:NF \l__stex_morphisms_ass_tl {
3571     \exp_args:No \stex_assign_do:n \l__stex_morphisms_ass_tl
3572   }
3573 }
3574
3575 \cs_new_protected:Nn \__stex_morphisms_do_parsed_newname: {
3576   \peek_charcode:NTF [ {
3577     \__stex_morphisms_do_parsed_newname:w
3578   }{
3579     \__stex_morphisms_do_parsed_newname:w []
3580   }
3581 }
3582
3583 \cs_new_protected:Npn \__stex_morphisms_do_parsed_newname:w [#1] #2 \__stex_morphisms_end: {
3584   \stex_renamedecl_do:nn{#1}{#2}
3585 }
3586
3587 \stex_new_stylable_cmd:nnnn{copymod}{m 0{} m m}{%
3588   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3589
3590   \clist_map_function:nN{#4}\__stex_morphisms_parse_assign:n
3591
3592   \stex_if_smsmode:F {
3593     \tl_set:Nn \thiscopyname { #1 }
3594     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3595     \stex_style_apply:
3596   }
3597 \stex_structural_feature_morphism_end:

```

```

3598   \stex_smsmode_do:
3599 }{}
3600 \stex_deactivate_macro:Nn \copymod {module~environments}
3601 \stex_every_module:n {
3602   \stex_reactivate_macro:N \copymod
3603 }
3604 \stex_sms_allow_escape:N\copymod
3605
3606
3607 \stex_new_stylable_cmd:nnnn{interpretmod}{m 0{} m m}{
3608   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3609
3610 \clist_map_function:nN{#4}\_stex_morphisms_parse_assign:n
3611
3612 \stex_if_smsmode:F {
3613   \tl_set:Nn \thiscopyname { #1 }
3614   \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3615   \stex_style_apply:
3616 }
3617 \stex_structural_feature_morphism_check_total:
3618 \stex_structural_feature_morphism_end:
3619 \stex_smsmode_do:
3620 }{}
3621 \stex_deactivate_macro:Nn \interpretmod {module~environments}
3622 \stex_every_module:n {
3623   \stex_reactivate_macro:N \interpretmod
3624 }
3625 \stex_sms_allow_escape:N\interpretmod
3626
3627
3628 \stex_new_stylable_cmd:nnnn{realize}{0{} m m}{
3629   \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}
3630
3631 \clist_map_function:nN{#3}\_stex_morphisms_parse_assign:n
3632
3633 \stex_if_smsmode:F {
3634   \tl_set:Nn \thiscopyname { #1 }
3635   \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3636   \stex_style_apply:
3637 }
3638 \stex_structural_feature_morphism_check_total:
3639 \stex_structural_feature_morphism_end:
3640 \stex_smsmode_do:
3641 }{}
3642 \stex_deactivate_macro:Nn \realize {module~environments}
3643 \stex_every_module:n {
3644   \stex_reactivate_macro:N \realize
3645 }
3646 \stex_sms_allow_escape:N\realize

```

13.8 Symbols

13.8.1 Declarations

```
3647 <@@=stex_symdecl>
```

Some setup:

```
\stex_if_check_terms:p:  
\stex_if_check_terms:TF  
3648 \stex_if_html_backend:TF {  
3649   \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {  
3650     \prg_return_false:  
3651   }  
3652 }{  
3653   \stex_get_env:Nn\__stex_symdecl_env_str{STEX_CHECKTERMS}  
3654   \str_if_empty:NF\__stex_symdecl_env_str{  
3655     \exp_args:No \str_if_eq:nnF \__stex_symdecl_env_str{false}{  
3656       \bool_set_true:N \c_stex_check_terms_bool  
3657     }  
3658   }  
3659   \bool_if:NTF \c_stex_check_terms_bool {  
3660     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {  
3661       \prg_return_true:  
3662     }  
3663   }{  
3664     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {  
3665       \prg_return_false:  
3666     }  
3667   }  
3668 }
```

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

```
\stex_check_term:n
```

```
3669 \stex_if_check_terms:TF{  
3670   \cs_new_protected:Nn \stex_check_term:n {  
3671     \hbox_set:Nn \l_tmpa_box {  
3672       \group_begin:  
3673         $$#1$$  
3674       \group_end:  
3675     }  
3676   }  
3677 }{  
3678   \cs_new_protected:Nn \stex_check_term:n {}  
3679 }
```

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

symdecl arguments:

```
3680 \stex_keys_define:nnnn{symargs}{  
3681   \str_clear:N \l_stex_key_args_str  
3682   \str_clear:N \l_stex_key_role_str  
3683   \str_clear:N \l_stex_key_reorder_str  
3684   \str_clear:N \l_stex_key_assoc_str  
3685 }{  
3686   args      .str_set:N  = \l_stex_key_args_str ,  
3687   reorder   .str_set:N  = \l_stex_key_reorder_str ,  
3688   assoc     .choices:nn  = {bin,binl,binr,pre,conj,pwconj}  
3689   {\str_set:Nx \l_stex_key_assoc_str \l_keys_choice_tl},  
3690   role      .str_set:N  = \l_stex_key_role_str
```

```

3691 }{}
3692
3693 \stex_keys_define:nnnn{decl}{
3694   \str_clear:N \l_stex_key_name_str
3695   \str_clear:N \l_stex_key_args_str
3696   \tl_clear:N \l_stex_key_type_tl
3697   \tl_clear:N \l_stex_key_def_tl
3698   \tl_clear:N \l_stex_key_return_tl
3699   \clist_clear:N \l_stex_key_argtypes_clist
3700 }{
3701   name      .str_set:N = \l_stex_key_name_str ,
3702
3703   return    .tl_set:N     = \l_stex_key_return_tl ,
3704   argtypes .clist_set:N = \l_stex_key_argtypes_clist ,
3705
3706   type      .tl_set:N     = \l_stex_key_type_tl ,
3707   def       .tl_set:N     = \l_stex_key_def_tl ,
3708
3709   align     .code:n      = {},
3710   gfc      .code:n      = {}
3711 }{style,deprecate,symargs}
3712 % \_stex_do_deprecation:n{#2}

\symdecl
3713 \str_new:N \l_stex_macroname_str
3714 \stex_new_stylable_cmd:nnnn {symdecl} { s m 0{} } {
3715   \stex_keys_set:nn{decl}{#3}
3716   \IfBooleanTF #1 {
3717     \str_clear:N \l_stex_macroname_str
3718   }{
3719     \str_set:Nx \l_stex_macroname_str { #2 }
3720   }
3721 \stex_symdecl_top:n{#2}
3722
3723 \stex_if_smsmode:F{
3724   \group_begin:
3725   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3726   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3727   \tl_set_eq:NN \thistype \l_stex_key_type_tl
3728   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
3729   \tl_set_eq:NN \thisargs \l_stex_key_args_str
3730   \tl_clear:N \thisstyle
3731   \stex_style_apply:
3732   \group_end:
3733 }
3734 \stex_smsmode_do:
3735 }{}
3736 \stex_deactivate_macro:Nn \symdecl {module~environments}
3737 \stex_every_module:n {\stex_reactivate_macro:N \symdecl}
3738 \stex_sms_allow_escape:N \symdecl

```

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

`\stex_symdecl_top:n`

```

3739 \cs_new_protected:Nn \stex_symdecl_top:n {
3740   \str_if_empty:NT \l_stex_key_name_str {
3741     \str_set:Nx \l_stex_key_name_str { #1 }
3742   }
3743   \stex_symdecl_do:
3744   \_stex_symdecl_check_terms:
3745   \__stex_symdecl_add_decl:
3746   \stex_if_do_html:T {
3747     \_stex_symdecl_html:
3748   }
3749 }
3750
3751 \cs_new_protected:Nn \__stex_symdecl_add_decl: {
3752   \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN} {
3753     {\l_stex_mroname_str}
3754     {\l_stex_key_name_str}
3755     {\int_use:N \l_stex_get_symbol_arity_int}
3756     {\l_stex_get_symbol_args_tl}
3757     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }%{\exp_args:No \exp_not:n \l_stex_key_def_tl
3758     {}%{\exp_args:No \exp_not:n \l_stex_key_type_tl}
3759     {}%{\exp_args:No \exp_not:n \l_stex_key_return_tl}
3760     \stex_invoke_symbol:
3761   }
3762   \exp_args:Ne \stex_ref_new_symbol:n
3763   {\l_stex_current_module_str?\l_stex_key_name_str}
3764 }
3765
3766 \cs_new:Nn \stex_return_args:nn {
3767   {\svar{ARGUMENT_#1}\stex_eat_exclamation_point:}
3768 }
3769
3770 \cs_new_protected:Nn \stex_symdecl_html: {
3771   \exp_args:Ne \stex_annotation_invisible:nn {
3772     \shtml:symdecl = {\l_stex_current_module_str ? \l_stex_key_name_str},
3773     \shtml:args = {\l_stex_key_args_str}
3774     \str_if_empty:NF \l_stex_mroname_str {
3775       \shtml:macroname={\l_stex_mroname_str}
3776     }
3777     \str_if_empty:NF \l_stex_key_assoc_str {
3778       \shtml:assocotype={\l_stex_key_assoc_str}
3779     }
3780     \str_if_empty:NF \l_stex_key_reorder_str {
3781       \shtml:reorderargs={\l_stex_key_reorder_str}
3782     }
3783     \str_if_empty:NF \l_stex_key_role_str {
3784       \shtml:role={\l_stex_key_role_str}
3785     }
3786   }{\hbox\bgroup\stex_annotation_force_break:n{
3787     \bool_set_true:N \stex_in_invisible_html_bool
3788     \tl_if_empty:NF \l_stex_key_type_tl {
3789       $ \stex_annotation:nn{\shtml:type={}}{\l_stex_key_type_tl} $
3790     }
3791     \tl_if_empty:NF \l_stex_key_def_tl {
3792       $ \stex_annotation:nn{\shtml:definiens={}}{\l_stex_key_def_tl} $
3793     }
3794   }
3795 }
```

```

3793     }
3794     \tl_if_empty:NF \l_stex_key_return_tl{
3795         \exp_args:Nno \use:n{
3796             \cs_generate_from_arg_count:NNnn \l__stex_symdecl_cs
3797             \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
3798             \tl_set:Nx \l__stex_symdecl_args_tl {\l_stex_map_args:N \l_stex_return_args:nn}
3799             $ \stex_annotation:nn{shtml:returntype={}}{
3800                 \exp_after:wN \l__stex_symdecl_cs \l__stex_symdecl_args_tl!
3801             }$}
3802     }
3803     \clist_if_empty:NF \l_stex_key_argtypes_clist {
3804         \stex_annotation:nn{shtml:argtypes={}}{\l_stex_annotation_force_break:n{
3805             \clist_map_inline:Nn \l_stex_key_argtypes_clist {
3806                 $ \stex_annotation:nn{shtml:type={}}{##1}$
3807             }
3808         }}
3809     }
3810 }
```

(End of definition for `\stex_symdecl_top:n`. This function is documented on page 115.)

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

```

3812 \cs_new_protected:Nn \stex_symdecl_do: {
3813     \stex_do_deprecation:n \l_stex_key_name_str
3814     \__stex_symdecl_parse_arity:
3815     \__stex_symdecl_do_args:
3816 }
3817
3818 \int_new:N \l_stex_assoc_args_count
3819
3820 \cs_new_protected:Nn \__stex_symdecl_parse_arity: {
3821     \int_zero:N \l_stex_get_symbol_arity_int
3822     \int_zero:N \l_stex_assoc_args_count
3823     \str_map_inline:Nn \l_stex_key_args_str {
3824         \str_case:nnF ##1 {
3825             0 { \str_map_break: }
3826             1 { \str_map_break:n{
3827                 \int_set:Nn \l_stex_get_symbol_arity_int {1}
3828                 \str_set:Nn \l_stex_key_args_str {i}
3829             } }
3830             2 { \str_map_break:n{
3831                 \int_set:Nn \l_stex_get_symbol_arity_int {2}
3832                 \str_set:Nn \l_stex_key_args_str {ii}
3833             } }
3834             3 { \str_map_break:n{
3835                 \int_set:Nn \l_stex_get_symbol_arity_int {3}
3836                 \str_set:Nn \l_stex_key_args_str {iii}
3837             } }
3838             4 { \str_map_break:n{
3839                 \int_set:Nn \l_stex_get_symbol_arity_int {4}
3840                 \str_set:Nn \l_stex_key_args_str {iiii}
3841             } }
3842             5 { \str_map_break:n{

```

```

3843     \int_set:Nn \l_stex_get_symbol_arity_int {5}
3844     \str_set:Nn \l_stex_key_args_str {iiiii}
3845   } }
3846   6 { \str_map_break:n{
3847     \int_set:Nn \l_stex_get_symbol_arity_int {6}
3848     \str_set:Nn \l_stex_key_args_str {iiiiii}
3849   } }
3850   7 { \str_map_break:n{
3851     \int_set:Nn \l_stex_get_symbol_arity_int {7}
3852     \str_set:Nn \l_stex_key_args_str {iiiiiii}
3853   } }
3854   8 { \str_map_break:n{
3855     \int_set:Nn \l_stex_get_symbol_arity_int {8}
3856     \str_set:Nn \l_stex_key_args_str {iiiiiiii}
3857   } }
3858   9 { \str_map_break:n{
3859     \int_set:Nn \l_stex_get_symbol_arity_int {9}
3860     \str_set:Nn \l_stex_key_args_str {iiiiiiiii}
3861   } }
3862   i {\int_incr:N \l_stex_get_symbol_arity_int}
3863   b {\int_incr:N \l_stex_get_symbol_arity_int}
3864   a {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3865   B {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3866 }{
3867   \msg_error:nnnx{stex}{error/wrongargs} {
3868     \l_stex_current_module_str ? \l_stex_key_name_str
3869   }{##1}
3870 }
3871 }
3872 }
3873
3874 \cs_new_protected:Nn \__stex_symdecl_do_args: {
3875   \tl_clear:N \l_stex_get_symbol_args_tl
3876   \int_step_inline:nn \l_stex_get_symbol_arity_int {
3877     \tl_put_right:Nn \l_stex_get_symbol_args_tl {##1}
3878     \tl_put_right:Nx \l_stex_get_symbol_args_tl {
3879       \str_item:Nn \l_stex_key_args_str {##1}
3880     }
3881   }
3882 }

```

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

`_stex_symdecl_check_terms:`

```

3883 \cs_new_protected:Nn \_stex_symdecl_check_terms: {
3884   \stex_check_term:n{
3885     \stex_debug:nn{check_terms}{Checking~type...}
3886     \group_begin:\l_stex_key_type_tl\group_end:
3887     \stex_debug:nn{check_terms}{Checking~definiens...}
3888     \group_begin:\l_stex_key_def_tl\group_end:
3889     \stex_debug:nn{check_terms}{Checking~return...}
3890     \group_begin:\l_stex_key_return_tl!\group_end:
3891     \stex_debug:nn{check_terms}{Checking~argument~types...}
3892     \group_begin:\l_stex_key_argtypes_clist\group_end:

```

```

3893     }
3894 }

```

(End of definition for `_stex_symdecl_check_terms`. This function is documented on page 115.)

\textsymdecl

```

3895
3896 \stex_keys_define:nnnn{textsymdecl}{
3897   \str_clear:N \l_stex_key_name_str
3898   \tl_clear:N \l_stex_key_type_tl
3899   \tl_clear:N \l_stex_key_def_tl
3900 }{
3901   name      .str_set:N  = \l_stex_key_name_str ,
3902   type      .tl_set:N    = \l_stex_key_type_tl ,
3903   def       .tl_set:N    = \l_stex_key_def_tl
3904 }{style,deprecate}
3905
3906 \stex_new_stylable_cmd:nnnn {textsymdecl} {m 0{} m} {
3907   \stex_keys_set:nn{symdef}{}
3908   \stex_keys_set:nn{textsymdecl}{#2}
3909   \str_set:Nx \l_stex_macroname_str { #1 }
3910   \str_if_empty:NT \l_stex_key_name_str {
3911     \str_set:Nn \l_stex_key_name_str {#1}
3912   }%
3913   % \str_set:Nx \l_stex_key_name_str {\l_stex_key_name_str-sym}
3914   %
3915   \str_set:Nn \l_stex_key_role_str {textsymdecl}
3916
3917 \stex_symdecl_do:
3918 \_stex_symdecl_check_terms:
3919 \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnN}{
3920   {\l_stex_macroname_str}
3921   {\l_stex_key_name_str}
3922   {0}{}
3923   {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
3924   {}% type
3925   {\use:c{\#1name_nospace}}% return
3926   \stex_invoke_text_symbol:
3927 }
3928 \exp_args:Ne \stex_ref_new_symbol:n
3929   {\l_stex_current_module_str?\l_stex_key_name_str}
3930 \stex_if_do_html:T {
3931   \_stex_symdecl_html:
3932 }
3933
3934 \int_set:Nn \l_stex_get_symbol_arity_int 0
3935 \tl_clear:N \l_stex_key_op_tl
3936 \str_clear:N \l_stex_key_intent_str
3937 \str_clear:N \l_stex_key_prec_str
3938 \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
3939 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
3940 \stex_notation_parse:n{\hbox{#3}}
3941 \_stex_notation_add:
3942 \stex_if_do_html:T {

```

```

3943   \def\comp{\_comp}
3944   \stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
3945 }
3946 \stex_execute_in_module:x{
3947   \__stex_symdecl_set_textsymdecl_macro:nnn{#1}{\l_stex_current_module_str?\l_stex_key_name}
3948   \exp_not:n{#3}
3949 }
3950
3951 \stex_if_smsmode:F{
3952   \group_begin:
3953   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3954   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3955   \tl_clear:N \thisstyle
3956   \stex_style_apply:
3957   \group_end:
3958 }
3959 \stex_smsmode_do:
3960 }{}
3961 \stex_deactivate_macro:Nn \textsymdecl {module-environments}
3962 \stex_every_module:n {\stex_reactivate_macro:N \textsymdecl}
3963 \stex_sms_allow_escape:N \textsymdecl
3964
3965 \cs_new_protected:Nn \__stex_symdecl_set_textsymdecl_macro:nnn {
3966   \cs_set_protected:cpn{#1name_nospace}{#3}
3967   \cs_set_protected:cpn{#1name} {
3968     \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3969     \mode_if_math:T\hbox{\let\xspace\relax #3}
3970     \mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3971   }
3972 }
3973
3974 \cs_new_protected:Nn \stex_invoke_text_symbol: {
3975   \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3976   \stex_term_oms_or_omv:nnn{}{}{\maincomp{\let\xspace\relax\l_stex_current_return_tl}}
3977   \group_end:\mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3978 }

```

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

\stex_get_symbol:n

```

3979 \cs_new_protected:Nn \stex_get_symbol:n {
3980   \stex_get_symbol:n{ #1 }
3981   \str_if_empty:NT \l_stex_get_symbol_name_str {
3982     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3983   }
3984 }
3985
3986 \cs_new_protected:Nn \stex_get_symbol:n {
3987   \str_clear:N \l_stex_get_symbol_mod_str
3988   \str_clear:N \l_stex_get_symbol_name_str
3989   \cs_if_exist:cTF { #1 }{
3990     \cs_set_eq:Nc \l_stex_symdecl_cs { #1 }
3991     % command name
3992     \exp_args:Ne \tl_if_empty:nTF { \cs_argument_spec:N \l_stex_symdecl_cs }{

```

```

3993     % ...that takes no arguments
3994     \exp_args:Ne \cs_if_eq:NNTF {\tl_head:N \l_stex_symdecl_cs}
3995         \_stex_invoke_symbol:nnnnnnnN
3996         \_stex_symdecl_get_symbol_from_cs:
3997             {\_stex_symdecl_get_symbol_from_string:n { #1 }}
3998     }{
3999         \_stex_symdecl_get_symbol_from_string:n { #1 }
4000     }
4001     }{
4002         \_stex_symdecl_get_symbol_from_string:n { #1 }
4003     }
4004 }
4005
4006 \int_new:N \l_stex_get_symbol_arity_int
4007 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_cs: {
4008     \stex_debug:nn{symbols}{Getting~from~cs...}
4009     \stex_pseudogroup_with:nn{\_stex_invoke_symbol:nnnnnnnN}{
4010         \cs_set:Npn \_stex_invoke_symbol:nnnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {
4011             \str_set:Nn \l_stex_get_symbol_mod_str {##1}
4012             \str_set:Nn \l_stex_get_symbol_name_str {##2}
4013             \int_set:Nn \l_stex_get_symbol_arity_int {##3}
4014             \tl_set:Nn \l_stex_get_symbol_args_tl {##4}
4015             \tl_set:Nn \l_stex_get_symbol_def_tl {##5}
4016             \tl_set:Nn \l_stex_get_symbol_type_tl {##6}
4017             \tl_set:Nn \l_stex_get_symbol_return_tl {##7}
4018             \tl_set:Nn \l_stex_get_symbol_invoke_cs {##8}
4019         }
4020         \l_stex_symdecl_cs
4021     }
4022 }
4023
4024 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_string:n {
4025     \stex_debug:nn{symbols}{Getting~from~string~#1...}
4026     \seq_set_split:Nnn \l_stex_symdecl_seq ? {#1}
4027     \seq_pop_right:NN \l_stex_symdecl_seq \l_stex_symdecl_name
4028     \seq_if_empty:NTF \l_stex_symdecl_seq {
4029         \exp_args:No \_stex_symdecl_get_from_one_string:n {#1}
4030     }{
4031         \exp_args:NNe \exp_args:Nno \_stex_symdecl_get_symbol_from_modules:nn {
4032             \seq_use:Nn \l_stex_symdecl_seq ?
4033         } \l_stex_symdecl_name
4034     }
4035 }
4036
4037 \cs_new_protected:Nn \_stex_symdecl_sym_from_str_i:nnnn {
4038     \bool_lazy_any:nTF{
4039         {\str_if_eq_p:nn{#2}{#3}}
4040         {\str_if_eq_p:nn{#2}{#4}}
4041         {\stex_str_if_ends_with_p:nn{#4}{/#2}}
4042     }{
4043         \_stex_symdecl_sym_i_finish:nnnnnnnN{#1}{#4}
4044     }{
4045         \_stex_symdecl_sym_i_gobble:nnnnnn
4046     }

```

```

4047 }
4048 \cs_new_protected:Nn \__stex_symdecl_sym_i_gobble:nnnnnn {}  

4049  

4050 \cs_new_protected:Nn \__stex_symdecl_sym_i_finish:nnnnnnN {
4051   \prop_map_break:n\seq_map_break:n{
4052     \str_set:Nn \l_stex_get_symbol_mod_str {\#1}
4053     \str_set:Nn \l_stex_get_symbol_name_str {\#2}
4054     \int_set:Nn \l_stex_get_symbol_arity_int {\#3}
4055     \tl_set:Nn \l_stex_get_symbol_args_tl {\#4}
4056     \tl_set:Nn \l_stex_get_symbol_def_tl {\#5}
4057     \tl_set:Nn \l_stex_get_symbol_type_tl {\#6}
4058     \tl_set:Nn \l_stex_get_symbol_return_tl {\#7}
4059     \tl_set:Nn \l_stex_get_symbol_invoke_cs {\#8}
4060   }{}}  

4061 }  

4062  

4063 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_modules:nn {
4064   \stex_debug:nn{symbols}{Getting~#2~in~#1...}
4065   \seq_map_inline:Nn \l_stex_all_modules_seq {
4066     \stex_str_if_ends_with:nnT{\##1}{\#1} {
4067       \prop_map_inline:cn{c_stex_module_##1_symbols_prop} {
4068         \__stex_symdecl_sym_from_str_i:nnn{\##1}{\#2} ####2
4069       }{}}  

4070     }{}}  

4071 }{}}  

4072 }{}}  

4073  

4074 \cs_new_protected:Nn \__stex_symdecl_get_from_one_string:n {
4075   \stex_debug:nn{symbols}{Getting~#1~anywhere...}
4076   \stex_iterate_symbols:n{
4077     \%stex_debug:nn{symbols}{>#1==##2~|~#1==##3<...}
4078     \bool_lazy_any:nT{
4079       {\str_if_eq_p:nn{\#1}{\##2}}
4080       {\str_if_eq_p:nn{\#1}{\##3}}
4081       {\stex_str_if_ends_with_p:nn{\##3}{/\#1}}}{}}{}}  

4082     \stex_iterate_break:n{
4083       \str_set:Nn \l_stex_get_symbol_mod_str {\##1}
4084       \str_set:Nn \l_stex_get_symbol_name_str {\##3}
4085       \int_set:Nn \l_stex_get_symbol_arity_int {\##4}
4086       \tl_set:Nn \l_stex_get_symbol_args_tl {\##5}
4087       \tl_set:Nn \l_stex_get_symbol_def_tl {\##6}
4088       \tl_set:Nn \l_stex_get_symbol_type_tl {\##7}
4089       \tl_set:Nn \l_stex_get_symbol_return_tl {\##8}
4090       \tl_set:Nn \l_stex_get_symbol_invoke_cs {\##9}
4091     }{}}{}}  

4092   }{}}{}}  

4093 }{}}{}}  

4094 }{}}{}}  

4095 }{}}{}}
```

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

13.8.2 Notations

```

4096 <@=stex_notations>

\_stex_map_args:N
\stex_map_notation_args:N
4097 \cs_new:Nn \stex_map_args:N {
4098   \tl_if_empty:NF \l_stex_get_symbol_args_tl {
4099     \exp_after:wN \stex_notations_map_args_i:w \exp_after:wN
4100     #1 \l_stex_get_symbol_args_tl \stex_notations_args_end:
4101   }
4102 }
4103 \cs_new:Npn \stex_notations_map_args_i:w #1 #2 #3 #4 \stex_notations_args_end: {
4104   #1 #2 #3
4105   \tl_if_empty:nF{#4} {
4106     \stex_notations_map_args_i:w #1 #4 \stex_notations_args_end:
4107   }
4108 }
4109
4110 \cs_new:Nn \stex_map_notation_args:N {
4111   \tl_if_empty:NF \l_stex_notation_args_tl {
4112     \exp_after:wN \stex_notations_map_args_ii:w \exp_after:wN
4113     #1 \l_stex_get_symbol_args_tl \stex_notations_args_end:
4114   }
4115 }
4116 \cs_new:Npn \stex_notations_map_args_ii:w #1 #2 #3 #4 #5 #6 \stex_notations_args_end: {
4117   #1 #2 #3 #4 #5
4118   \tl_if_empty:nF{#6} {
4119     \stex_notations_map_args_ii:w #1 #6 \stex_notations_args_end:
4120   }
4121 }

(End of definition for \stex_map_args:N and \stex_map_notation_args:N. These functions are documented on page ??.)

notation arguments:
4122 \stex_keys_define:nnnn{notation}{

4123   \str_clear:N \l_stex_key_variant_str
4124   \str_clear:N \l_stex_key_prec_str
4125   \str_clear:N \l_stex_key_op_tl
4126   \str_clear:N \l_stex_key_intent_str
4127   \clist_clear:N \l_stex_key_intent_args_clist
4128 }{
4129   variant .str_set_x:N = \l_stex_key_variant_str ,
4130   prec .str_set_x:N = \l_stex_key_prec_str ,
4131   op .tl_set:N = \l_stex_key_op_tl ,
4132   intent .str_set:N = \l_stex_key_intent_str ,
4133   argnames .clist_set:N = \l_stex_key_intent_args_clist ,
4134   unknown .code:n = {
4135     \str_if_empty:NTF \l_keys_key_str {
4136       \str_set:Nx \l_stex_key_variant_str {\l_keys_key_tl}
4137     }{
4138       \str_set_eq:NN \l_stex_key_variant_str \l_keys_key_str
4139     }
4140   }
4141 }{style}

\notation

```

```

4142 \stex_new_stylable_cmd:nnnn {notation} { s m 0{} m} {
4143   \stex_keys_set:nn{notation}{#3}
4144   \stex_get_symbol:n{#2}
4145   \stex_notation_parse:n{#4}
4146   \stex_if_check_terms:T{ \_stex_notation_check: }
4147   \_stex_notation_add:
4148   \stex_if_do_html:T {
4149     \def\comp{\_comp}
4150     \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4151   }
4152   \IfBooleanTF#1{
4153     \_stex_notation_set_default:n{
4154       \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4155     }
4156   }{}
4157   \stex_if_smsmode:F{
4158     \group_begin:
4159     \_\_stex_notations_styledefs:
4160     \stex_style_apply:
4161     \group_end:
4162   }
4163   \stex_smsmode_do:
4164 }{}
4165
4166 \cs_new_protected:Nn \_\_stex_notations_styledefs: {
4167   \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4168   \str_set:Nn \thisdeclname \l_stex_get_symbol_name_str
4169   \tl_set:Nx \thisdecluri {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4170   \def\thisnotation{
4171     $
4172     \tl_set_eq:NN \l_stex_current_symbol_str \thisdecluri
4173     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{
4174       \_stex_notation_make_args:
4175     }$
4176   }
4177 }
4178
4179 \stex_deactivate_macro:Nn \notation {module~environments}
4180 \stex_every_module:n {\stex_reactivate_macro:N \notation}
4181 \stex_sms_allow_escape:N \notation

```

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

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

```

4182 \cs_new_protected:Nn \stex_notation_parse:n {
4183   \tl_if_empty:NF \l_stex_key_op_tl {
4184     \tl_set:Nx \l_stex_key_op_tl { \exp_not:N \maincomp {
4185       \exp_args:No \exp_not:n \l_stex_key_op_tl
4186     } }
4187   }
4188   \seq_clear:N \l_\_stex_notations_precs_seq
4189   \tl_clear:N \l_stex_notation_args_tl
4190   \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0 {

```

```

4191   \__stex_notations_const_precs:
4192   \tl_if_empty:NT \l_stex_key_op_tl {
4193     \tl_set:Nn \l_stex_key_op_tl { \maincomp{#1} }
4194   }
4195 }{
4196   \__stex_notations_fun_precs:
4197   \str_set:Nn \l__stex_notations_missing_str {#1}
4198   \tl_clear:N \l__stex_notations_missing_tl
4199   \stex_map_args:N \__stex_notations_add_missing_args:nn
4200   \tl_if_empty:NT \l_stex_key_op_tl {
4201     \hbox_set:Nn \l_tmpa_box {
4202       \str_set:Nn \l_stex_current_symbol_str {}
4203       \cs_set:Npn \l_tmpa_cs ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
4204       \cs_set:Npn \maincomp ##1 {
4205         \tl_gset:Nn \l_stex_key_op_tl { \maincomp{##1} }
4206         ##1
4207       }
4208       \cs_set:Npn \argsep ##1 ##2 {##1 ##2}
4209       \cs_set:Npn \argmap ##1 ##2 ##3 {##1 ##3}
4210       \cs_set:Npn \argarraymap ##1 ##2 ##3 ##4 {
4211         ##1 ##2
4212       }
4213       \stex_suppress_html:n{$\l_tmpa_cs abcdefghj$}
4214     }
4215   }
4216 }
4217 \exp_args:NNe
4218 \tl_set:Nn \l_stex_notation_macrocode_cs {
4219   \STEXInternalNotation
4220   { \l_stex_key_variant_str }
4221   { \l__stex_notations_oppref_tl }
4222   { \l_stex_key_intent_str }
4223   { \l_stex_notation_args_tl }
4224   {
4225     \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0
4226     { \exp_not:n { \maincomp{ #1 } } }
4227     { \exp_not:n { #1 } \l__stex_notations_missing_tl }
4228   }
4229 }
4230 \stex_debug:nn{notation}{Notation:~\meaning\l_stex_notation_macrocode_cs}
4231 }
4232
4233 \cs_new_protected:Nn \__stex_notations_const_precs: {
4234   \str_if_empty:NTF \l_stex_key_prec_str {
4235     \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4236   }{
4237     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4238       \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4239     }{
4240       \tl_set_eq:NN \l__stex_notations_oppref_tl \l_stex_key_prec_str
4241     }
4242   }
4243 }
4244

```

```

4245 \cs_new_protected:Nn \__stex_notations_fun_precs: {
4246   \str_if_empty:NTF \l_stex_key_prec_str {
4247     \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4248   }{
4249     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4250       \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4251     }{
4252       \tl_set_eq:NN \l__stex_notations_oppref_tl \l_stex_key_prec_str
4253     }
4254   }
4255   \str_if_empty:NTF \l_stex_key_prec_str {
4256     \tl_set:Nn \l__stex_notations_oppref_tl { 0 }
4257     \int_step_inline:nn \l_stex_get_symbol_arity_int {
4258       \seq_put_right:Nn \l__stex_notations_precs_seq {0}
4259     }
4260   }{
4261     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4262       \stex_debug:nn{notation}{No~brackets}
4263       \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4264       \int_step_inline:nn \l_stex_get_symbol_arity_int {
4265         \exp_args:NNo \seq_put_right:Nn \l__stex_notations_precs_seq \infpref
4266       }
4267     } \__stex_notations_parse_precs:
4268   }
4269   \__stex_notations_do_argnames:
4270 }
4271
4272 \cs_new_protected:Nn \__stex_notations_parse_precs: {
4273   \stex_debug:nn{notation}{parsing-precedence~\l_stex_key_prec_str}
4274   \seq_set_split:NnV \l__stex_notations_seq ; \l_stex_key_prec_str
4275   \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4276     \tl_set_eq:NN \l__stex_notations_oppref_tl \l__stex_notations_str
4277     \seq_pop_left:NNT \l__stex_notations_seq \l__stex_notations_str {
4278       \exp_args:NNo \seq_set_split:NnV \l__stex_notations_seq
4279       {\tl_to_str:n{x}} \l__stex_notations_str
4280     }
4281   }{
4282     \tl_set:No \l__stex_notations_oppref_tl { 0 }
4283   }
4284   \int_step_inline:nn \l_stex_get_symbol_arity_int {
4285     \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4286       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_str
4287     }{
4288       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_oppref_tl
4289     }
4290   }
4291 }
4292
4293 \cs_new_protected:Nn \__stex_notations_do_argnames: {
4294   \tl_clear:N \l_stex_notation_args_tl
4295   \stex_map_args:N \__stex_notations_do_argname:nn
4296 }
4297
4298 \cs_new_protected:Nn \__stex_notations_do_argname:nn {

```

```

4299 \clist_if_empty:NTF \l_stex_key_intent_args_clist {
4300   \tl_put_right:Nx \l_stex_notation_args_tl {
4301     #1#2{\seq_item:Nn \l_stex_notations_precs_seq #1} {
4302       \str_if_empty:NF \l_stex_key_intent_str {#1}
4303     }
4304   }
4305 }{
4306   \tl_put_right:Nx \l_stex_notation_args_tl {
4307     #1#2{\seq_item:Nn \l_stex_notations_precs_seq #1}
4308     {\c_dollar_str\clist_item:Nn \l_stex_key_intent_args_clist 1}
4309   }
4310   \clist_pop:NN \l_stex_key_intent_args_clist \l_tmpa_tl
4311 }
4312 }
4313
4314 \cs_new:Nn \__stex_notations_add_missing_args:nn {
4315   \exp_args:NNe \str_if_in:NnF \l__stex_notations_missing_str {\c_hash_str\c_hash_str#1} {
4316     \tl_put_right:Nn \l__stex_notations_missing_tlf{\STEXinvis{## #1}}
4317   }
4318 }

```

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

```

\__stex_notation_check:
  \__stex_notation_add:
    \__stex_notation_do_html:n
\__stex_notation_make_args:
4319 \cs_new_protected:Nn \__stex_notation_check: {
4320   \stex_check_term:n{
4321     \str_set:Nn \l_stex_current_symbol_str {test}
4322     \cs_set:Npn \comp ##1 {##1}
4323     \stex_debug:nn{check_terms}{Checking~notation...}
4324     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}{
4325       \__stex_notation_make_args:
4326     }
4327   }
4328 }
4329
4330 \cs_new_protected:Nn \__stex_notation_add: {
4331   \stex_module_add_notation:eoeeo{
4332     \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4333     }\l_stex_key_variant_str
4334     {\int_use:N \l_stex_get_symbol_arity_int}
4335     \l_stex_notation_macrocode_cs
4336     \l_stex_key_op_tl
4337 }
4338
4339 \cs_new_protected:Nn \__stex_notation_do_html:n {
4340   \hbox{\stex_annotate_invisible:nn {
4341     \shtml:notation={#1},
4342     \shtml:notationfragment={\l_stex_key_variant_str},
4343     \shtml:precedence={\l__stex_notations_opprec_tl},
4344     \shtml:argprecs={\seq_use:Nn \l__stex_notations_precs_seq ,}
4345   }{
4346     \cs_set_protected:Npn \argsep ##1 ##2 {
4347       \stex_annotate:nn{\shtml:argsep={}}{
4348         ##1 ##2

```

```

4349     }
4350   }
4351 \cs_set_protected:Npn \argmap ##1 ##2 ##3 {
4352   \cs_set:Npn \__stex_notations_map_cs: #####1 { ##2 }
4353   \stex_annotate:nn{shtml:argmap={}}{
4354     \__stex_notations_map_cs:{##1} ##3
4355   }
4356 }
4357 \cs_set_protected:Npn \maincomp {
4358   \do_comp:nNn {maincomp}\compemph@uri
4359 }
4360 $ 
4361 \str_set:Nx \l_stex_current_symbol_str {#1}
4362 \stex_annotate:nn{shtml:notationcomp={}}{
4363   \exp_args:Nne \use:nn {
4364     \l_stex_notation_macrocode_cs {}
4365   }
4366   \__stex_map_args:N \__stex_notations_make_arg_html:nn
4367 }
4368 }
4369 $ 
4370 \tl_if_empty:NF \l_stex_key_op_tl {
4371   $
4372   \str_set:Nx \l_stex_current_symbol_str {#1}
4373   \stex_annotate:nn{shtml:notationopcomp={}}{
4374     \__stex_term_oms:nnn{\l_stex_key_variant_str}{}{\l_stex_key_op_tl}
4375   }
4376   $
4377 }
4378 }
4379 }
4380 \cs_new:Nn \__stex_notations_make_arg_html:nn {
4381 % \str_case:nnF #2 {
4382 %   a {{%
4383 %     \stex_annotate:nn{shtml:argnum=#1a}{x},
4384 %     \stex_annotate:nn{shtml:argnum=#1b}{x}
4385 %   }%
4386 %   B {{%
4387 %     \stex_annotate:nn{shtml:argnum=#1a}{x},
4388 %     \stex_annotate:nn{shtml:argnum=#1b}{x}
4389 %   }%
4390 % }%
4391 % }{%
4392 {
4393   \stex_annotate:nn{shtml:argnum=#1}{x}
4394 }
4395 % }
4396 }
4397 \cs_new:Nn \stex_notation_make_args: {
4398   \__stex_map_notation_args:N \__stex_notations_make_arg:nnnn
4399 }
4400 }
4401
4402

```

```

4403 \cs_new:Nn \__stex_notations_make_arg:nnnn {
4404   \str_case:nnF #2 {
4405     a {{{
4406       a\c_math_subscript_token{#1,1},
4407       a\c_math_subscript_token{#1,2}
4408     } }
4409     B {{{
4410       B\c_math_subscript_token{#1,1},
4411       B\c_math_subscript_token{#1,2}
4412     } }
4413   }{
4414     \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}
4415     {{#2}\c_math_subscript_token{#1}}
4416   }
4417 }

```

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

`\setnotation`

```

\_stex_notation_set_default:n
4418 \cs_new_protected:Npn \setnotation #1 #2 {
4419   \stex_get_symbol:n{#1}
4420   \cs_if_exist:cTF{l_stex_notation_
4421     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4422     _#2_cs
4423   }{
4424     \tl_set_eq:Nc \l_stex_notation_macrocode_cs {l_stex_notation_
4425       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4426       _#2_cs
4427   }
4428   \cs_if_exist:cTF{l_stex_notation_
4429     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4430     _op_#2_cs
4431   }{
4432     \tl_set_eq:Nc \l_stex_key_op_tl {l_stex_notation_
4433       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4434       _op_#2_cs
4435   }
4436   }{
4437     \tl_clear:N \l_stex_key_op_tl
4438   }
4439   \_stex_notation_set_default:n{
4440     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4441   }
4442   }{
4443     \msg_error:nnxx{stex}{unknownnotation}{#2}{
4444       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4445     }
4446   }
4447 }
4448 \cs_new_protected:Nn \_stex_notation_set_default:n{
4449   \stex_module_add_notation:eoeeo{#1}{}
4450   {\int_use:N \l_stex_get_symbol_arity_int}

```

```

4452     \l_stex_notation_macrocode_cs
4453     \l_stex_key_op_tl
4454 }

```

(End of definition for `\setnotation` and `_stex_notation_set_default:n`. These functions are documented on page 81.)

\varnotation

```

4455 \stex_new_stylable_cmd:nnnn {varnotation} { s m 0{} m} {
4456   \stex_keys_set:nn{notation}{#3}
4457   \stex_get_var:n{#2}
4458   \str_set_eq:NN \l_stex_key_name_str \l_stex_get_symbol_name_str
4459   \stex_notation_parse:n{#4}
4460   \stex_if_check_terms:T{ \_stex_notation_check: }
4461   \_stex_vardecl_notation_macro:
4462   \IfBooleanTF#1{
4463     \_stex_notation_set_default:n{\l_stex_get_symbol_name_str}
4464   }{}
4465   \group_begin:
4466   \tl_set_eq:NN \thisvarname \l_stex_get_symbol_name_str
4467   \tl_clear:N \thisstyle
4468   \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4469   \def\thisnotation{
4470     \$\let\l_stex_current_symbol_str\thisvarname
4471       \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}
4472       \_stex_notation_make_args:
4473     }$
4474   }
4475   \stex_style_apply:
4476   \group_end:
4477 }{}

```

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

\symdef

```

4478 \stex_keys_define:nnnn{symdef}{}{}{decl,notation}
4479
4480 \cs_new_protected:Nn \_stex_symdef_styledefs: {
4481   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
4482   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
4483   \tl_set_eq:NN \thistype \l_stex_key_type_tl
4484   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
4485   \tl_set_eq:NN \thisargs \l_stex_key_args_str
4486   \tl_clear:N \thisstyle
4487   \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4488   \def\thisnotation{
4489     \$\let\l_stex_current_symbol_str\thisdecluri
4490       \def\comp{\_comp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}
4491       \_stex_notation_make_args:
4492     }$
4493   }
4494 }
4495
4496 \stex_new_stylable_cmd:nnnn {symdef} { m 0{} m} {
4497   \stex_keys_set:nn{symdef}{#2}

```

```

4498 \str_set:Nx \l_stex_macroname_str { #1 }
4499 \stex_symdecl_top:n{#1}
4500 \stex_debug:nn{symdef}{Doing~\l_stex_current_module_str ? \l_stex_key_name_str}
4501 \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
4502 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
4503 \stex_notation_parse:n{#3}
4504 \stex_debug:nn{Here!}{\meaning\l_stex_notation_args_tl}
4505 \l_stex_notation_check:
4506 \l_stex_notation_add:
4507 \stex_if_do_html:T{
4508   \l_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4509 }
4510 \stex_if_smsmode:F{
4511   \group_begin:
4512   \l_stex_symdef_styledefs:
4513   \stex_style_apply:
4514   \group_end:
4515 }
4516 \l_stex_smsmode_do:
4517 }{}
4518
4519 \stex_deactivate_macro:Nn \symdef {module~environments}
4520 \stex_every_module:n {\l_stex_reactivate_macro:N \symdef}
4521 \stex_sms_allow_escape:N \symdef

```

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

```

\stex_do_default_notation_op:
4522 \cs_new_protected:Nn \stex_do_default_notation: {
4523   \stex_do_default_notation_op:
4524   \tl_if_empty:NTF \l_stex_current_args_tl {
4525     \tl_clear:N \l_stex_notations_args_tl
4526   }{
4527     \__stex_notations_make_name:
4528     \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
4529     \tl_set:Nx \l_stex_notations_args_tl {
4530       \stex_map_args:N \__stex_notations_augment_arg:nn
4531     }
4532     \tl_put_right:Nn \l_stex_default_notation {\comp{}}
4533     \seq_clear:N \l_tmpa_seq
4534     \int_step_inline:nn \l_stex_current_arity_str {
4535       \seq_put_right:Nn \l_tmpa_seq {#### ##1}
4536     }
4537     \tl_put_right:Nx \l_stex_default_notation {
4538       \seq_use:Nn \l_tmpa_seq {\mathpunct{\comp{,}}}}
4539     }
4540     \tl_put_right:Nn \l_stex_default_notation {\comp{}}
4541   }
4542   \tl_set:Nx \l_stex_default_notation {\STEXInternalNotation{}{0}{}{\l_stex_notations_args_}
4543   \exp_args:No \exp_not:n \l_stex_default_notation
4544 }
4545 }
4546
4547 \cs_new:Nn \__stex_notations_augment_arg:nn {

```

```

4548     #1#2{0}{}
4549 }
4550
4551 \cs_new_protected:Nn \__stex_notations_make_name: {
4552     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq ? \l_stex_current_symbol_str
4553     \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
4554     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq / \l__stex_notations_name_str
4555     \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
4556 }
4557
4558 \cs_new_protected:Nn \stex_do_default_notation_op: {
4559     \__stex_notations_make_name:
4560     \tl_set:Nx \l_stex_default_notation {\exp_not:N \maincomp{ \exp_not:N \mathrm {\l__stex_no
4561 }

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

\STEXInternalNotation

```

4562 % 1: variant 2: operator precedence 3: intent 4: arguments 5: code 6: next
4563
4564 \cs_new_protected:Npn \STEXInternalNotation #1 #2 #3 #4 #5 #6 {
4565     \__stex_notations_process_notation:nnnnnn{#1}{#2}{#3}{#4}{#5}{#
4566         \l__stex_notations_code_tl
4567         #6
4568     }
4569 }
4570
4571 \cs_new_protected:Npn \__stex_notations_process_notation:nnnnnn #1 #2 #3 #4 {
4572     \tl_if_empty:nTF{#4}{#
4573         \__stex_notations_simple:nnnnn{#1}{#2}{#3}
4574     }{
4575         \__stex_notations_complex:nnnnnn{#1}{#2}{#3}{#4}
4576     }
4577 }
4578
4579 \cs_new_protected:Nn \__stex_notations_simple:nnnnn {
4580     \stex_debug:nn{Notation~code}{\tl_to_str:n{#4}}
4581     \tl_set:Nn \l__stex_notations_code_tl {
4582         \cs_set:Npn \l__stex_notations_code_tl {
4583             \__stex_maybe_brackets:nn{#2}{#
4584                 \__stex_term_oms_or_omv:nnn{#1}{#3}{#4}
4585             }
4586         }
4587         \l__stex_notations_code_tl
4588     }
4589     \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4590     #5
4591 }
4592
4593 \cs_new_protected:Nn \__stex_notations_complex:nnnnnn {
4594     \stex_debug:nn{Notation~code}{\tl_to_str:n{#5}}
4595     \int_zero:N \l_tmpa_int
4596     \tl_set:Nn \l__stex_notations_pre_tl {\cs_set_eq:NN \__stex_term_oma_or_omb:nnn \__stex_term
4597     \tl_set:Nn \l__stex_notations_code_tl {
```

```

4598 \cs_generate_from_arg_count:NNnn \l__stex_notations_cs \cs_set:Npn \l_tmpa_int
4599 {
4600   \stex_maybe_brackets:nn{#2}{
4601     \stex_term_oma_or_omb:nnn{#1}{#3}{#5}
4602     \bool_set_false:N \l_stex_brackets_dones_bool
4603   }
4604 }
4605 }
4606 }
4607 \l__stex_notations_cs
4608 }
4609 \tl_set:Nn \l__stex_notations_after_tl{
4610   \exp_args:NNo
4611   \tl_put_left:Nn \l__stex_notations_code_tl \l__stex_notations_pre_tl
4612   \tl_put_left:Nx \l__stex_notations_code_tl {
4613     \int_set:Nn \l_tmpa_int {\int_use:N \l_tmpa_int}
4614   }
4615   \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4616   #6
4617 }
4618 \__stex_notations_parse_notation_args:nnnw #4 \__stex_notations_args_end:
4619 }
4620
4621 \cs_new_protected:Npn \__stex_notations_parse_notation_args:nnnw #1 #2 #3 #4 #5 \__stex_no
4622   \tl_if_empty:nTF{#5}{
4623     \__stex_notations_add_last:nnnnn{#1}{#2}{#3}{#4}{#5}
4624   }{
4625     \__stex_notations_add_next:nnnnnn{#1}{#2}{#3}{#4}{#5}{#6}
4626   }
4627 }
4628
4629 \cs_new_protected:Nn \__stex_notations_add_next:nnnnnn {
4630   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#6}
4631   \__stex_notations_parse_notation_args:nnnw #5 \__stex_notations_args_end:
4632 }
4633
4634 \cs_new_protected:Nn \__stex_notations_add_last:nnnnn {
4635   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#5}
4636   \l__stex_notations_after_tl
4637 }
4638
4639 \cs_new_protected:Nn \__stex_notations_add:nnnnn {
4640   \int_incr:N \l_tmpa_int
4641   \str_case:nn{#2}{{
4642     i {
4643       \tl_put_right:Nn \l__stex_notations_code_tl {
4644         {\stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4645     }
4646   }
4647   b {
4648     \tl_set:Nn \l__stex_notations_pre_tl {
4649       \cs_set_eq:NN \stex_term_oma_or_omb:nnn \stex_term_omb:nnn
4650     }
4651     \tl_put_right:Nn \l__stex_notations_code_tl {

```

```

4652     {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4653   }
4654 }
4655 a {
4656   \tl_put_right:Nn \l__stex_notations_code_tl {
4657     {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4658   }
4659 }
4660 B {
4661   \tl_set:Nn \l__stex_notations_pre_tl {
4662     \cs_set_eq:NN \stex_term_oma_or_omb:nnn \stex_term_omb:nnn
4663   }
4664   \tl_put_right:Nn \l__stex_notations_code_tl {
4665     {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4666   }
4667 }
4668 }
4669 }

```

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

a/B-mode argument handling

`\argsep`

```

4670 \cs_new_protected:Nn \__stex_notations_check_aB_arg:Nn {
4671   \exp_args:Ne \cs_if_eq:NNF {\tl_head:n{#2}}
4672     \_stex_term_arg_aB:nnnnn {
4673       \msg_error:nnx{\stex}{error/assocarg}{\tl_to_str:n{#1}}
4674     }
4675   }
4676 
4677 \cs_new_protected:Npn \argsep #1 #2 {
4678   \__stex_notations_check_aB_arg:Nn\argsep{#1}
4679   \stex_pseudogroup_with:nn{\__stex_term_do_aB_clist:}{
4680     \tl_set:Nn \__stex_term_do_aB_clist: {
4681       \seq_use:Nn \l_stex_aB_args_seq {#2}
4682     }
4683     #1
4684   }
4685 }

```

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

`\argmap`

```

4686 \cs_new_protected:Npn \argmap #1 #2 #3 {
4687   \__stex_notations_check_aB_arg:Nn\argmap{#1}
4688   \stex_pseudogroup_with:nn{
4689     \__stex_term_do_aB_clist:
4690     \__stex_notations_map_cs:
4691   }{
4692     \cs_set:Npn \__stex_notations_map_cs: ##1 { #2 }
4693     \tl_set:Nn \__stex_term_do_aB_clist: {
4694       \seq_clear:N \l_tmpa_seq
4695       \seq_map_inline:Nn \l_stex_aB_args_seq {

```

```

4696   \tl_if_eq:nnTF{##1}{\ellipses}{
4697     \seq_put_right:Nn \l_tmpa_seq \ellipses
4698   }{
4699     \seq_put_right:Nx \l_tmpa_seq {
4700       \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
4701     }
4702   }
4703 }
4704 \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4705 \seq_use:Nn \l_stex_aB_args_seq {#3}
4706 }
4707 #1
4708 }
4709 }

```

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

`\argarraymap`

```

4710 \int_new:N \l__stex_notations_clist_count_int
4711 \cs_new_protected:Npn \argarraymap #1 #2 #3 #4 {
4712   \__stex_notations_check_aB_arg:Nn\argarraymap{#1}
4713   \stex_pseudogroup_with:nn{
4714     \stex_term_do_aB_clist:
4715     \__stex_notations_map_cs:
4716   }{
4717     \cs_set:Npn \__stex_notations_map_cs: ##1 { #3 }
4718     \int_set:Nn \l__stex_notations_clist_count_int {\exp_args:No\clist_count:n{\tl_to_str:n{%
4719       \tl_set:Nn \stex_term_do_aB_clist: {
4720         \tl_clear:N \l_tmpa_tl
4721         \int_zero:N \l_tmpa_int
4722         \seq_map_inline:Nn \l_stex_aB_args_seq {
4723           \int_incr:N \l_tmpa_int
4724           \int_compare:nNnT \l_tmpa_int > \l__stex_notations_clist_count_int {
4725             \int_set:Nn \l_tmpa_int 1
4726           }
4727           \tl_put_right:Nx \l_tmpa_tl {
4728             \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
4729             \clist_item:nn{#4}\l_tmpa_int
4730           }
4731         }
4732         \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4733         \begin{array}{#2}
4734           \l_tmpa_tl
4735         \end{array}
4736       }
4737     #1
4738   }
4739 }

```

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

13.8.3 Variables

```

4740 <@=stex_vars>

```

```

\vardef
4741 \tl_new:N \l_stex_variables_prop
4742 \bool_new:N \l__stex_vars_bind_bool
4743 \cs_new_protected:Nn \_stex_variable:nnnnnnnN {}
4744 \stex_keys_define:nnnn{vardef}{
4745   \bool_set_false:N \l__stex_vars_bind_bool
4746 }{
4747   bind .bool_set:N = \l__stex_vars_bind_bool
4748 }{symdef}
4749
4750 \stex_new_stylable_cmd:nnn {vardef} { m 0{} m} {
4751   \stex_keys_set:nn{vardef}{#2}
4752   \str_set:Nx \l_stex_macroname_str { #1 }
4753   \str_if_empty:NT \l_stex_key_name_str {
4754     \str_set:Nx \l_stex_key_name_str { #1 }
4755   }
4756
4757 \stex_symdecl_do:
4758 \_stex_symdecl_check_terms:
4759 \_stex_vars_add:
4760 \_stex_vars_macro:
4761 \stex_if_do_html:T \_stex_vars_html:
4762
4763 \int_set:Nn \l_stex_get_symbol_arity_int {\l_stex_get_symbol_arity_int}
4764 \stex_debug:nn{vardef}{Doing~\l_stex_key_name_str}
4765 \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4766 \stex_notation_parse:n{#3}
4767 \stex_if_check_terms:T{ \_stex_notation_check: }
4768 \_stex_vardecl_notation_macro:
4769 \stex_if_do_html:T {
4770   \def\comp{\_varcomp}
4771   \_stex_notation_do_html:n \l_stex_key_name_str
4772 }
4773 \group_begin:
4774 \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4775 \tl_clear:N \thisstyle
4776 \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4777 \def\thisnotation{
4778   \$\let\l_stex_current_symbol_str\thisvarname
4779   \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}
4780   \_stex_notation_make_args:
4781 }
4782 }
4783 \stex_style_apply:
4784 \group_end:\ignorespaces
4785 }{ }

4786 \cs_new_protected:Nn \_stex_vars_add: {
4787   \exp_args:NNNo \exp_args:NNnx
4788   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4789     {\l_stex_macroname_str}
4790     {\l_stex_key_name_str}
4791     {\int_use:N \l_stex_get_symbol_arity_int}
4792     {\l_stex_get_symbol_args_tl}

```

```

4794 {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4795 {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4796 {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4797 \stex_invoke_symbol:
4798 }
4799 }
4800
4801 \cs_new_protected:Nn \__stex_vars_macro: {
4802   \tl_set:cx{\l_stex_mroname_str}{%
4803     \stex_invoke_variable:nnnnnN
4804       {\l_stex_key_name_str}
4805       {\int_use:N \l_stex_get_symbol_arity_int}
4806       {\l_stex_get_symbol_args_tl}
4807       {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4808       {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4809       {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4810     \stex_invoke_symbol:
4811   }
4812 }
4813
4814 \cs_new_protected:Nn \__stex_vars_html: {
4815   \stex_if_do_html:T {
4816     \hbox\bgroup\exp_args:Ne \stex_annotation_invisible:nn {
4817       \shtml:vardef = {\l_stex_key_name_str},
4818       \shtml:args = {\l_stex_key_args_str}
4819       \str_if_empty:NF \l_stex_mroname_str {,
4820         \shtml:macroname={\l_stex_mroname_str}
4821       }
4822       \str_if_empty:NF \l_stex_key_assoc_str {,
4823         \shtml:assocotype={\l_stex_key_assoc_str}
4824       }
4825       \str_if_empty:NF \l_stex_key_role_str {,
4826         \shtml:role={\l_stex_key_role_str}
4827       }
4828       \str_if_empty:NF \l_stex_key_reorder_str {,
4829         \shtml:reorderargs={\l_stex_key_reorder_str}
4830       }
4831       \bool_if:NT \l_stex_vars_bind_bool {,
4832         \shtml:bind={}
4833       }
4834     }{
4835       \stex_annotation_force_break:n{
4836         \bool_set_true:N \stex_in_invisible_html_bool
4837         \tl_if_empty:NF \l_stex_key_type_tl {
4838           \stex_annotation:nn{\shtml:type={}}{$\l_stex_key_type_tl$}
4839         }
4840         \tl_if_empty:NF \l_stex_key_def_tl {
4841           \stex_annotation:nn{\shtml:definiens={}}{$\l_stex_key_def_tl$}
4842         }
4843         \tl_if_empty:NF \l_stex_key_return_tl{
4844           \exp_args:Nno \use:n{
4845             \cs_generate_from_arg_count:NNnn \l__stex_vars_cs
4846             \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4847             \tl_set:Nx \l_stex_vars_args_tl {\stex_map_args:N \stex_return_args:nn}

```

```

4848     $\\stex_annotate:nn{shtml:returnrntype={}){
4849         \\exp_after:wN \\l__stex_vars_cs \\l__stex_vars_args_tl!}$
4850     }
4851     \\tl_if_empty:NF \\l_stex_key_argtypes_clist {
4852         \\stex_annotate:nn{shtml:argtypes={}){
4853             \\_stex_annotate_force_break:n{
4854                 \\clist_map_inline:Nn \\l_stex_key_argtypes_clist {
4855                     $\\stex_annotate:nn{shtml:type={}{{##1}}$}
4856                 }
4857             }
4858         }
4859     }
4860 }
4861 }\\egroup
4862 }
4863 }
```

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

`_stex_vardecl_notation_macro:`

```

4864 \\cs_new_protected:Nn \\_stex_vardecl_notation_macro: {
4865     \\tl_set_eq:cN {l_stex_notation_
4866         \\l_stex_key_name_str _ 
4867         \\l_stex_key_variant_str _cs
4868     }\\l_stex_notation_macrocode_cs
4869     \\cs_if_exist:cF {l_stex_notation_\\l_stex_key_name_str __cs} {
4870         \\tl_set_eq:cN{l_stex_notation_\\l_stex_key_name_str __cs}
4871             \\l_stex_notation_macrocode_cs
4872     }
4873     \\tl_if_empty:NF \\l_stex_key_op_tl {
4874         \\tl_set_eq:cN {l_stex_notation_\\l_stex_key_name_str _op_
4875             \\l_stex_key_variant_str _cs}\\l_stex_key_op_tl
4876         \\cs_if_exist:cF {l_stex_notation_\\l_stex_key_name_str _op__cs} {
4877             \\cs_set_eq:cN{l_stex_notation_\\l_stex_key_name_str _op__cs}
4878             \\l_stex_key_op_tl
4879         }
4880     }
4881 }
```

(End of definition for `_stex_vardecl_notation_macro::`. This function is documented on page 115.)

`\stex_get_symbol_or_var:n`

`\stex_get_var:n`

```

4882 \\cs_new_protected:Nn \\_stex_vars_set_vars:nnnnnnN {
4883     \\stex_debug:nn{symbols}{Variable~#1~found}
4884     \\cs_set:Npn \\_stex_variable:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {}
4885     \\str_clear:N \\l_stex_get_symbol_mod_str
4886     \\str_set:Nn \\l_stex_get_symbol_name_str {#1}
4887     \\int_set:Nn \\l_stex_get_symbol_arity_int {#2}
4888     \\tl_set:Nn \\l_stex_get_symbol_args_tl {#3}
4889     \\tl_set:Nn \\l_stex_get_symbol_def_tl {#4}
4890     \\tl_set:Nn \\l_stex_get_symbol_type_tl {#5}
4891     \\tl_set:Nn \\l_stex_get_symbol_return_tl {#6}
4892     \\tl_set:Nn \\l_stex_get_symbol_invoke_cs {#7}
4893 }
```

```

4895 \cs_new_protected:Nn \__stex_vars_get_var:n {
4896   \prop_map_inline:Nn \l_stex_variables_prop {
4897     \__stex_vars_check_var:nnnnnnnnN {#1} ##2
4898   }
4899 }
4900
4901 \cs_new_protected:Nn \__stex_vars_check_var:nnnnnnnnN {
4902   \str_if_eq:nnTF{#1}{#2} {
4903     \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}}
4904   }{
4905     \str_if_eq:nnT{#1}{#3} {
4906       \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}}
4907     }
4908   }
4909 }
4910
4911 \cs_new_protected:Nn \stex_get_var:n {
4912   \str_clear:N \l_stex_get_symbol_name_str
4913   \__stex_vars_get_var:n{#1}
4914   \str_if_empty:NT \l_stex_get_symbol_name_str {
4915     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4916   }
4917 }
4918
4919 \cs_new_protected:Nn \stex_get_symbol_or_var:n {
4920   \str_clear:N \l_stex_get_symbol_name_str
4921   \__stex_vars_get_var:n{#1}
4922   \str_if_empty:NT \l_stex_get_symbol_name_str {
4923     \stex_debug:nn{symbols}{No~variable~#1~found}
4924     \stex_get_symbol:n{#1}
4925   }
4926 }

```

(End of definition for `\stex_get_symbol_or_var:n` and `\stex_get_var:n`. These functions are documented on page 115.)

\svar

```

4927 \NewDocumentCommand \svar {O{} m} {
4928   \group_begin:
4929   \tl_if_empty:nTF{#1} {
4930     \str_set:Nn \l_stex_current_symbol_str {#2}
4931   }{
4932     \str_set:Nn \l_stex_current_symbol_str {#1}
4933   }
4934   \bool_if:NTF \l_stex_allow_semantic_bool {
4935     \tl_clear:N \l_stex_current_term_tl
4936     \stex_term_omv:nnn{}{}{\l_varcomp{#2}}
4937   }{
4938     \msg_error:nnxx{stex}{error/notallowed}{Variable}{\l_stex_current_symbol_str}
4939   }
4940   \group_end:
4941 }

```

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

13.8.4 Sequences

```
4942 <@=stex_seqs>
4943 \stex_new_stylable_cmd:nnnn {varseq}{m O{} m m} {
4944   \stex_keys_set:nn{symdef}{#2}
4945   \str_set:Nx \l_stex_mroname_str { #1 }
4946   \str_if_empty:NT \l_stex_key_name_str {
4947     \str_set:Nx \l_stex_key_name_str { #1 }
4948   }
4949   \str_if_empty:NT \l_stex_key_args_str {
4950     \str_set:Nn \l_stex_key_args_str {1}
4951   }
4952 \stex_symdecl_do:
4953
4954 \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4955 \clist_set:Nn \l_stex_seqs_range_clist {#3}
4956 \tl_if_empty:NTF \l_stex_key_op_tl {
4957   \stex_notation_parse:n{#4}
4958   \tl_clear:N \l_stex_key_op_tl
4959 }{
4960   \stex_notation_parse:n{#4}
4961 }
4962 \stex_if_do_html:T \__stex_seqs_html:
4963 \stex_if_check_terms:T \__stex_seqs_check_terms:
4964 \__stex_seqs_add:
4965 \__stex_seqs_macro:
4966 \stex_if_check_terms:T \_stex_notation_check:
4967 \_stex_vardecl_notation_macro:
4968 \group_begin:
4969 \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4970 \tl_clear:N \thisstyle
4971 \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4972 \def\thisnotation{
4973   \$\let\l_stex_current_symbol_str\thisvarname
4974   \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}
4975   \__stex_seqs_make_args:
4976 }$
4977 }
4978 \stex_style_apply:
4979 \group_end:\ignorespaces
4980 }{}}
4981
4982 \cs_new_protected:Nn \__stex_seqs_add: {
4983   \exp_args:NNNo \exp_args:NNnx
4984   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4985     {\l_stex_mroname_str}
4986     {\l_stex_key_name_str}
4987     {\int_use:N \l_stex_get_symbol_arity_int}
4988     {\l_stex_get_symbol_args_tl}
4989     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4990     {\exp_args:No \exp_not:n \l_stex_seqs_range_clist}
4991     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4992   \stex_invoke_sequence:
```

```

4993     }
4994 }
4995
4996 \cs_new_protected:Nn \__stex_seqs_macro: {
4997   \tl_set:cx{\l_stex_mroname_str}{
4998     \stex_invoke_variable:nnnnnN
4999       {\l_stex_key_name_str}
5000       {\int_use:N \l_stex_get_symbol_arity_int}
5001       {\l_stex_get_symbol_args_tl}
5002       {\exp_args:No \exp_not:n \l_stex_key_def_tl}
5003       {\exp_args:No \exp_not:n \l_stex_seqs_range_clist}
5004       {\exp_args:No \exp_not:n \l_stex_key_return_tl}
5005       \stex_invoke_sequence:
5006   }
5007 }
5008
5009 \cs_new_protected:Nn \__stex_seqs_make_args: { \TODO }
5010 \cs_new_protected:Nn \__stex_seqs_check_terms: { \TODO }
5011
5012 \cs_new_protected:Nn \__stex_seqs_html: {
5013   \exp_args:Ne \stex_annotation_invisible:nn {
5014     \shtml:vareq = {\l_stex_key_name_str},
5015     \shtml:args = {\l_stex_key_args_str}
5016     \str_if_empty:NF \l_stex_mroname_str {,
5017       \shtml:macroname={\l_stex_mroname_str}
5018     }
5019     \str_if_empty:NF \l_stex_key_assoc_str {,
5020       \shtml:assocotype={\l_stex_key_assoc_str}
5021     }
5022     \str_if_empty:NF \l_stex_key_role_str {,
5023       \shtml:role={\l_stex_key_role_str}
5024     }
5025     \str_if_empty:NF \l_stex_key_reorder_str {,
5026       \shtml:reorderargs={\l_stex_key_reorder_str}
5027     }
5028   }{\hbox\bgroup
5029     \stex_annotation_force_break:n{
5030       \tl_if_empty:NF \l_stex_key_type_tl {
5031         \stex_annotation:nn{\shtml:type={}}{$\l_stex_key_type_tl$}
5032       }
5033       \tl_if_empty:NF \l_stex_key_def_tl {
5034         \stex_annotation:nn{\shtml:definiens={}}{$\l_stex_key_def_tl$}
5035       }
5036       \tl_if_empty:NF \l_stex_key_return_tl{
5037         \exp_args:Nno \use:n{
5038           \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs
5039           \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
5040           \tl_set:Nx \l_stex_seqs_args_tl {\l_stex_map_args:N \stex_return_args:nn}
5041           \stex_annotation:nn{\shtml:returntype={}}{
5042             $\exp_after:wN \l_stex_seqs_cs \l_stex_seqs_args_tl!$}
5043         }
5044       \tl_if_empty:NF \l_stex_key_argtypes_clist {
5045         \stex_annotation:nn{\shtml:argtypes={}}{
5046           \stex_annotation_force_break:n{

```

```

5047           \clist_map_inline:Nn \l_stex_key_argtypes_clist {
5048               \stex_annotation:nn{shml:type={}}{$##1$}
5049           }
5050       }
5051   }
5052 }
5053 }
5054 \egroup
5055 }

```

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

```

\stex_invoke_sequence:
5056 \cs_new_protected:Nn \stex_invoke_sequence: {
5057     \peek_charcode_remove:NTF ! {
5058         \peek_charcode:NTF [ \__stex_seqs_do_op:w { \__stex_seqs_do_op:w [] } ]
5059     }\__stex_seqs_do_first:
5060 }
5061
5062 \cs_new_protected:Npn \__stex_seqs_do_op:w [#1] {
5063     \cs_if_exist:cTF {l_stex_notation_\l_stex_current_symbol_str _op_#1_cs} {
5064         \stex_maybe_brackets:nn{\neginfpref}{%
5065             \stex_term_oms_or_omv:nnn{#1}{%
5066                 {\use:c{l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}}%
5067             }%
5068             \group_end:
5069         }%
5070         \__stex_seqs_get_index_notation:n{#1}%
5071         \peek_charcode:NTF [ \__stex_seqs_doop_range:w { \__stex_seqs_doop_range:w[] } ]
5072     }%
5073 }
5074
5075 \cs_new_protected:Npn \__stex_seqs_doop_range:w [#1] {
5076     \bool_set_true:N \l_stex_allow_semantic_bool
5077     \clist_clear:N \l_stex_seqs_clist
5078     \clist_map_function:NN \l_stex_current_type_tl \__stex_seqs_doop_arg:n
5079         \stex_annotation:nn{
5080             shml:term=OMV,
5081             shml:head={\l_stex_current_symbol_str},
5082             shml:notationid={}
5083         }%
5084         \l_stex_seqs_clist
5085     }%
5086     \group_end:
5087 }
5088
5089 \cs_new_protected:Nn \__stex_seqs_doop_arg:n {
5090     \tl_if_eq:nnTF{#1}{\ellipses}{%
5091         \clist_put_right:Nn \l_stex_seqs_clist {
5092             \ellipses
5093         }%
5094     }%
5095     \clist_put_right:Nn \l_stex_seqs_clist {
5096         \exp_args:No \str_if_eq:nnTF \l_stex_current_arity_str {1}{%

```

```

5097   \group_begin:
5098     \l__stex_seqs_cs \group_end: {#1}
5099
5100   }{
5101     \group_begin:
5102       \l__stex_seqs_cs \group_end: #1
5103     }
5104   }
5105 }
5106 }
5107
5108 \cs_new_protected:Nn \__stex_seqs_get_index_notation:n {
5109   \cs_if_exist:cTF {l_stex_notation} {\l_stex_current_symbol_str _#1_cs} {
5110     \cs_set_eq:Nc \l__stex_seqs_cs {l_stex_notation} {\l_stex_current_symbol_str _#1_cs}
5111   }
5112   \stex_do_default_notation:
5113   \cs_set_eq:NN \l__stex_seqs_cs \l_stex_default_notation
5114 }
5115 }
5116
5117
5118 \cs_new:Nn \__stex_seqs_do_first_arg:n {{\exp_not:n{## #1}}}
5119
5120 \cs_new_protected:Nn \__stex_seqs_do_first: {
5121   \exp_args:Nnx \use:nn{
5122     \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs \cs_set:Npn
5123     \l_stex_current_arity_str }{
5124       \tl_set:Nn \exp_not:N \l__stex_seqs_first_args_tl {
5125         \int_step_function:nN \l_stex_current_arity_str \__stex_seqs_do_first_arg:n
5126       }
5127       \exp_not:N \__stex_seqs_do_first_next:
5128     }
5129   \l__stex_seqs_cs
5130 }
5131
5132 \cs_new_protected:Nn \__stex_seqs_do_first_next: {
5133   \peek_charcode_remove:NTF ! {
5134     \peek_charcode:NTF [ \__stex_seqs_do_one:w {\__stex_seqs_do_one:w []}]
5135   }
5136   \peek_charcode:NTF [ \__stex_seqs_do_all:w {\__stex_seqs_do_all:w []}]
5137 }
5138 }
5139
5140 \cs_new_protected:Npn \__stex_seqs_do_one:w [#1] {
5141   \__stex_seqs_get_index_notation:n{#1}
5142   \stex_debug:nn{HERE~seq~one}{\meaning\l__stex_seqs_cs^~J\meaning\l__stex_seqs_first_args_t1}
5143   \exp_args:Nno\use:nn{\l__stex_seqs_cs\group_end:}\l__stex_seqs_first_args_t1
5144 }
5145
5146 \cs_new_protected:Npn \__stex_seqs_do_all:w [#1] {
5147   \stex_debug:nn{HERE~seq~all}{\meaning\l__stex_seqs_first_args_t1}
5148   \exp_args:Nno\use:nn{\l_stex_invoke_notation:w [#1]}\l__stex_seqs_first_args_t1
5149 }

```

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

```
\seqmap
5150 \cs_new_protected:Npn \seqmap #1 #2 {
5151   \symuse{Metatheory?sequence-expression}{\seqmap{#1}{#2}}%\l_tmpa_t1 {#2}
5152 }
```

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

13.8.5 Expressions

```
5153 <@=stex_expr>
      Various variables:
5154 \bool_new:N \l_stex_allow_semantic_bool
5155 \bool_set_true:N \l_stex_allow_semantic_bool
5156 \tl_new:N \l_stex_current_term_tl
5157 \tl_set:Nn \l_stex_every_symbol_tl {
5158   \bool_set_false:N \l_stex_allow_semantic_bool
5159 }
```

```
\_stex_next_symbol:n
5160 \tl_new:N \l__stex_expr_reset_tl
5161 \cs_new_protected:Nn \_stex_next_symbol:n {
5162   \tl_set:Nx \l_stex_every_symbol_tl {
5163     \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5164       \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
5165         \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5166       }
5167       \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5168         \exp_args:No \exp_not:n \l__stex_expr_reset_tl
5169       }
5170     }
5171     \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
5172       \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5173     }
5174     \exp_not:n{ \aftergroup \l__stex_expr_reset_tl }
5175     \exp_not:N \l_stex_every_symbol_tl
5176     \exp_not:n{ #1 }
5177   }
5178 }
5179 \cs_generate_variant:Nn \_stex_next_symbol:n {e}
```

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

\STEXinvisible

```
5180 \cs_new_protected:Npn \STEXinvisible #1 {
5181   \stex_annotation_invisible:n { #1 }
5182 }
```

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

Invoking Semantic Macros

```
\_stex_invoke_symbol:nnnnnnN
5183 \cs_new_protected:Nn \_stex_invoke_symbol:nnnnnnN {
5184     \bool_if:NTF \l_stex_allow_semantic_bool{
5185         \stex_if_html_backend:T{ifvmode\indent\fi}
5186         \__stex_expr_setup:nnnnnf{\_comp}{#1?#2}{#3}{#4}{#7}{#6}
5187         \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
5188         \tl_put_right:Nn \l_stex_current_redo_tl{
5189             \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
5190         }
5191         #8
5192     }{
5193         \msg_error:nnxx{stex}{error/notallowed}{#1?#2}{\l_stex_current_symbol_str}
5194     }
5195 }
5196 \cs_generate_variant:Nn \_stex_invoke_symbol:nnnnnnN {ooxooooN}
5197
5198 \cs_new_protected:Nn \__stex_expr_setup:nnnnnn {
5199     \group_begin:
5200     \tl_clear:N \l_stex_return_notation_tl
5201     \tl_set:Nn \l_stex_current_redo_tl {
5202         \let \this \stex_current_this:
5203         \def\comp{#1}
5204         \def\maincomp{\comp}
5205         \str_set:Nn \l_stex_current_symbol_str {#2}
5206         \str_set:Nn \l_stex_current_arity_str{ #3 }
5207         \tl_set:Nn \l_stex_current_args_tl{ #4 }
5208         \tl_set:Nn \l_stex_current_return_tl{ #5 }
5209         \tl_set:Nn \l_stex_current_type_tl{ #6 }
5210         \tl_clear:N \l_stex_current_term_tl
5211     }
5212     \tl_put_right:Nx \l_stex_current_redo_tl {
5213         \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5214     }
5215     \l_stex_current_redo_tl
5216 }
```

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

```
\_stex_invoke_variable:nnnnnn
5217 \cs_new_protected:Nn \_stex_invoke_variable:nnnnnn {
5218     \bool_if:NTF \l_stex_allow_semantic_bool{
5219         \stex_if_html_backend:T{ifvmode\indent\fi}
5220         \__stex_expr_setup:nnnnnf{\_varcomp}{#1}{#2}{#3}{#6}{#5}
5221         \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
5222         \tl_put_right:Nn \l_stex_current_redo_tl {
5223             \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
5224         }
5225         #7
5226     }{
5227         \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
5228     }
5229 }
```

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

\symuse

```
5230 \cs_new_protected:Npn \symuse #1 {
5231   \stex_get_symbol:n{#1}
5232   \exp_args:Nno \use:n {\_stex_invoke_symbol:oooooN
5233   \l_stex_get_symbol_mod_str
5234   \l_stex_get_symbol_name_str
5235   {\int_use:N \l_stex_get_symbol_arity_int}
5236   \l_stex_get_symbol_args_tl
5237   \l_stex_get_symbol_def_tl
5238   \l_stex_get_symbol_type_tl
5239   \l_stex_get_symbol_return_tl}
5240   \l_stex_get_symbol_invoke_cs
5241 }
```

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

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

```
5242 \cs_new_protected:Nn \stex_invoke_symbol: {
5243   \stex_debug:nn{expressions}{Invoking-\l_stex_current_symbol_str}
5244   \mode_if_math:TF \_stex_expr_invoke_math: \_stex_expr_invoke_text:
5245 }
5246
5247 \cs_new_protected:Nn \__stex_expr_invoke_text: {
5248   \stex_debug:nn{expressions}{text-mode}
5249   \peek_charcode_remove:NTF ! \_stex_expr_invoke_op_custom:n \__stex_expr_invoke_custom:n
5250 }
5251
5252 \cs_new_protected:Nn \__stex_expr_invoke_math: {
5253   \stex_debug:nn{expressions}{math-mode}
5254   \peek_charcode_remove:NTF !
5255     % operator
5256     \peek_charcode_remove:NTF * \_stex_expr_invoke_op_custom:n {
5257       % op notation
5258       \peek_charcode:NTF [ \__stex_expr_invoke_op_notation:w {
5259         \_stex_expr_invoke_op_notation:w []
5260       }
5261     }
5262   }{
5263     \peek_charcode_remove:NTF * \_stex_expr_invoke_custom:n {
5264       % normal
5265       \peek_charcode:NTF [ \_stex_invoke_notation:w {
5266         \_stex_invoke_notation:w []
5267       }
5268     }
5269   }
5270 }
```

Notations:

```
5271 \cs_new_protected:Npn \_stex_invoke_notation:w [#1] {
5272   \stex_debug:nn{expressions}{using-notiation-#1-for-\l_stex_current_symbol_str}
5273   \cs_if_exist:cTF{\l_stex_notation_\l_stex_current_symbol_str _#1_cs}{
```

```

5274 \tl_if_empty:NTF \l_stex_current_return_tl {
5275   \stex_debug:nn{expressions}{return~empty}
5276   \use:c{l_stex_notation_}\l_stex_current_symbol_str _#1_cs}{\group_end:\_stex_eat_exclam
5277 }{
5278   \stex_debug:nn{expressions}{return?}
5279   \exp_after:wN\exp_after:wN\exp_after:wN
5280   \_\_stex_expr_invoke_return_maybe:n
5281   \exp_after:wN\exp_after:wN\exp_after:wN
5282   {\cs:w l_stex_notation_}\l_stex_current_symbol_str _#1_cs \cs_end: {}}
5283 }
5284 }{
5285   \stex_do_default_notation:
5286   \tl_if_empty:NTF \l_stex_current_return_tl {
5287     \l_stex_default_notation{\group_end:\_stex_eat_exclamation_point:}
5288   }{
5289     \exp_after:wN
5290     \_\_stex_expr_invoke_return_maybe:n
5291     \exp_after:wN
5292     {\l_stex_default_notation {}}
5293   }
5294 }
5295 }
5296
5297 \cs_new_protected:Npn \_\_stex_expr_invoke_op_notation:w [#1] {
5298   \stex_debug:nn{expressions}{op~notation~for~\l_stex_current_symbol_str}
5299   \cs_if_exist:cTF{l_stex_notation_}\l_stex_current_symbol_str _op_#1_cs){
5300     \_\_stex_maybe_brackets:nn{\neginfpref}{%
5301       \_\_stex_term_oms_or_omv:nnn{#1}{}%
5302       {\use:c{l_stex_notation_}\l_stex_current_symbol_str _op_#1_cs}}%
5303     }
5304     \group_end:
5305   }{
5306     \int_compare:nNnTF \l_stex_current_arity_str = 0 {
5307       \tl_clear:N \l_stex_current_return_tl
5308       \_\_stex_invoke_notation:w [#1]
5309     }{
5310       \stex_do_default_notation_op:
5311       \_\_stex_maybe_brackets:nn{\neginfpref}{%
5312         \_\_stex_term_oms_or_omv:nnn{#1}{}%
5313         {\l_stex_default_notation}}
5314       }
5315       \group_end:
5316     }
5317   }
5318 }

```

Return:

```

5319 \cs_new_protected:Nn \_\_stex_expr_invoke_return_maybe:n {
5320   \tl_clear:N \l_\_stex_expr_return_args_tl
5321   \tl_set:Nn \l_\_stex_expr_return_this_tl {#1}
5322   \exp_args:Nnx \use:n {
5323     \cs_generate_from_arg_count:NNnn \_\_stex_expr_ret_cs
5324     \cs_set:Npn \l_stex_current_arity_str } {
5325       \int_step_function:nN \l_stex_current_arity_str \_\_stex_expr_return_arg:n

```

```

5326      \__stex_expr_invoke_return_next:
5327    }
5328  \__stex_expr_ret_cs
5329 }
5330
5331 \cs_new:Nn \__stex_expr_return_arg:n {
5332   \tl_put_right:Nn \exp_not:N \l__stex_expr_return_args_tl {{#### #1}}
5333 }
5334
5335 \cs_new_protected:Nn \__stex_expr_invoke_return_next: {
5336   \peekCharCode_remove:NTF ! {
5337     \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl \group_end:
5338   }\__stex_expr_invoke_return:
5339 }
5340
5341 \cs_new_protected:Nn \__stex_expr_invoke_return: {
5342   \tl_set:Nx \l__stex_expr_return_this_tl {
5343     \__stex_expr_return_notation:n {
5344       \exp_after:wN \exp_after:wN \exp_after:wN
5345       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN \exp_after:wN {
5346         \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl
5347       }
5348     }
5349   }
5350 \stex_debug:nn{return}{Notation:~\meaning\l__stex_expr_return_this_tl}
5351 \tl_put_left:Nx \l__stex_expr_return_this_tl {
5352   \group_begin:\exp_args:No \exp_not:n \l_stex_current_redo_tl
5353 }
5354 \exp_args:Nnx \use:n {
5355 \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
5356   \cs_set:Npn \l_stex_current_arity_str } {
5357     \exp_args:No \exp_not:n \l_stex_current_return_tl
5358   }
5359 \stex_debug:nn{return}{
5360   \meaning\__stex_expr_ret_cs^~J
5361   \meaning\l__stex_expr_return_this_tl^~J
5362   \exp_args:No \exp_not:n \l__stex_expr_return_args_tl^~J
5363 }
5364 \exp_args:Nnx \use:nn {
5365   \exp_after:wN \group_end: \__stex_expr_ret_cs
5366 }{
5367   \exp_args:No \exp_not:n \l__stex_expr_return_args_tl
5368   {
5369     \exp_args:No \exp_not:n \l__stex_expr_return_this_tl
5370     \group_end:
5371   }
5372 }
5373 }
5374
5375
5376 \cs_new_protected:Nn \__stex_expr_return_notation:n {
5377   \tl_if_empty:NTF \l_stex_return_notation_tl { #1 }{
5378     \l_stex_return_notation_tl
5379   }

```

```

5380 }
5381
      Custom Notations:
5382 \cs_new_protected:Nn \__stex_expr_invoke_op_custom:n {
5383   \stex_debug:nn{expressions}{custom~op}
5384   \bool_set_true:N \l_stex_allow_semantic_bool
5385   \stex_term_oms_or_omv:nnn{}{}{\maincomp{#1}}
5386   \group_end:
5387 }
5388
5389 \int_new:N \l_stex_expr_arg_counter_int
5390 \cs_new_protected:Nn \__stex_expr_invoke_custom:n {
5391   \stex_debug:nn{custom}{custom~notation~for~\l_stex_current_symbol_str}
5392   \stex_pseudogroup:nn{
5393     \bool_set_true:N \l_stex_allow_semantic_bool
5394     \prop_gclear:N \l_stex_expr_customs_prop
5395     \seq_gclear:N \l_stex_expr_customs_seq
5396     \int_gzero:N \l_stex_expr_arg_counter_int
5397     \tl_if_empty:NF \l_stex_current_args_tl {
5398       \exp_after:wn \__stex_expr_add_prop_arg:nnw \l_stex_current_args_tl \stex_args_end:
5399       \cs_set_eq:NN \arg \__stex_expr_arg:n
5400     }
5401     \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
5402     \cs_set_eq:NN \__stex_expr_do_ab_next:nnn \stex_term_oma:nnn
5403     \stex_map_args:N \__stex_expr_check_b:nn
5404     \__stex_expr_do_ab_next:nnn{}{}{#1}
5405   }{
5406     \prop_if_exist:NT \l_stex_expr_customs_prop {
5407       \prop_gset_from_keyval:Nn \exp_not:N \l_stex_expr_customs_prop {
5408         \prop_to_keyval:N \l_stex_expr_customs_prop
5409       }
5410     }
5411     \int_gset:Nn \l_stex_expr_arg_counter_int { \int_use:N \l_stex_expr_arg_counter_int}
5412     \seq_if_exist:NT \l_stex_expr_customs_seq {
5413       \seq_gset_split:Nnn \exp_not:N \l_stex_expr_customs_seq , {
5414         \seq_use:Nn \l_stex_expr_customs_seq ,
5415       }
5416     }
5417   }
5418   % TODO check that all arguments are present
5419   \group_end:
5420 }
5421
5422 \cs_new_protected:Npn \__stex_expr_add_prop_arg:nnw #1 #2 #3\stex_args_end: {
5423   \prop_gput:Nnn \l_stex_expr_customs_prop {#1} {}
5424   \seq_gput_right:Nn \l_stex_expr_customs_seq {#2}
5425   \tl_if_empty:nF{#3}{\__stex_expr_add_prop_arg:nnw #3 \stex_args_end:}
5426 }
5427
5428 \cs_new:Nn \__stex_expr_check_b:nn {
5429   \str_case:nn #2 {
5430     b {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \stex_term_omb:nnn}
5431     B {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \stex_term_omb:nnn}

```

```

5432     }
5433 }
5434
5435 \NewDocumentCommand \__stex_expr_arg:n {s O{} m} {
5436   \IfBooleanTF #1 {
5437     \stex_annotation_invisible:n{
5438       \__stex_expr_inner:nn{#2}{#3}
5439     }
5440   }{
5441     \__stex_expr_inner:nn{#2}{#3}
5442   }
5443 }
5444
5445 \cs_new_protected:Nn \__stex_expr_inner:nn {
5446   \tl_if_empty:nTF{#1}{%
5447     \int_gincr:N \l__stex_expr_arg_counter_int
5448     \exp_args:Ne \__stex_expr_check:nTF{ \int_use:N \l__stex_expr_arg_counter_int }{%
5449       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl
5450     }{%
5451       \__stex_expr_inner:nn{}
5452     }{ #2 }
5453   }{%
5454     \__stex_expr_check:nTF {#1}{%
5455       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5456     }{%
5457       \exp_args:No \str_case:nnTF \l_tmpb_tl {
5458         {a}{%
5459           \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{%
5460             \l_tmpa_tl X
5461           }%
5462           \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5463         }%
5464         {B}{%
5465           \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{%
5466             \l_tmpa_tl X
5467           }%
5468           \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5469         }%
5470       }{%
5471         \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5472       }{%
5473         \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5474       }%
5475     }%
5476   }%
5477 }%
5478
5479 \prg_new_conditional:Nnn \__stex_expr_check:n {TF} {
5480   \exp_args:NNe \prop_get:NnTF \l__stex_expr_customs_prop {#1} \l_tmpa_tl {%
5481     \tl_set:Nx \l_tmpb_tl {\seq_item:Nn \l__stex_expr_customs_seq {#1} }%
5482     \tl_if_empty:NTF \l_tmpa_tl {%
5483       \exp_args:NNe \prop_gput:Nnn \l__stex_expr_customs_prop
5484         { #1 }{X}
5485       \exp_args:No \str_case:nnF \l_tmpb_tl {%

```

```

5486 {a}{
5487   \tl_set:Nx \l_tmpa_tl{ #1 1 }
5488 }
5489 {B}){
5490   \tl_set:Nx \l_tmpa_tl{ #1 1 }
5491 }
5492 }{
5493   \tl_set:Nx \l_tmpa_tl{ #1 1 }
5494 }
5495 \prg_return_true:
5496 }{
5497   \prg_return_false:
5498 }
5499 }{
5500   \msg_error:nxxx{\stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5501   \prg_return_false:
5502 }
5503 }
5504 % #1 argnum #2 argmode #3 code
5505 \cs_new_protected:Nn \__stex_expr_arg_do:nnn {
5506   \stex_debug:nn{custom}{Doing~argument~#1~of~mode~#2:~\tl_to_str:n{#3}}
5507   \group_begin:
5508     \bool_set_true:N \l_stex_allow_semantic_bool
5509     \stex_term_arg:nnn {#2}{#1}{#3}
5510   \group_end:
5511 }
5512 \cs_generate_variant:Nn \__stex_expr_arg_do:nnn {oon}

```

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

Argument Handling and Annotating

```

\stex_term_arg:nnnnn
\stex_term_arg:nnn
5514 % 1: argnum 2: argmode 3: precedence 4: argname 5: code
5515 \cs_new_protected:Nn \stex_term_arg:nnnnn {
5516   \group_begin:
5517     \str_clear:N \l_stex_current_symbol_str
5518     \tl_clear:N \l_stex_current_term_tl
5519     \int_set:Nn \l_stex_notation_downprec { #3 }
5520     \bool_set_true:N \l_stex_allow_semantic_bool
5521     \stex_term_arg:nnn {#2}{#1}{#3}
5522     \tl_if_empty:nTF{#4}{#5}
5523     }{
5524       \stex_annotation:nn{mml:arg={#4}}{#5}
5525     }
5526   }
5527 }
5528 \group_end:
5529 }
5530
5531 \cs_new_protected:Nn \stex_term_arg:nnn {
5532   \stex_annotation:nn{ shtml:arg={#2}, shtml:argmode={#1}}{
5533     \stex_annotation_force_break:n{ #3 }

```

```
5534     }
5535 }
```

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

`_stex_term_arg_aB:nnnnn`

```
5536 \tl_set:Nn \_stex_expr_do_aB_clist: {
5537   \seq_use:Nn \l_stex_aB_args_seq {
5538     \mathpunct{\comp{,}{}}}
5539   }
5540 }
5541 \tl_set_eq:NN \_stex_term_do_aB_clist: \_stex_expr_do_aB_clist:
5542
5543 \int_new:N \l__stex_expr_count_int
5544 \cs_new_protected:Nn \_stex_term_arg_aB:nnnnn {
5545   \tl_if_empty:nTF{#5} {
5546     \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{}
5547   }{
5548     \seq_clear:N \l_stex_aB_args_seq
5549     \int_zero:N \l__stex_expr_count_int
5550     \clist_map_inline:nn{#5} {
5551       \_stex_expr_aB_arg:nnnnn{##1}{#1}{#2}{#3}{#4}
5552     }
5553     \_stex_term_do_aB_clist:
5554   }
5555 }
5556
5557 % 1: code 2: argnum 3: argmode 4: precedence 5: argname
5558 \cs_new_protected:Npn \_stex_expr_aB_arg:nnnnn #1 {
5559   \int_incr:N \l__stex_expr_count_int
5560   \_stex_expr_is_varseq:nTF{#1} {
5561     \exp_after:wN \exp_after:wN \exp_after:wN
5562     \_stex_expr_assoc_seq:nnnnnnn
5563     \exp_after:wN
5564     \_stex_expr_gobble:nnnnnnnn #1 \_stex_expr_end:
5565   }{
5566     \_stex_expr_is_seqmap:nTF{#1} {
5567       \exp_args:NNe \use:nn \_stex_expr_do_seqmap:nnnnn {\tl_tail:n{#1}}
5568     }{
5569       \_stex_expr_aB_simple_arg:nnnnn{#1}
5570     }
5571   }
5572 }
5573
5574 \cs_new:Npn \_stex_expr_gobble:nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8 #9 \_stex_expr_end: {
5575   {#2} #3 {#6}
5576 }
5577
5578 \cs_new_protected:Nn \_stex_expr_aB_simple_arg:nnnnn{
5579   \seq_put_right:Nx \l_stex_aB_args_seq {
5580     \_stex_term_arg:nnnnn{#2}\int_use:N\l__stex_expr_count_int}{#3}{#4}{#5}{}
5581     \exp_not:n{
5582       \tl_set_eq:NN \_stex_term_do_aB_clist: \_stex_expr_do_aB_clist:
```

```

5583 #1
5584 }
5585 }
5586 }
5587 }

5588 \cs_new_protected:Nn \_stex_is_sequentialized:n {
5589   \group_begin: #1 \group_end:
5590 }
5591 }

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

5592 \prg_new_conditional:Nnn \_stex_expr_is_varseq:n {TF} {
5593   \int_compare:nNnTF {\tl_count:n{#1}} = 1 {
5594     \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_head:n{#1}}
5595     \_stex_invoke_variable:nnnnnnN {
5596       \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No\tl_item:nn{#1}{8}}
5597         \stex_invoke_sequence:
5598         \prg_return_true:\prg_return_false:
5599     }\prg_return_false:
5600   }\prg_return_false:
5601 }

5602

5603 \prg_new_conditional:Nnn \_stex_expr_is_seqmap:n {TF} {
5604   \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
5605     \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\seqmap}
5606     \prg_return_true:\prg_return_false:
5607   }\prg_return_false:
5608 }

Sequence variable:

5609 % 1: name 2: arity 3: clist 4: argnum 5: argmode 6: precedence 7: argname
5610 \cs_new_protected:Nn \_stex_expr_assoc_seq:nnnnnnn {
5611   \group_begin:
5612     \seq_clear:N \l_stex_aB_args_seq
5613     \_stex_expr_assoc_make_seq:nnn{#1}{#3}{#2}
5614   \exp_args:NNe \use:nn \group_end: {
5615     \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5616       \_stex_is_sequentialized:n{
5617         \_stex_term_arg:nnnn{\#4\int_use:N\l_stex_expr_count_int}{\#5}{\#6}{\#7}{\#8}
5618           \bool_set_true:N \l_stex_allow_semantic_bool
5619           \str_set:Nn \exp_not:N \l_stex_current_symbol_str
5620             {\l_stex_current_symbol_str}
5621           \tl_if_empty:NF \l_stex_current_term_tl {
5622             \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5623               \exp_args:No \exp_not:n \l_stex_current_term_tl
5624             }
5625           }
5626           \stex_annotation:nn{
5627             \shtml:term=OMV,
5628             \shtml:head={#1},
5629             \shtml:notationid={}
5630           }{
5631             \_stex_annotation_force_break:n{
5632               \_stex_term_do_aB_clist:

```

```

5633         }
5634     }
5635   }
5636 }
5637 }
5638 }
5639 }
5640
5641 % #1: name, #2: clist, #3:arity
5642 \cs_new_protected:Nn \__stex_expr_assoc_make_seq:nnn {
5643   \cs_if_exist:cTF{l_stex_notation_#1__cs} {
5644     \cs_set_eq:Nc \l__stex_expr_cs {l_stex_notation_#1__cs}
5645   }{
5646     \stex_do_default_notation:
5647     \cs_set_eq:NN \l__stex_expr_cs \l_stex_default_notation
5648   }
5649 \clist_map_inline:nn{#2}{%
5650   \tl_if_eq:nnTF{##1}{\ellipses}{%
5651     \seq_put_right:Nn \l_stex_aB_args_seq { ##1 }
5652   }{
5653     \int_compare:nNnTF {#3} = 1 {
5654       \tl_set:Nn \l_stex_expr_iarg_tl { ##1 }
5655     }{
5656       \tl_set:Nn \l_stex_expr_iarg_tl { ##1 }
5657     }
5658     \seq_put_right:Nx \l_stex_aB_args_seq {
5659       \group_begin:
5660       \exp_not:n {
5661         \tl_set_eq:NN \stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5662         \def\comp{\varcomp}
5663         \str_set:Nn \l_stex_current_symbol_str{#1}
5664       }
5665       \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
5666       \exp_after:wN \exp_after:wN \exp_after:wN {
5667         \exp_after:wN \l_stex_expr_cs \exp_after:wN \group_end: \l_stex_expr_iarg_tl
5668       }
5669     }
5670   }
5671 }
5672 }

\seqmap:

5673 % 1: fun 2: clist 3: argnum 4: argmode 5: precedence 6: argname
5674 \cs_new_protected:Nn \__stex_expr_do_seqmap:nnnnnn {
5675   \group_begin:
5676   \cs_set:Npn \l_tmpa_cs ##1 {##1}
5677   \seq_clear:N \l_stex_aB_args_seq
5678   \clist_map_inline:nn{#2}{%
5679     \__stex_expr_is_varseq:nTF{##1}{%
5680       \exp_after:wN
5681       \__stex_expr_varseq_in_map:nnnnnnnn ##1
5682     }{
5683       \seq_put_right:Nn \l_stex_aB_args_seq {##1}
5684     }

```

```

5685 }
5686 \seq_clear:N \l__stex_expr_old_seq
5687 \seq_map_inline:Nn \l__stex_aB_args_seq {
5688   \tl_if_eq:nnTF{##1}{\ellipses}{
5689     \seq_put_right:Nn \l__stex_expr_old_seq {##1}
5690   }
5691   % TODO \stex_is_sequentialized:n
5692   {
5693     \exp_args:NNo \seq_put_right:Nn \l__stex_expr_old_seq {
5694       \l_tmpa_cs {##1}
5695     }
5696   }
5697 }
5698 \seq_set_eq:NN \l__stex_aB_args_seq \l__stex_expr_old_seq
5699 \exp_args:NNe \use:nn \group_end: {
5700   \seq_put_right:Nn \exp_not:N \l__stex_aB_args_seq {
5701     \stex_is_sequentialized:n{
5702       \stex_term_arg:nnnnn{#4}\int_use:N\l__stex_expr_count_int}{#4}{#5}{#6}{%
5703         \bool_set_true:N \l__stex_allow_semantic_bool
5704         \str_set:Nn \exp_not:N \l__stex_current_symbol_str
5705           {\l__stex_current_symbol_str}
5706         \tl_set:Nn \exp_not:N \l__stex_current_term_tl {
5707           \symuse{Metatheory?sequence-map}
5708           {\exp_args:No \exp_not:n \l_tmpa_tl}
5709           {\stex_term_do_aB_clist:}
5710         }
5711         \__stex_expr_do_headterm:nn{}{
5712           \stex_term_do_aB_clist:
5713         }
5714       }
5715     }
5716   }
5717 }
5718 }
5719
5720 \cs_new_protected:Nn \__stex_expr_varseq_in_map:nnnnnnnn {
5721   \__stex_expr_assoc_make_seq:nnn{#2}{#6}{#3}
5722 }

```

(End of definition for _stex_term_

```
\_stex_term_oms_or_omv:nnn  
  \_stex_term_oms:nnn  
  \_stex_term_omv:nnn
```

```

5733 \cs_new_protected:Nn \_stex_term_oms:nnn {
5734   \tl_if_empty:NTF \l_stex_current_term_tl {
5735     \stex_annotation:nn{
5736       shtml:term=OMID,
5737       shtml:head={\l_stex_current_symbol_str},
5738       shtml:notationid={#1},
5739     }{
5740       \_stex_annotation_force_break:n{#3}
5741     }
5742   }{
5743     \__stex_expr_do_headterm:nn{#1}{#3}
5744   }
5745 }
5746 \cs_new_protected:Nn \_stex_term_omv:nnn {
5747   \tl_if_empty:NTF \l_stex_current_term_tl {
5748     \stex_annotation:nn{
5749       shtml:term=OMV,
5750       shtml:head={\l_stex_current_symbol_str},
5751       shtml:notationid={#1}
5752     }{
5753       \_stex_annotation_force_break:n{#3}
5754     }
5755   }{
5756     \__stex_expr_do_headterm:nn{#1}{#3}
5757   }
5758 }
5759 \cs_new_protected:Nn \__stex_expr_do_headterm:nn {
5760   \bool_if:NTF \stex_in_invisible_html_bool {
5761     {\bool_set_true:N \l_stex_allow_semantic_bool
5762      \ensuremath{\l_stex_current_term_tl}}
5763   }{
5764     \stex_annotation:nn{
5765       shtml:term=complex,
5766       shtml:head={\l_stex_current_symbol_str},
5767       shtml:notationid={#1}
5768     }{
5769       \_stex_annotation_force_break:n{
5770         \stex_annotation_invisible:nn{shtml:headterm={}}{
5771           {\bool_set_true:N \l_stex_allow_semantic_bool
5772             \ensuremath{\l_stex_current_term_tl}}
5773           }
5774         }
5775       }
5776     }
5777     #2
5778   }
5779 }
5780 }
5781 }{
5782   \cs_new_protected:Nn \_stex_term_oms:nnn {#3}
5783   \cs_new_protected:Nn \_stex_term_omv:nnn {#3}
5784   \cs_new_protected:Nn \__stex_expr_do_headterm:nn { #2 }
5785 }
5786 \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
5787 \stex_if_html_backend:TF {
5788   \cs_new_protected:Nn \_stex_term_oma:nnn {
5789     \tl_if_empty:NTF \l_stex_current_term_tl {
5790       \stex_annotation:nn{
5791         shtml:term=OMA,
5792         shtml:head={\l_stex_current_symbol_str},
5793         shtml:notationid={#1}
5794       }{
5795         \_stex_annotation_force_break:n{#3}
5796       }
5797     }{
5798       \stex_annotation:nn{
5799         shtml:term=OMA,
5800         shtml:head={\l_stex_current_symbol_str},
5801         shtml:notationid={#1}
5802       }{
5803         \_stex_annotation_force_break:n{
5804           \stex_annotation_invisible:nn{shtml:headterm={}}{
5805             \bool_set_true:N \l_stex_allow_semantic_bool
5806             \l_stex_current_term_tl
5807           }
5808           #3
5809         }
5810       }
5811     }
5812   }{
5813   }{
5814     \cs_new_protected:Nn \_stex_term_oma:nnn {#3}
5815   }
```

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

```
\_stex_term_omb:nnn
5816 \stex_if_html_backend:TF {
5817   \cs_new_protected:Nn \_stex_term_omb:nnn {
5818     \tl_if_empty:NTF \l_stex_current_term_tl {
5819       \stex_annotation:nn{
5820         shtml:term=OMBIND,
5821         shtml:head={\l_stex_current_symbol_str},
5822         shtml:notationid={#1}
5823       }{
5824         \_stex_annotation_force_break:n{#3}
5825       }
5826     }{
5827       \stex_annotation:nn{
5828         shtml:term=OMBIND,
5829         shtml:head={\l_stex_current_symbol_str},
5830         shtml:notationid={#1}
5831       }{
5832         \_stex_annotation_force_break:n{
```

```

5833     \stex_annotation_invisible:nn{shml:headterm={}}{{
5834         \bool_set_true:N \l_stex_allow_semantic_bool
5835         \l_stex_current_term_tl
5836     }}
5837     #3
5838 }
5839 }
5840 }
5841 }
5842 }{
5843     \cs_new_protected:Nn \_stex_term_omb:nnn { #3 }
5844 }

```

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

Automated Bracketing

```

\infprec
\neginfprec
5845 \tl_const:Nx \infprec {\int_use:N \c_max_int}
5846 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}

(End of definition for \infprec and \neginfprec. These functions are documented on page 81.)

```

```

\dobrackets
5847 \int_new:N \l_stex_notation_downprec
5848 \int_set:Nn \l_stex_notation_downprec \infprec
5849 \tl_set:Nn \l_stex_expr_left_bracket_str (
5850 \tl_set:Nn \l_stex_expr_right_bracket_str )
5851 \bool_new:N \l_stex_brackets_dones_bool
5852
5853 \cs_new_protected:Nn \_stex_maybe_brackets:nn {
5854     \bool_if:NTF \l_stex_brackets_dones_bool {
5855         \bool_set_false:N \l_stex_brackets_dones_bool
5856         #2
5857     } {
5858         \stex_debug:n{brackets}{#1}\int_eval:n \l_stex_notation_downprec?
5859         \int_compare:nNnTF { #1 } > \l_stex_notation_downprec {
5860             \%bool_if:NTF \l_stex_inpararray_bool { #2 }{
5861                 \dobrackets {
5862                     #2
5863                 }
5864             %}
5865         }{
5866             #2
5867         }
5868     }
5869 }
5870 \%RequirePackage{scalerel}
5871 \cs_new_protected:Npn \dobrackets #1 {
5872     \%ThisStyle{\if D\m@switch
5873     % \exp_args:Nnx \use:nn
5874     % { \exp_after:wN \left\l_stex_expr_left_bracket_str #1 }
5875     % { \exp_not:N\right\l_stex_expr_right_bracket_str }

```

```

5877 % \else
5878 \group_begin:
5879 \%stex_pseudogroup_with:nn{\l_stex_brackets_dones_bool\l_stex_notation_downprec}%
5880   \bool_set_true:N \l_stex_brackets_dones_bool
5881   \%int_set:Nn \l_stex_notation_downprec \infprec
5882   \mathopen{\cs_if_exist:NT\l_stex_current_symbol_str\comp
5883     \l__stex_expr_left_bracket_str
5884   }
5885   #1
5886 \group_end:%
5887 \mathclose{\cs_if_exist:NT\l_stex_current_symbol_str\comp
5888   \l__stex_expr_right_bracket_str
5889 }
5890 \%fi}
5891 }

```

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

`\withbrackets`

```

5892 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
5893   \%stex_pseudogroup_with:nn{\l__stex_expr_left_bracket_str\l__stex_expr_right_bracket_str}%
5894     \tl_set:Nn \l__stex_expr_left_bracket_str { #1 }
5895     \tl_set:Nn \l__stex_expr_right_bracket_str { #2 }
5896     #3
5897   }
5898 }

```

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

`\dowithbrackets`

```

5899 \cs_new_protected:Npn \dowithbrackets #1 #2 #3 {
5900   \withbrackets{#1}{#2}{\dobrackets{#3}}
5901 }

```

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

Symname and Variants

```

\symname
\sn
\sns
\Symname
\Sn
\Sns
\symref
\sr
\varref
\varname
\Varname
\def\maincomp{\comp}
\stex_keys_define:nnnn{symname}%
  \tl_clear:N \l_stex_key_pre_tl
  \tl_clear:N \l_stex_key_post_tl
  \%tl_clear:N \l_stex_key_proot_tl
\{
  pre .tl_set:N = \l_stex_key_pre_tl ,
  post .tl_set:N = \l_stex_key_post_tl ,
  root .code:n = {}%.tl_set:N = \l_stex_key_root_tl
\}{}%
\NewDocumentCommand \symref { O{} m m } {
  \group_begin:
  \stex_keys_set:nn{symname}{#1}
  \stex_get_symbol:n{#2}
}

```

```

5918     \__stex_expr_do_ref:nNn{\#3}\symrefemph@uri\stex_term_oms:nnn
5919 }
5920 \let\sr\symref
5921
5922 \NewDocumentCommand \symname { O{} m} {
5923     \group_begin:
5924     \stex_keys_set:nn{symname}{#1}
5925     \stex_get_symbol:n{#2}
5926     \__stex_expr_do_ref:nNn{
5927         \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
5928     }\symrefemph@uri\stex_term_oms:nnn
5929 }
5930 \let\sn\symname
5931 \protected\def\sns{\symname[post=s]}
5932
5933 \NewDocumentCommand \Symname { O{} m} {
5934     \group_begin:
5935     \stex_keys_set:nn{symname}{#1}
5936     \stex_get_symbol:n{#2}
5937     \__stex_expr_do_ref:nNn{
5938         \l_stex_key_pre_tl\exp_after:wN\stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
5939     }\symrefemph@uri\stex_term_oms:nnn
5940 }
5941 \cs_new_protected:Nn \stex_capitalize:n {
5942     \uppercase{#1}
5943 }
5944 \let\Sn\Symname
5945 \protected\def\Sns{\Symname[post=s]}
5946
5947 \cs_new:Npn \stex_split_slash: #1/#2/#3\stex_args_end: {
5948     \tl_if_empty:nTF{#3}{
5949         #2
5950     }{
5951         \stex_split_slash: #2 / #3 \stex_args_end:
5952     }
5953 }
5954
5955 \NewDocumentCommand \varref { O{} m m} {
5956     \group_begin:
5957     \stex_keys_set:nn{symname}{#1}
5958     \stex_get_var:n{#2}
5959     \__stex_expr_do_ref:nNn{\#3}\varempth@uri{
5960         \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5961         \def\comp{\_varcomp}
5962         \stex_term_omv:nnn
5963     }
5964 }
5965
5966 \NewDocumentCommand \varname { O{} m} {
5967     \group_begin:
5968     \stex_keys_set:nn{symname}{#1}
5969     \stex_get_var:n{#2}
5970     \__stex_expr_do_ref:nNn{
5971         \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl

```

```

5972 } \varempm@uri{
5973   \str_set_eq:NN \l_stex_current_symbol_str \l_stex_get_symbol_name_str
5974   \def\comp{\_varcomp}
5975   \_stex_term_omv:nnn
5976 }
5977 }
5978
5979 \NewDocumentCommand \Varname { O{} m} {
5980   \group_begin:
5981   \stex_keys_set:nn{symname}{#1}
5982   \stex_get_var:n{#2}
5983   \_stex_expr_do_ref:nNn{
5984     \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
5985   } \varempm@uri{
5986     \str_set_eq:NN \l_stex_current_symbol_str \l_stex_get_symbol_name_str
5987     \def\comp{\_varcomp}
5988     \_stex_term_omv:nnn
5989   }
5990 }
5991
5992
5993 \cs_new_protected:Nn \_stex_expr_do_ref:nNn {
5994   \stex_if_html_backend:T{\ifvmode\indent\fi}
5995   \bool_if:NTF \l_stex_allow_semantic_bool{
5996     \str_set:Nx \l_stex_current_symbol_str
5997       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
5998     \str_if_in:NNT \l_stex_get_symbol_name_str / {
5999       \str_set:Nx \l_stex_get_symbol_name_str {
6000         \exp_after:wN \_stex_split_slash: \l_stex_get_symbol_name_str
6001         /\_stex_args_end:
6002       }
6003     }
6004     \tl_clear:N \l_stex_current_term_tl
6005     \def\comp{\_comp}
6006     \let\compemph@uri#2
6007     #3{}-{}\comp{#1}}
6008   }{
6009     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
6010   }
6011   \group_end:
6012 }

```

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

Highlighting

```
6013 <@=stex_notationcomps>
      \comp
\compemph@uri 6014 \cs_new_protected:Nn \_do_comp:nNn {
\compemph     6015 \stex_pseudogroup_with:nnf{\comp}{%
\defemph       6016 \def\comp##1##1}
\defemph@uri 6017 \str_if_empty:NTF \l_stex_current_symbol_str {
\symrefemph   6018 #3
\symrefemph@uri 6019 }{
```

```

6020     \stex_if_html_backend:TF {
6021         \stex_annotate:nn { shtml:#1 = \l_stex_current_symbol_str}{ #3 }
6022     }{
6023         \exp_args:Nno #2 { #3 } \l_stex_current_symbol_str
6024     }
6025 }
6026 }
6027 }
6028
6029 \cs_new_protected:Npn \_comp {
6030     \do_comp:nNn {comp}\compemph@uri
6031 }
6032
6033 \cs_new_protected:Npn \_varcomp {
6034     \do_comp:nNn {varcomp}\varempth@uri
6035 }
6036
6037 \cs_new_protected:Npn \_defcomp {
6038     \do_comp:nNn {definiendum}\defemph@uri
6039 }
6040
6041 \cs_set_protected:Npn \comp {}
6042
6043 \cs_new_protected:Npn \compemph@uri #1 #2 {
6044     \compemph{ #1 }
6045 }
6046
6047 \cs_new_protected:Npn \compemph #1 {
6048     #1
6049 }
6050
6051 \cs_new_protected:Npn \defemph@uri #1 #2 {
6052     \defemph{#1}
6053 }
6054
6055 \cs_new_protected:Npn \defemph #1 {
6056     \ifmmode\else\expandafter\textbf\fi{#1}
6057 }
6058
6059 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
6060     \symrefemph{#1}
6061 }
6062
6063 \cs_new_protected:Npn \symrefemph #1 {
6064     \emph{#1}
6065 }
6066
6067 \cs_new_protected:Npn \varempth@uri #1 #2 {
6068     \varempth{#1}
6069 }
6070
6071 \cs_new_protected:Npn \varempth #1 {
6072     #1
6073 }

```

(End of definition for `\comp` and others. These functions are documented on page 80.)

13.9 Mathematical Structures

6074 `<@=stex_structures>`

`\this`

```
6075 \cs_new_protected:Npn \stex_current_this: {
6076   { \bool_set_true:N \l_stex_allow_semantic_bool
6077     \tl_if_empty:NTF \l_stex_current_this_tl {}{
6078       \str_set:Nx \l_stex_current_symbol_str {
6079         \l_stex_current_module_str ? \l__stex_structures_name_str
6080       }
6081       \maincomp{\l_stex_current_this_tl}
6082     }
6083   }
6084 }
6085 \let \this \stex_current_this:
```

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

`mathstructure (env.)`

```
6086 \stex_new_stylable_env:nnnnnnn {mathstructure}{m 0{} }{
6087   \__stex_structures_begin:nn{#1}{#2}
6088   \stex_smsmode_do:
6089 }{
6090   \stex_structural_feature_module_end:
6091   \__stex_structures_do_externals:
6092 }{}{}{}
6093
6094 \stex_keys_define:nnnn{mathstructure}){
6095   \tl_clear:N \l_stex_current_this_tl
6096   \str_clear:N \l__stex_structures_name_str
6097 }{
6098   this .tl_set:N = \l_stex_current_this_tl ,
6099   unknown .code:n = {
6100     \str_if_empty:NTF \l_keys_key_str {
6101       \str_set:Nx \l__stex_structures_name_str {\l_keys_key_tl}
6102     }{
6103       \str_set_eq:NN \l__stex_structures_name_str \l_keys_key_str
6104     }
6105   }
6106 }{}{
6107
6108 \cs_new_protected:Nn \__stex_structures_begin:nn {
6109   \stex_keys_set:nn {mathstructure}{#2}
6110   \str_if_empty:NT \l__stex_structures_name_str {
6111     \str_set:Nn \l__stex_structures_name_str {#1}
6112   }
6113   \def\comp{\_comp}
6114
6115 \exp_args:Nne \use:nn { \stex_module_add_symbol:nnnnnnN }
6116   { {#1}{\l__stex_structures_name_str}{0}{}{defed}{{
6117     \l_stex_current_module_str / \l__stex_structures_name_str-module
```

```

6118     }}
6119     {} \stex_invoke_structure:
6120 \str_set:Nx \l_stex_mroname_str {#1}
6121 \stex_execute_in_module:x{
6122     \seq_clear:c {l_stex_structures_macros_} \l_stex_current_module_str / \l_stex_structures_na
6123     \seq_put_right:cn {l_stex_structures_macros_} \l_stex_current_module_str / \l_stex_structur
6124 }
6125 \exp_args:No \stex_structural_feature_module:nn
6126     {\l_stex_structures_name_str}{structure}
6127 }
6128
6129 \stex_sms_allow_env:n{mathstructure}
6130 \stex_deactivate_macro:Nn \mathstructure {module-environments}
6131 \stex_every_module:n {\stex_reactivate_macro:N \mathstructure}
6132
6133 \cs_new_protected:Nn \__stex_structures_do_externals: {
6134     \tl_set:Nn \l_stex_structures_replace_this_tl {####1}
6135     \exp_args:No \stex_iterate_symbols:nnf {g_stex_last_feature_str} {
6136         \__stex_structures_external_decl:nnnn {##5}{##4}{##3}{##8}
6137     }
6138 }
6139
6140 \cs_new_protected:Nn \__stex_structures_external_decl:nnnn {
6141     \%stex_debug:nn{structure}%
6142     % Generating~external~declaration~\l_stex_structures_name_str/#3-in-
6143     % \l_stex_current_module_str^{^J}
6144     % \tl_to_str:n{#1}^{^J}\tl_to_str:n{#2}^{^J}\tl_to_str:n{#4}
6145     %
6146     \%tl_set:Nn \l_stex_get_symbol_args_tl {#1}
6147     \%exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnN} {
6148         {}{\l_stex_structures_name_str/#3}{\int_eval:n{#2 + 1}}
6149         % {ii}\tl_if_empty:nF{#1}{\stex_map_args:N \__stex_structures_shift_argls:nn}
6150         % {defed}{typed}
6151         %}{#4}\stex_invoke_outer_field:
6152     }
6153
6154 \cs_new:Nn \__stex_structures_shift_argls:nn {
6155     \int_eval:n{#1+1}#2
6156 }

```

\stex_get_mathstructure:n

```

6157 \cs_new_protected:Nn \stex_get_mathstructure:n {
6158     \stex_get_mathstructure:n{#1}
6159     \str_if_empty:NT \l_stex_get_structure_module_str {
6160         \msg_error:nnn{stex}{error/unknownstructure}{#1}
6161     }
6162 }
6163 \cs_new_protected:Nn \stex_get_mathstructure:n {
6164     \str_clear:N \l_stex_get_structure_module_str
6165     \stex_get_symbol:n{#1}
6166     \str_if_empty:NF \l_stex_get_symbol_name_str {
6167         \exp_args:No \tl_if_eq:NNT \l_stex_get_symbol_invoke_cs \stex_invoke_structure: {
6168             \str_set_eq:NN \l_stex_get_structure_module_str \l_stex_get_symbol_type_tl
6169         }

```

```
6170     }
6171 }
```

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

`extstructure (env.)`

```
6172 \stex_new_stylable_env:nnnnnnn {extstructure}{m 0{} m} {
6173   \seq_clear:N \l__stex_structures_imports_seq
6174   \clist_map_inline:nn{#3} {
6175     \stex_get_mathstructure:n{##1}
6176     \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6177       \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{####1}}{
6178         \seq_put_right:Nn \l__stex_structures_imports_seq{####1}
6179       }
6180     }
6181     \stex_execute_in_module:x{
6182       \seq_put_right:cn{\l_stex_structure_macros_ \l_stex_get_structure_module_str _seq}{#1}
6183     }
6184   }
6185   \__stex_structures_begin:nn{#1}{#2}
6186   \seq_map_inline:Nn\l__stex_structures_imports_seq{
6187     \stex_if_do_html:T {
6188       \hbox{\stex_annotation_invisible:nn
6189         {shtml:import={##1}} {}}
6190     }
6191     \stex_module_add_morphism:nonn
6192       {}{##1}{import}{}
6193     \stex_execute_in_module:x{
6194       \stex_activate_module:n{##1}
6195     }
6196   }
6197   \stex_smsmode_do:
6198 }{
6199   \stex_structural_feature_module_end:
6200   \__stex_structures_do_exernals:
6201 }{}{}{}
```

6202
6203 \stex_sms_allow_env:n{extstructure}
6204 \stex_deactivate_macro:Nn \extstructure {module-environments}
6205 \stex_every_module:n {
6206 \stex_reactivate_macro:N \extstructure
6207 }
6208
6209 \cs_new:Nn __stex_structures_extend_structure_i:NnnnnnnN {
6210 \exp_not:n{#1{#2}{#3}{#4}{#5}{defed}}{\l__stex_structures_extmod_str,#7}\exp_not:n{#8}{#9}
6211 }
6212 \cs_new_protected:Nn __stex_structures_extend_structure:nn {
6213 \stex_debug:nn{ext}{Extending~#1~by~#2}
6214 \str_set:Nn \l__stex_structures_extmod_str{#2}
6215 \tl_set:cx{#1} {
6216 \exp_after:wN \exp_after:wN \exp_after:wN
6217 __stex_structures_extend_structure_i:NnnnnnnN \cs:w #1 \cs_end:
6218 }
6219 }

```

6220 \stex_new_stylable_env:nnnnnnn {extstructure*}{m} {
6221   \__stex_structures_new_extstruct_name:
6222   \seq_clear:N \l__stex_structures_imports_seq
6223   \stex_get_mathstructure:n{#1}
6224
6225
6226   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6227     \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6228       \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6229     }
6230   }
6231
6232   \stex_execute_in_module:x{
6233     \seq_map_inline:cn{l_stex_structure_macros_\l_stex_get_structure_module_str _seq} {
6234       \exp_not:N \tl_if_exist:cT{####1} {
6235         \__stex_structures_extend_structure:nn{####1}{\l_stex_current_module_str/\l__stex_st
6236       }
6237     }
6238   }
6239
6240   \exp_args:No \__stex_structures_begin:nn\l__stex_structures_exstruct_name_str{}
6241
6242   \seq_map_inline:Nn \l__stex_structures_imports_seq {
6243     \stex_if_do_html:T {
6244       \stex_annotation_invisible:nn
6245         {shtml:import= {##1}} {}
6246     }
6247     \stex_module_add_morphism:nonn
6248       {}{##1}{import} {}
6249     \stex_execute_in_module:x{
6250       \stex_activate_module:n{##1}
6251     }
6252   }
6253
6254   \stex_smsmode_do:
6255 }{
6256   \prop_map_inline:cn{
6257     c_stex_module_ \l_stex_current_module_str _symbols_prop
6258   }{
6259     \__stex_structures_check_def:nnnnnnn ##2
6260   }
6261   \stex_structural_feature_module_end:
6262   \__stex_structures_do_externals:
6263 }{}{}{}
6264
6265 \stex_sms_allow_env:n{extstructure*}
6266 \exp_after:wN \stex_deactivate_macro:Nn
6267   \cs:w extstructure*\cs_end: {module-environments}
6268 \stex_every_module:n {
6269   \exp_after:wN \stex_reactivate_macro:N \cs:w extstructure*\cs_end:
6270 }
6271
6272 \cs_new_protected:Nn \__stex_structures_check_def:nnnnnnn {
6273   \tl_if_empty:nT{#5}{
```

```

6274     \msg_error:nnnn{stex}{error/needsdefiniens}{#2}{extstructure*}
6275   }
6276 }
6277
6278 \stex_every_module:n{ \str_set:Nn \l__stex_structures_extname_count 0}
6279
6280 \cs_new_protected:Nn \__stex_structures_new_extstruct_name: {
6281   \stex_do_up_to_module:n {
6282     \str_set:Nx \l__stex_structures_extname_count {\int_eval:n{\l__stex_structures_extname_c
6283   }
6284   \str_set:Nx \l__stex_structures_exstruct_name_str {EXTSTRUCT_\l__stex_structures_extname_c
6285 }
6286

```

Invoking structures:

```

6287 \cs_new_protected:Nn \stex_invoke_structure: {
6288   \tl_set:Nn \l__stex_structures_set_comp_tl {\__stex_structures_set_thiscomp:}
6289   \__stex_structures_invoke_top:n {}
6290 }
6291
6292 \cs_new_protected:Nn \__stex_structures_invoke_top:n {
6293   \stex_debug:nn{structure}{
6294     invoking~structure~{\l_stex_current_type_tl}<\tl_to_str:n{#1}>
6295   }
6296   \peekCharCode:NTF [ {
6297     \__stex_structures_merge:nw{#1}
6298   }{
6299     \__stex_structures_invocation_type:n {#1}
6300     \tl_set:Nn \l__stex_structures_this_tl {}
6301     \peekCharCode_remove:NTF ! {
6302       \peekCharCode:NTF [ {
6303         \__stex_structures_maybe_notation:w
6304       }{
6305         \__stex_structures_maybe_notation:w []
6306       }
6307     }{
6308       \__stex_structures_invoke_this:n
6309     }
6310   }
6311 }
6312
6313 \cs_new_protected:Npn \__stex_structures_merge:nw #1 [ #2 ] {
6314   \exp_args:Ne \stex_if_starts_with:nnTF {\tl_to_str:n{#2}}{comp=}{%
6315     \__stex_structures_set_customcomp: #2 \__stex_structures_end:
6316     \__stex_structures_invoke_top:n{#1}
6317   }{
6318     \exp_args:Ne \stex_if_starts_with:nnTF {\tl_to_str:n{#2}}{this=}{%
6319       \__stex_structures_set_thisnotation: #2 \__stex_structures_end:
6320       \__stex_structures_invoke_top:n{#1}
6321     }{
6322       \tl_if_empty:nTF{#1}{%
6323         \__stex_structures_invoke_top:n{#2}
6324       }{
6325         \tl_if_empty:nTF{#2}{%

```

```

6326     \__stex_structures_invoke_top:n{#1}
6327   }{
6328     \__stex_structures_invoke_top:n{#1,#2}
6329   }
6330 }
6331 }
6332 }
6333 }
6334
6335 \cs_new_protected:Npn \__stex_structures_set_thisnotation: this= #1 \__stex_structures_end:
6336   \tl_set:Nn \l_stex_return_notation_tl { \comp{#1} }
6337   \tl_set:Nn \l__stex_structures_set_comp_tl {}
6338 }
6339
6340 \cs_new_protected:Npn \__stex_structures_set_customcomp: comp= #1 \__stex_structures_end: {
6341   \tl_set:Nn \l__stex_structures_set_comp_tl {
6342     \__stex_structures_set_custom_comp:n{#1}
6343   }
6344   \tl_set:Nn \l_stex_return_notation_tl { \comp{} }
6345 }

```

The structure type:

```

6346 \cs_new_protected:Nn \__stex_structures_invokation_type:n {
6347   \__stex_structures_do_assign_list:n{#1}
6348   \clist_if_empty:NTF \l__stex_structures_fields_clist {
6349     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6350       = 1 {
6351         \tl_set:Nx \l__stex_structures_current_type_tl {
6352           \exp_args:No \exp_not:n \l_stex_current_redo_tl
6353           \stex_term_oms_or_omv:nnn{}{}{}
6354         }
6355       }{
6356         \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6357       }
6358   }{
6359     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6360       = 1 {
6361         \__stex_structures_make_type:n {}
6362       }{
6363         \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6364       }
6365   }
6366 }
6367
6368 \cs_new_protected:Nn \__stex_structures_do_assign_list:n {
6369   \clist_clear:N \l__stex_structures_fields_clist
6370   \tl_if_empty:nF {#1} {
6371     \keyval_parse:NNn\TODO\__stex_structures_do_assign:nn{#1}
6372   }
6373 }
6374
6375 \cs_new_protected:Nn \__stex_structures_do_assign:nn {
6376   \clist_put_right:Nn \l__stex_structures_fields_clist {{#1}{#2}}
6377 }
6378

```

```

6379 \cs_new_protected:Nn \__stex_structures_make_type:n {
6380   \tl_if_empty:nTF{#1} {
6381     \seq_clear:N \l_tmpa_seq
6382   }{
6383     \seq_set_split:Nnn \l_tmpa_seq ,{#1}
6384     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
6385     \seq_reverse:N \l_tmpa_seq
6386   }
6387   \tl_set:Nx \l__stex_structures_current_type_tl {
6388     \symuse{Metatheory?module-type~merge}{}
6389   {
6390     \exp_args:No \exp_not:n \l_stex_current_redo_tl
6391     \__stex_term_oms_or_omv:nnn{}{}{}
6392   }
6393   \seq_map_function:NN \l_tmpa_seq \__stex_structures_make_mod:n
6394   \clist_if_empty:NF \l__stex_structures_fields_clist {
6395     ,\symuse{Metatheory?anonymous-record}{}
6396     \exp_args:Ne \tl_tail:n{
6397       \clist_map_function:NN \l__stex_structures_fields_clist \__stex_structures_make_
6398     }
6399   }
6400 }
6401 }
6402 }
6403 }
6404
6405 \cs_new:Nn \__stex_structures_make_mod:n {
6406   ,\symuse{Metatheory?module-type}{}
6407   \stex_annotation:nn{shtml:term=OMMOD,shtml:head={#1}}{}
6408 }
6409 }
6410
6411 \cs_new:Nn \__stex_structures_make_oml:n {
6412   \__stex_structures_make_oml:nn #1
6413 }
6414 \cs_new:Nn \__stex_structures_make_oml:nn {
6415   ,\stex_annotation:nn{
6416     shtml:term=OML,
6417     shtml:head={#1}
6418   }{
6419     \__stex_annotation_force_break:n{
6420       \stex_annotation:nn{shtml:definiens={}}{\exp_not:n{#2!}}
6421     }
6422   }
6423 }

```

Insert the structure type as a term:

```

6424 \cs_new:Nn \__stex_structures_current_type: {
6425   \%exp_args:No \exp_not:n \l_stex_current_redo_tl
6426   \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6427     \exp_args:No\exp_not:n\l__stex_structures_current_type_tl
6428   }
6429   \__stex_term_oms_or_omv:nnn{}{}{}
6430 }

```

The structure type itself:

```
6431 \cs_new_protected:Npn \__stex_structures_maybe_notation:w [ #1 ] {
6432   \tl_set_eq:NN \l_stex_current_term_tl \l_stex_structures_current_type_tl
6433   \cs_if_exist:cTF{\l_stex_notation_\l_stex_current_symbol_str _#1_cs} {
6434     \use:c{\l_stex_notation_\l_stex_current_symbol_str _#1_cs}\group_end:
6435   }{
6436     \__stex_structures_make_prop:
6437     \__stex_structures_make_prop_assign:
6438     \__stex_structures_present_i:w [#1]
6439   }
6440 }
6441
6442 \cs_new_protected:Nn \__stex_structures_present: {
6443   \peek_charcode:NTF [ {
6444     \__stex_structures_present_i:w
6445   }{
6446     \__stex_structures_present:nn{}{}
6447   }
6448 }
6449
6450 \cs_new_protected:Npn \__stex_structures_present_i:w [#1] {
6451   \int_compare:nNnTF{\clist_count:n{#1}} = 1 {
6452     \__stex_structures_present:nn{}{#1}
6453   }{
6454     \peek_charcode:NTF [ {
6455       \__stex_structures_present_ii:nw{#1}
6456     }{
6457       \__stex_structures_present:nn{#1}{}
6458     }
6459   }
6460 }
6461
6462 %First: clist, second:notation-id
6463 \cs_new_protected:Npn \__stex_structures_present_ii:nw #1 [#2] {
6464   \__stex_structures_present:nn{#1}{#2}
6465 }
6466
6467 \cs_new_protected:Nn \__stex_structures_present:nn {
6468   \clist_clear:N \l_stex_structures_clist
6469   \tl_if_empty:nTF{#1} {
6470     \cs_set:Npn \l_stex_structures_cs ##1 ##2 ##3 {
6471       \tl_if_empty:nF{##2} {
6472         \__stex_structures_present_entry:nn {##1}{##3}
6473       }
6474     }
6475   }{
6476     \cs_set:Npn \l_stex_structures_cs ##1 ##2 ##3 {
6477       \exp_args:Ne \clist_if_in:nnT{\tl_to_str:n{#1}}{##1} {
6478         \__stex_structures_present_entry:nn {##1}{##3}
6479       }
6480     }
6481   }
6482 \prop_map_inline:Nn \l_stex_structures_prop {
6483   \l_stex_structures_cs {##1} ##2
```

```

6484   }
6485   \_stex_term_oms_or_omv:nnn{}{}{
6486     \exp_args:Nno \use:n{
6487       \bool_set_true:N \l_stex_allow_semantic_bool
6488       \symuse{Metatheory?mathematical-structure}[#2]
6489     }{\l__stex_structures_clist}
6490   }\group_end:
6491 }
6492
6493 \cs_new_protected:Nn \__stex_structures_present_entry:nn {
6494   \seq_if_in:NnTF \l_stex_structures_assigned_seq {#1}{
6495     \clist_put_right:Nn \l_stex_structures_clist {#2!}
6496   }{
6497     \exp_args:NNe \clist_put_right:Nn \l_stex_structures_clist {
6498       \stex_next_symbol:n {
6499         \exp_args:No \exp_not:n \l_stex_structures_set_comp_tl
6500         \tl_set:Nn \exp_not:N \l_stex_structures_this_tl {
6501           \exp_args:No \exp_not:n \l_stex_structures_this_tl
6502         }
6503         \exp_not:n {
6504           \tl_set_eq:NN \this \l_stex_structures_this_tl
6505         }
6506         \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6507           \exp_args:No \exp_not:n \l_stex_return_notation_tl
6508         }
6509       }
6510       \exp_not:n{#2!}
6511     }
6512   }
6513 }
6514
6515 \cs_new_protected:Npn \_thiscomp #1 #2 {
6516   {\tl_set:cn{\this}{\{}}#1{#2}\c_math_subscript_token{
6517     \group_begin:
6518       \bool_set_true:N \l_stex_allow_semantic_bool
6519       \l_stex_structures_this_tl
6520     \group_end:
6521   }
6522 }
6523 }
6524
6525 \cs_new_protected:Nn \__stex_structures_set_thiscomp: {
6526   \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_thiscomp {
6527     \edef\maincomp {\_thiscomp\comp}
6528   }
6529 }
6530
6531 \cs_new_protected:Nn \__stex_structures_set_custom_comp:n {
6532   \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
6533     \cs_set_protected:Npx \customthiscomp ##1 {
6534       \group_begin:
6535         \bool_set_true:N \l_stex_allow_semantic_bool
6536         \exp_not:n{
6537           \cs_set:Npn \l_stex_structures_comp_cs ##1 {

```

```

6538          #1
6539      }
6540      \def\maincomp
6541      {\comp}
6542      \exp_not:N\l_stex_structures_comp_{\comp{##1}}
6543      \group_end:
6544    }
6545    \def\maincomp {\_customthiscomp}
6546  }
6547 }
6548

this (of type structure):
6549 \cs_new_protected:Nn \__stex_structures_invoke_this:n {
6550   \peekCharCode_remove:NTF ! {
6551     \exp_args:Nne\use:nn{
6552       \group_end:\symuse{Metatheory?of~type}[invisible]{#1}
6553     }{
6554       {\__stex_structures_current_type:}
6555     }
6556   }{
6557     \__stex_structures_invoke_maybe_field:nn{#1}
6558   }
6559 }
6560 }

6561 \cs_new_protected:Nn \__stex_structures_invoke_maybe_field:nn {
6562   \__stex_structures_make_prop:
6563   \__stex_structures_set_this:n{#1}
6564   \tl_if_empty:nTF{#2} {
6565     \__stex_structures_make_prop_assign:
6566     \__stex_structures_present:
6567   }{
6568     \__stex_structures_invoke_field:n{#2}
6569   }
6570 }
6571 }

6572 \cs_new_protected:Nn \__stex_structures_set_this:n {
6573   \tl_if_empty:nTF{#1} {
6574     \%tl_put_right:Nn \l_stex_current_redo_tl {
6575       \%tl_clear:N \l_stex_structures_this_tl
6576     }
6577   }{
6578     \tl_set:Nx \l_stex_structures_this_tl {{%
6579       \bool_set_true:N \l_stex_allow_semantic_bool
6580       \tl_set:Nn \exp_not:N \this {
6581         \exp_args:No \exp_not:n \this
6582       }
6583       \exp_not:n{#1}
6584     }%
6585     \tl_set_eq:NN \this \l_stex_structures_this_tl
6586     \%l_stex_return_notation_tl
6587   }
6588 }
6589 }

6590

```

```

6591 \cs_new_protected:Nn \__stex_structures_get_field_name:n {
6592   \str_set:Nx \l__stex_structures_field_name_str {
6593     \exp_args:Nne \use:n {\exp_after:wN \use_i:nn \use:n}
6594     {\prop_item:Nn \l__stex_structures_prop {#1}}
6595   }
6596   \str_if_empty:NT \l__stex_structures_field_name_str {
6597     \str_set:Nn \l__stex_structures_field_name_str {#1}
6598   }
6599 }
6600
6601 \cs_new_protected:Nn \__stex_structures_invoke_field:n {
6602   \prop_if_in:NnTF \l__stex_structures_prop {#1} {
6603     \__stex_structures_get_field_name:n{#1}
6604     \tl_clear:N \l__stex_structures_more_nextsymbol_tl
6605     \%exp_args:NNe \seq_if_in:NnF \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}{
6606       \tl_set:Nx \l__stex_structures_more_nextsymbol_tl {
6607         \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
6608           \exp_args:No \exp_not:n \l__stex_structures_this_tl
6609         }
6610         \exp_not:n {
6611           \tl_set_eq:NN \this \l__stex_structures_this_tl
6612         }
6613       \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6614         \exp_args:No \exp_not:n \l_stex_return_notation_tl
6615       }
6616       \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
6617     }
6618   }
6619 \%}
6620 \exp_args:NNx \use:nn \group_end: {
6621   \stex_next_symbol:n {
6622     \exp_args:No \exp_not:n \l__stex_structures_redo_tl
6623     \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6624       \symuse{Metatheory?record~field}{%
6625         \symuse{Metatheory?of~type}{%
6626           \exp_args:No \exp_not:n \l__stex_structures_this_tl
6627         }{ \__stex_structures_current_type: }
6628       }{%
6629         \stex_annotation:nn{shtml:term=OML,shtml:head={\l__stex_structures_field_name_str}}
6630       }
6631       \exp_args:No \exp_not:n \l__stex_structures_more_nextsymbol_tl
6632     }
6633     \exp_not:N \use_ii:nn
6634     \prop_item:Nn \l__stex_structures_prop {#1}
6635   }
6636 }{
6637   \msg_error:nnn{\stex}{error/unknownfield}{#1}
6638 }
6639 }
6640
6641 \cs_new_protected:Nn \__stex_structures_make_prop: {
6642   \prop_clear:N \l__stex_structures_prop
6643   \seq_clear:N \l__stex_structures_seq
6644   \seq_clear:N \l__stex_structures_assigned_seq

```

```

6645 \tl_clear:N \l__stex_structures_redo_tl
6646 \__stex_structures_prop_do_decls:
6647 \__stex_structures_prop_do_notations:
6648 }
6649
6650 \cs_new_protected:Nn \__stex_structures_make_prop_assign: {
6651   \clist_if_empty:NF \l__stex_structures_fields_clist {
6652     \clist_map_inline:Nn \l__stex_structures_fields_clist {
6653       \__stex_structures_make_prop_assign:nn ##1
6654     }
6655   }
6656 }
6657
6658 \cs_new_protected:Nn \__stex_structures_make_prop_assign:nn {
6659   \prop_if_in:NnTF \l__stex_structures_prop {#1} {
6660     \exp_args:NNe \seq_put_right:Nn \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}
6661     \exp_args:Nne \use:nn {\__stex_structures_make_prop_assign_replace:nnnn {#1}{#2}}
6662     {\prop_item:Nn \l__stex_structures_prop {#1}}
6663   }{
6664     \msg_error:nnn{stex}{error/unknownfieldass}{#1}
6665   }
6666 }
6667 \cs_new_protected:Nn \__stex_structures_make_prop_assign_replace:nnnn {
6668   \prop_put:Nnn \l__stex_structures_prop {#1}{##3}{#2}
6669   \tl_if_empty:nF{#3} {
6670     \tl_set:cn{#1}{ #2 }
6671     \tl_put_right:Nn \l__stex_structures_redo_tl {
6672       \tl_set:cn{#1}{ #2 }
6673     }
6674   }
6675 }
6676
6677 \cs_new_protected:Nn \__stex_structures_prop_do_decls: {
6678   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_type_tl {
6679     \tl_if_empty:nTF{##2} {
6680       \__stex_structures_do_decl_nomacro:nnnnnnnn{##3}
6681     }{
6682       \__stex_structures_do_decl:nnnnnnnn{##2}
6683     }
6684     {##1}{##3}{##4}{##5}{##6}{##7}{##8}{##9}
6685   }
6686 }
6687
6688 \cs_new_protected:Nn \__stex_structures_do_decl_nomacro:nnnnnnnn {
6689   \prop_if_in:NnF \l__stex_structures_prop {#1} {
6690     \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
6691     \prop_put:Nnn \l__stex_structures_prop {#1} {
6692       {}{
6693         \stex_invoke_symbol:nnnnnnN
6694         {#2}
6695         {#3}
6696         {#4}{#5}{#6}{#7}{#8}{#9}
6697       }
6698     }

```

```

6699     }
6700 }
6701
6702 \cs_new_protected:Nn \__stex_structures_do_decl:nnnnnnnn {
6703   \prop_if_in:NnF \l__stex_structures_prop {#1} {
6704     \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2##3}}
6705     \prop_put:Nnn \l__stex_structures_prop {#1}{#3} {
6706       \stex_invoke_symbol:nnnnnnnN
6707       {#2}
6708       {#3}
6709       {#4}{#5}{#6}{#7}{#8}#9
6710     }
6711   }
6712 }
6713 }
6714 \%tl_set:cn{#1}{%
6715   \stex_invoke_symbol:nnnnnnnN
6716   {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
6717 }
6718 \%tl_put_right:Nn \l__stex_structures_redo_tl {
6719   \tl_set:cn{#1}{%
6720     \stex_invoke_symbol:nnnnnnnN
6721     {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
6722   }
6723 }
6724 }
6725
6726 \cs_new_protected:Nn \__stex_structures_prop_do_notations: {
6727   \exp_args:No \stex_iterate_notations:nn\l_stex_current_type_tl{
6728     \exp_args:NNe \seq_if_in:NnT \l__stex_structures_seq {\tl_to_str:n{##1}}{
6729       \tl_put_right:Nn \l__stex_structures_redo_tl {
6730         \cs_if_exist:cF{\l_stex_notation_##1 _##2_cs} {
6731           \tl_set:cn{\l_stex_notation_##1 _##2_cs}{##4}
6732         }
6733         \cs_if_exist:cF{\l_stex_notation_##1 __cs} {
6734           \tl_set:cn{\l_stex_notation_##1 __cs}{##4}
6735         }
6736       }
6737       \cs_if_exist:cF{\l_stex_notation_##1 _##2_cs} {
6738         \tl_set:cn{\l_stex_notation_##1 _##2_cs}{##4}
6739       }
6740       \cs_if_exist:cF{\l_stex_notation_##1 __cs} {
6741         \tl_set:cn{\l_stex_notation_##1 __cs}{##4}
6742       }
6743       \tl_if_empty:nF{##5} {
6744         \tl_put_right:Nn \l__stex_structures_redo_tl {
6745           \cs_if_exist:cF{\l_stex_notation_##1 _op_##2_cs} {
6746             \tl_set:cn{\l_stex_notation_##1 _op_##2_cs}{##5}
6747           }
6748           \cs_if_exist:cF{\l_stex_notation_##1 _op__cs} {
6749             \tl_set:cn{\l_stex_notation_##1 _op__cs}{##5}
6750           }
6751         }
6752       \cs_if_exist:cF{\l_stex_notation_##1 _op_##2_cs} {

```

```

6753           \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
6754       }
6755   \cs_if_exist:cF{l_stex_notation_##1 _op__cs}{
6756     \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
6757   }
6758 }
6759 }
6760 }
6761 }

\usestructure
6762 \cs_new_protected:Npn \usestructure #1 {
6763   \stex_get_mathstructure:n{#1}
6764   \seq_clear:N \l_stex_structures_imports_seq
6765   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6766     \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6767       \seq_put_right:Nn \l_stex_structures_imports_seq{##1}
6768     }
6769   }
6770   \seq_map_inline:Nn \l_stex_structures_imports_seq {
6771     \stex_if_do_html:T {
6772       \hbox{\stex_annotation_invisible:nn
6773         {shtml:usemodule=##1} {}}
6774     }
6775     \stex_activate_module:n {##1}
6776   }
6777 }

```

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

13.10 Statements

```

6778 <@=stex_statements>
6779
6780 \stex_keys_define:nnnn{statement}{
6781   \str_clear:N \l_stex_key_name_str
6782   \str_clear:N \l_stex_key_mroname_str
6783   \clist_clear:N \l_stex_key_for_clist
6784   \str_clear:N \l_stex_key_args_str
6785   \tl_clear:N \l_stex_key_type_tl
6786   \tl_clear:N \l_stex_key_def_tl
6787   \tl_clear:N \l_stex_key_return_tl
6788   \clist_clear:N \l_stex_key_argtypes_clist
6789 }{
6790   name .str_set:N = \l_stex_key_name_str ,
6791   for .clist_set:N = \l_stex_key_for_clist ,
6792   macro .str_set:N = \l_stex_key_mroname_str ,
6793   % start .str_set:N = \l_stex_key_title_str , % TODO remove
6794   type .tl_set:N = \l_stex_key_type_tl ,
6795   judgment .code:n = {},
6796   from .code:n= {}, % TODO remove
6797   to .code:n={} % TODO remove
6798 }{id,title,style,symargs}

```

```

\stex_new_statement:nn
6799 \cs_new_protected:Npn \stex_do_for_list: {
6800   \seq_clear:N \l_stex_fors_seq
6801   \clist_map_inline:Nn \l_stex_key_for_clist {
6802     \exp_args:N\stex_get_symbol:n{\tl_to_str:n{##1}}
6803     \seq_put_right:Nx \l_stex_fors_seq
6804       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6805   }
6806 }
6807
6808 \cs_new_protected:Nn \__stex_statements_setup:nn {
6809   \str_if_empty:NF \l_stex_key_macroname_str {
6810     \str_if_empty:NT \l_stex_key_name_str {
6811       \str_set_eq:NN \l_stex_key_name_str \l_stex_key_macroname_str
6812     }
6813   }
6814   \stex_do_for_list:
6815   \str_if_empty:NF \l_stex_key_name_str {
6816     \__stex_statements_force_id:
6817     \seq_put_right:Nx \l_stex_fors_seq {
6818       \l_stex_current_module_str ? \l_stex_key_name_str
6819     }
6820     \str_set_eq:NN \l_stex_macroname_str \l_stex_key_macroname_str
6821     \str_set:Nn \l_stex_key_role_str {#2}
6822     \stex_symdecl_do:
6823     \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnN} {
6824       {\l_stex_key_macroname_str}{\l_stex_key_name_str}
6825       {\int_use:N \l_stex_get_symbol_arity_int}
6826       {\l_stex_get_symbol_args_tl}
6827       {#1}{}{}\stex_invoke_symbol:
6828     }
6829     \stex_if_do_html:T \__stex_symdecl_html:
6830   }
6831   \str_clear:N \l__stex_statements_uri_str
6832   \str_if_empty:NTF \l_stex_key_name_str {
6833     \stex_debug:nn{statement}{no~name}
6834     \int_compare:nNnTF {\seq_count:N \l_stex_fors_seq} = 1 {
6835       \str_set:Nx \l__stex_statements_uri_str {\seq_item:Nn \l_stex_fors_seq 1}
6836       \stex_debug:nn{statement}{for:~\l__stex_statements_uri_str}
6837     }{
6838       \stex_debug:nn{statement}{no~for}
6839     }
6840   }{
6841     \str_set:Nx \l__stex_statements_uri_str {\l_stex_current_module_str ? \l_stex_key_name_s
6842     \stex_debug:nn{statement}{name:~\l__stex_statements_uri_str}
6843   }
6844 }
6845
6846 \cs_new:Nn \__stex_statements_html_keyvals:nn {
6847   shtml:#1={},
6848   shtml:inline={#2},
6849   \seq_if_empty:NF \l_stex_fors_seq {
6850     shtml:fors={\seq_use:Nn \l_stex_fors_seq ,}
6851   }

```

```

6852 \str_if_empty:NF \l_stex_key_id_str {
6853     shtml:id={\stex_uri_use:N \l_stex_current_doc_uri ? \l_stex_key_id_str}
6854 }
6855 \clist_if_empty:NF \l_stex_key_style_clist {
6856     shtml:styles={\l_stex_key_style_clist}
6857 }
6858 }
6859
6860 \cs_new_protected:Nn \stex_new_statement:nnn {
6861     \stex_new_stylable_env:nnnnnnn {\#1}{0{} }{
6862         \stex_keys_set:nn{statement}{##1}
6863         #3
6864
6865         \stex_if_smsmode:F {
6866             \exp_args:Nne \begin{stex_annotation_env} {
6867                 \__stex_statements_html_keyvals:nn{\#1}{false}
6868             }
6869             \tl_set_eq:NN \thistitle \l_stex_key_title_tl
6870             \str_set_eq:NN \thisname \l_stex_key_name_str
6871             \clist_set_eq:NN \thisfor \l_stex_key_for_str
6872             \stex_if_html_backend:TF {
6873                 \noindent
6874                 \stex_annotation:nn{shtml:statementtitle={}}{\stex_annotation_force_break:n\l_stex_key_}
6875             }
6876             \stex_style_apply:
6877         }
6878         \stex_do_id:
6879         \stex_smsmode_do:
6880     }{
6881         \stex_if_smsmode:F {
6882             \stex_if_html_backend:F \stex_style_apply:
6883             \end{stex_annotation_env}
6884         }
6885     }{}{}{s}
6886     \stex_sms_allow_env:n{s#1}
6887
6888     \tl_if_empty:nF{#2} {
6889         \exp_after:wN \NewDocumentCommand \cs:w inline#2\cs_end: { 0{} m} {
6890             \group_begin:
6891                 \stex_keys_set:nn{statement}{##1}
6892                 #3
6893                 \stex_do_id:
6894                 \stex_if_smsmode:F{
6895                     \exp_args:Ne \stex_annotation:nn{\__stex_statements_html_keyvals:nn{\#1}{true}}{
6896                         \stex_annotation_force_break:n{##2}
6897                     }
6898                 }
6899                 \group_end:
6900                 \stex_smsmode_do:
6901             }
6902             \exp_after:wN \stex_sms_allow_escape:N\cs:w inline#2\cs_end:
6903         }
6904     }
6905

```

```

6906 \cs_new_protected:Nn \__stex_statements_setup_def: {
6907   \stex_if_smsmode:F{
6908     \seq_map_inline:Nn \l_stex_fors_seq {
6909       \stex_ref_new_sym_target:n{##1}
6910     }
6911   }
6912   \stex_reactivate_macro:N \definiendum
6913   \stex_reactivate_macro:N \defnotation
6914   \stex_reactivate_macro:N \defname
6915   \stex_reactivate_macro:N \Defname
6916   \stex_reactivate_macro:N \varbind
6917 }
6918
6919 \cs_new_protected:Nn \__stex_statements_force_id: {
6920   \str_if_empty:NT \l_stex_key_id_str {
6921     \stex_ref_new_id:n{}
6922     \str_set_eq:NN \l_stex_key_id_str \l__stex_refs_str
6923   }
6924 }
6925
6926 \stex_new_statement:nnn{definition}{def} {
6927   \__stex_statements_force_id:
6928   \__stex_statements_setup:nn{}{}
6929   \__stex_statements_setup_def:
6930   \stex_reactivate_macro:N \definiens
6931 }
6932 \stex_new_statement:nnn{assertion}{ass} {
6933   \__stex_statements_setup:nn{}{assertion}
6934   \stex_if_smsmode:F{
6935     \seq_map_inline:Nn \l_stex_fors_seq {
6936       \stex_ref_new_sym_target:n{##1}
6937     }
6938   }
6939   \stex_reactivate_macro:N \varbind
6940   \stex_reactivate_macro:N \conclusion
6941   \stex_reactivate_macro:N \premise
6942   \stex_reactivate_macro:N \definiendum
6943   \stex_reactivate_macro:N \defnotation
6944   \stex_reactivate_macro:N \defname
6945   \stex_reactivate_macro:N \Defname
6946 }
6947 \stex_new_statement:nnn{example}{ex} {\__stex_statements_setup:nn{}{example}}
6948 \stex_new_statement:nnn{paragraph}{} {
6949   \clist_if_in:NnTF \l_stex_key_style_clist {symdoc} {
6950     \__stex_statements_force_id:
6951     \__stex_statements_setup:nn{}{}
6952     \__stex_statements_setup_def:
6953   }{
6954     \__stex_statements_setup:nn{}{}
6955   }
6956 }

```

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

definiendum

```
6957 \cs_new_protected:Nn \__stex_statements_do_deref:nn {
6958   \stex_if_html_backend:T{\ifvmode\indent\fi}
6959   \group_begin:
6960   \stex_get_symbol:n{#1}
6961   \bool_if:NTF \l_stex_allow_semantic_bool{
6962     \str_set:Nx \l_stex_current_symbol_str
6963     {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6964     \str_if_in:NnT \l_stex_get_symbol_name_str / {
6965       \str_set:Nx \l_stex_get_symbol_name_str {
6966         \exp_after:wN \_stex_split_slash: \l_stex_get_symbol_name_str
6967         /\_stex_args_end:
6968       }
6969     }
6970     \exp_args:No \stex_ref_new_sym_target:n \l_stex_current_symbol_str
6971     \def\comp{\_defcomp}
6972     \stex_annotation:nn{shtml:definiendum=\l_stex_current_symbol_str}{\comp{#2}}
6973   }{
6974     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
6975   }
6976   \group_end:
6977 }
6978
6979 \NewDocumentCommand \defnotation{ m } {
6980   \l_stex_next_symbol:n { \def\comp{\_defcomp}}#1
6981 }
6982 \stex_deactivate_macro:Nn \defnotation {definition~environments}
6983
6984 \NewDocumentCommand \definiendum { O{} m m} {
6985   \stex_keys_set:nn{symname}{ #1 }
6986   \__stex_statements_do_deref:nn{#2}{#3}
6987 }
6988 \stex_deactivate_macro:Nn \definiendum {definition~environments}
6989
6990 \NewDocumentCommand \defname { O{} m } {
6991   \stex_keys_set:nn{symname}{#1}
6992   \__stex_statements_do_deref:nn{#2}(
6993     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
6994   )
6995 }
6996 \stex_deactivate_macro:Nn \defname {definition~environments}
6997
6998 \NewDocumentCommand \Defname { O{} m } {
6999   \stex_keys_set:nn{symname}{#1}
7000   \__stex_statements_do_deref:nn{#2}(
7001     \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
7002   )
7003 }
7004 \stex_deactivate_macro:Nn \Defname {definition~environments}
7005
7006
7007 \NewDocumentCommand \defniens { O{} m }{
7008   \group_begin:
7009   \str_clear:N \l_stex_get_symbol_name_str
```

```

7010 \tl_if_empty:nF {#1} {
7011   \stex_get_symbol:n { #1 }
7012   \str_set:Nx \l__stex_statements_uri_str
7013     {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7014 }
7015 \str_if_empty:NT \l__stex_statements_uri_str {
7016   \msg_error:nn{stex}{error/definiensfor}
7017 }
7018 \stex_debug:nn{definiens}{Checking-\l__stex_statements_uri_str}
7019
7020 \exp_args:No \_stex_add_definiens:nn \l__stex_statements_uri_str{#2}
7021
7022 \group_end:
7023 \stex_smsmode_do:
7024 }
7025 \stex_deactivate_macro:Nn \definiens {definition-environments}
7026 \stex_sms_allow_escape:N \definiens
7027
7028 \cs_new_protected:Nn \_stex_add_definiens:nn {
7029   \exp_args:Nno \stex_str_if_starts_with:nnT{#1} \l_stex_current_module_str {
7030     \prop_map_inline:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop} {
7031       \stex_debug:nn{definiens}{#1 == \l_stex_current_module_str?##1}
7032       \str_if_eq:not {#1} {\l_stex_current_module_str?##1} {
7033         \prop_map_break:n{\_stex_add_definiens_inner:nnnnnnnn ##2}
7034       }
7035     }
7036   }
7037   \stex_if_smsmode:F{
7038     \stex_annotation:nn{ shtml:definiens={#1}}{
7039       #2 \%_stex_annotation_force_break:n{ #2 }
7040     }
7041   }
7042 }
7043
7044 \cs_new_protected:Nn \_stex_add_definiens_inner:nnnnnnnn {
7045   \stex_debug:nn{definiens}{Adding-definiens-to-\l_stex_current_module_str?#2}
7046   \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
7047     {#2}{##1}{#2}{#3}{#4}{defed}{#6}{#7}{#8}}
7048 }
7049
7050 \NewDocumentCommand \varbind {m} {
7051   \clist_map_inline:nn {#1} {
7052     \stex_get_var:n {##1}
7053     \stex_if_do_html:T {
7054       \stex_annotation_invisible:nn {shtml:bind=\l_stex_get_symbol_name_str}{}
7055     }
7056   }
7057 }
7058 \stex_deactivate_macro:Nn \varbind {definition-or-assertion-environments}
7059
7060 \NewDocumentCommand \conclusion { O{} m } {
7061   \group_begin:
7062   \str_clear:N \l_stex_get_symbol_name_str
7063   \tl_if_empty:nF {#1} {

```

```

7064     \stex_get_symbol:n { #1 }
7065     \str_set:Nx \l__stex_statements_uri_str
7066         {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7067     }
7068     \str_if_empty:NT \l__stex_statements_uri_str {
7069         \msg_error:nn{stex}{error/conclusionfor}
7070     }
7071     \stex_annotate:nn{ shtml:conclusion=\l__stex_statements_uri_str}{
7072         #2 \%_stex_annotate_force_break:n{ #2 }
7073     }
7074     \group_end:
7075 }
7076 \stex_deactivate_macro:Nn \conclusion {assertion-environments}
7077
7078 \NewDocumentCommand \premise {O{} m} {
7079     \tl_if_empty:nF {#1} {
7080         \stex_debug:nn{Here:}{Variable~#1}
7081         \exp_args:NNx\exp_args:Nnx\vardef{v#1}{#1}
7082     }
7083     \stex_annotate:nn{shtml:premise={#1}}{#2}
7084 }
7085 \stex_deactivate_macro:Nn \premise {assertion-environments}

```

(End of definition for `definiendum`. This function is documented on page 90.)

13.11 Proofs

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

```

7086 <@=stex_proof>
7087 \stex_keys_define:nnnn{ spf }{
7088 \tl_clear:N \l_stex_key_for_clist
7089 \tl_clear:N \l_stex_key_from_tl
7090 \tl_set_eq:NN \l_stex_key_proofend_tl \__stex_proof_proof_box_tl
7091 \tl_clear:N \l_stex_key_continues_tl
7092 \tl_clear:N \l_stex_key_term_tl
7093 \tl_clear:N \l_stex_key_functions_tl
7094 \tl_clear:N \l_stex_key_method_tl
7095 \bool_set_false:N \l_stex_key_hide_bool
7096 }{
7097     for      .clist_set:N = \l_stex_key_for_clist ,
7098     from     .tl_set:N   = \l_stex_key_from_tl ,
7099     proofend .tl_set:N   = \l_stex_key_proofend_tl,
7100     continues .tl_set:N = \l_stex_key_continues_tl,
7101     functions .tl_set:N = \l_stex_key_functions_tl,
7102     term     .tl_set:N   = \l_stex_key_term_tl,
7103     method    .tl_set:N = \l_stex_key_method_tl,
7104     hide      .bool_set:N = \l_stex_key_hide_bool
7105 }{id,style,title}
7106
7107 \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^AT_EX only allows `enumerate` environments up to nesting depth 4 and general list environments up to listing depth 6. This is not enough for us. Therefore

we have decided to go along the route proposed by Leslie Lamport to use a single top-level list with dotted sequences of numbers to identify the position in the proof tree. Unfortunately, we could not use his `pf.sty` package directly, since it does not do automatic numbering, and we have to add keyword arguments all over the place, to accomodate semantic information.

```

7108 \intarray_new:Nn\l__stex_proof_counter_intarray{50}
7109 \cs_new_protected:Npn \__stex_proof_insert_number: {
7110   \int_set:Nn \l_tmpa_int {1}
7111   \bool_while_do:nn {
7112     \int_compare_p:nNn {
7113       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7114     } > 0
7115   }{
7116     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .
7117     \int_incr:N \l_tmpa_int
7118   }
7119 }
7120 \cs_new_protected:Nn \__stex_proof_number_as_string:N {
7121   \str_clear:N #1
7122   \int_set:Nn \l_tmpa_int {1}
7123   \bool_while_do:nn {
7124     \int_compare_p:nNn {
7125       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7126     } > 0
7127   }{
7128     \str_put_right:Nx #1 {\intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .}
7129     \int_incr:N \l_tmpa_int
7130   }
7131 }
7132
7133 \cs_new_protected:Npn \__stex_proof_inc_counter: {
7134   \int_set:Nn \l_tmpa_int {1}
7135   \bool_while_do:nn {
7136     \int_compare_p:nNn {
7137       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7138     } > 0
7139   }{
7140     \int_incr:N \l_tmpa_int
7141   }
7142   \int_compare:nNnF \l_tmpa_int = 1 {
7143     \int_decr:N \l_tmpa_int
7144   }
7145   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int {
7146     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int + 1
7147   }
7148 }
7149
7150 \cs_new_protected:Npn \__stex_proof_add_counter: {
7151   \int_set:Nn \l_tmpa_int {1}
7152   \bool_while_do:nn {
7153     \int_compare_p:nNn {
7154       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7155     } > 0

```

```

7156 }{
7157   \int_incr:N \l_tmpa_int
7158 }
7159 \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 1 }
7160 }
7161
7162 \cs_new_protected:Npn \__stex_proof_remove_counter: {
7163   \int_set:Nn \l_tmpa_int {1}
7164   \bool_while_do:nn {
7165     \int_compare_p:nNn {
7166       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7167     } > 0
7168   }{
7169     \int_incr:N \l_tmpa_int
7170   }
7171   \int_decr:N \l_tmpa_int
7172   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 0 }
7173 }

```

`spfsketch`

```

7174 \newenvironment{spfsketchenv}{}{}
7175 \stex_new_stylable_cmd:nnnn{spfsketch}{0{}} m}{\par
7176   \begin{spfsketchenv}
7177     \stex_keys_set:nn{spf}{#1}
7178     \stex_do_for_list:
7179     \stex_do_id:
7180     \exp_args:Ne \stex_annotate:nn{
7181       shtml:proofsketch={
7182         \seq_if_empty:NF \l_stex_fors_seq {
7183           \seq_use:Nn \l_stex_fors_seq ,
7184         }
7185       }
7186     }{
7187       \stex_style_apply:
7188       #2
7189     }
7190   \end{spfsketchenv}
7191 }{
7192   \noindent\emph{\spfsketchenv\autorefname :}-
7193 }

```

(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

```

7194 \tl_set:Nn \__stex_proof_proof_box_tl {
7195   \ltx@ifpackageloaded{amssymb}{$\square$}{
7196     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
7197   }
7198 }
7199
7200 \tl_set:Nn \sproofend {
7201   \tl_if_empty:NF \l_stex_key_proofend_tl {
7202     \hfil\null\nobreak\hfill\l_stex_key_proofend_tl\par\smallskip

```

```

7203     }
7204 }

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

\stexcommentfont
7205 \cs_new_protected:Npn \stexcommentfont {
7206   \small\itshape
7207 }

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

\sproof (env.)
7208 \cs_new_protected:Nn \__stex_proof_start_list:n {
7209   \begin{list}{}{%
7210     \setlength\topsep{0pt}
7211     \setlength\parsep{0pt}
7212     \setlength\rightmargin{0pt}
7213   }\item[#1]
7214 }
7215 \cs_new_protected:Nn \__stex_proof_end_list: {
7216   \end{list}
7217 }

7218 \cs_new_protected:Nn \__stex_proof_html: {
7219   \stex_annotation_invisible:n{\hbox{
7220     \tl_if_empty:NF \l_stex_key_term_tl {
7221       $\stex_annotation:nn{shtml:proofterm={}}{\l_stex_key_term_tl}$
7222     }
7223     \tl_if_empty:NF \l_stex_key_method_tl {
7224       \stex_annotation:nn{shtml:proofmethod={}}{\l_stex_key_method_tl}
7225     }
7226   }}
7227 }
7228 }

7229 \cs_new_protected:Nn \__stex_proof_html_env:n {
7230   \exp_args:Nne \begin{stex_annotation_env}{%
7231     shtml:#1={%
7232       \seq_if_empty:NF \l_stex_fors_seq {
7233         \seq_use:Nn \l_stex_fors_seq ,
7234       }
7235     }
7236     \bool_if:NT \l_stex_key_hide_bool {,
7237       shtml:proofhide=true
7238     }
7239   }
7240 }
7241 \__stex_proof_html:
7242 }

7243 \bool_set_false:N \l_stex_proof_in_spfblock_bool
7244 \cs_new_protected:Nn \__stex_proof_begin_proof:nn {\par
7245   \intarray_gzero:N \l_stex_proof_counter_intarray
7246   \intarray_gset:Nnn \l_stex_proof_counter_intarray 1 1
7247   \stex_keys_set:nn{spfsteps}{#1}
7248   \stex_do_for_list:

```

```

7250  \stex_if_do_html:T {
7251    \__stex_proof_html_env:n{proof}
7252  }
7253  \seq_map_inline:Nn \l_stex_fors_seq {
7254    \stex_debug:nn{definiens}{Adding-definiens-to-##1}
7255    \_stex_add_definiens:nn {##1}{\STEXinvisibl{proven}}
7256  }
7257  \stex_style_apply:
7258  \_stex_do_id:
7259  \stex_reactivate_macro:N \subproof
7260  \stex_reactivate_macro:N \spfstep
7261  \stex_reactivate_macro:N \conclude
7262  \stex_reactivate_macro:N \assumption
7263  \stex_reactivate_macro:N \eqstep
7264  \stex_reactivate_macro:N \yield
7265  \stex_reactivate_macro:N \spfblock
7266  \stex_reactivate_macro:N \spfjust
7267  \stex_annotation:nn{html:prooftitle={}}{#2}
7268  \stex_if_do_html:T{
7269    \begin{stex_annotation_env}{html:proofbody={}}
7270  }
7271 }
7272 \stex_new_stylable_env:nnnnnnn{proof}{0{} m}{
7273   \__stex_proof_begin_proof:nn{#1}{#2}
7274   \bool_set_true:N\l_stex_proof_in_spfblock_bool\__stex_proof_start_list:n{}
7275   \group_begin:\stexcommentfont
7276 }{
7277   \stex_style_apply:
7278   \stex_if_do_html:T{\end{stex_annotation_env}\end{stex_annotation_env}}
7279 }{
7280   \emph{\sproofautorefname :}~
7281 }{
7282   \sproofend
7283 }{s}
7284 \AddToHook{env/sproof/end}{
7285   \bool_if:NT\l_stex_proof_in_spfblock_bool {
7286     \group_end:\__stex_proof_end_list:
7287   }
7288 }
7289 \stex_new_stylable_env:nnnnnnn{proof*}{0{}>{
7290   \__stex_proof_begin_proof:nn{#1}{}
7291   \bool_set_false:N\l_stex_proof_in_spfblock_bool
7292 }{
7293   \stex_style_apply:
7294   \stex_if_do_html:T{\end{stex_annotation_env}\end{stex_annotation_env}}
7295 }{
7296   \emph{Proof:}~
7297 }{
7298   \sproofend
7299 }{s}
7300

subproof (env.)
7301 \str_set_eq:NN \subproofautorefname \spfstepautorefname

```

```

7302 \stex_new_stylable_env:nnnnnnn{subproof}{s 0{} m}{\par
7303   \stex_keys_set:nn{spf}{#2}
7304   \_stex_do_for_list:
7305   \stex_if_do_html:T {
7306     \__stex_proof_html_env:n{subproof}
7307   }
7308   \seq_map_inline:Nn \l_stex_fors_seq {
7309     \stex_debug:nn{definiens}{Adding-definiens-to-##1}
7310     \_stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7311   }
7312
7313   \IfBooleanTF #1 {
7314     \stex_style_apply:
7315     \str_if_empty:NF \l_stex_key_id_str {
7316       \__stex_proof_number_as_string:N \@currentlabel
7317       \str_set:Nx \@currentHref{subproof.\@currentlabel}
7318       \_stex_do_id:
7319     }
7320     \bool_set_false:N \l__stex_proof_in_spfblock_bool
7321     \stex_annotation:nn{shtml:prooftitle={}}{#3}
7322   }{
7323     \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7324       \str_if_empty:NF \l_stex_key_id_str {
7325         \__stex_proof_number_as_string:N \@currentlabel
7326         \str_set:Nx \@currentHref{subproof.\@currentlabel}
7327         \_stex_do_id:
7328       }
7329       \__stex_proof_start_list:n\__stex_proof_insert_number:
7330         \stex_annotation:nn{shtml:prooftitle={}}{#3}
7331         \__stex_proof_add_counter:
7332         \stex_style_apply:
7333       }{
7334         \stex_annotation:nn{shtml:prooftitle={}}{#3}
7335         \stex_style_apply:
7336         \_stex_do_id:
7337       }
7338     }
7339     \stex_if_do_html:T{
7340       \begin{stex_annotation_env}{shtml:proofbody={}}
7341     }
7342     \bool_if:NT \l__stex_proof_in_spfblock_bool {\group_begin:\stexcommentfont}
7343   }{
7344     \stex_style_apply:
7345     \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_inc_counter:
7346     \stex_if_do_html:T{\end{stex_annotation_env}}
7347     \bool_if:NT\l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
7348     \stex_if_do_html:T{\end{stex_annotation_env}}
7349     \aftergroup \__stex_proof_inblock_restore:
7350   }{}{}{}
7351 \AddToHook{env/subproof/before}{
7352   \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7353 }
7354 \AddToHook{env/subproof/end}{
7355   \bool_if:NT\l__stex_proof_in_spfblock_bool {

```

```

7356     \group_end:\__stex_proof_remove_counter:
7357     \%__stex_proof_end_list:
7358 }
7359 }
7360 \stex_deactivate_macro:Nn \subproof {sproof~environments}
7361
7362 \cs_new_protected:Nn \__stex_proof_inblock_restore: {
7363     \bool_if:NT\l__stex_proof_in_spfblock_bool {
7364         \group_begin:\stexcommentfont
7365     }
7366 }

\spfstep
\conclude
\assumption
\have
\eqstep
7367
7368 \stex_keys_define:nnnn { spfsteps } {
7369     \clist_clear:N \l__stex_key_for_clist
7370     \str_clear:N \l__stex_key_name_str
7371     \tl_clear:N \l__stex_key_method_tl
7372     \tl_clear:N \l__stex_key_term_tl
7373 }{
7374     for .clist_set:N = \l__stex_key_for_clist ,
7375     method .tl_set:N = \l__stex_key_method_tl,
7376     term .tl_set:N = \l__stex_key_term_tl,
7377     name .str_set_x:N = \l__stex_key_name_str
7378     % todo: style=inline
7379 }{id,style,title}
7380
7381 \newenvironment{spfstepenv}{
7382     \str_set_eq:NN \spfstepenvautorefname \spfstepautorefname
7383 }{}
7384
7385 \cs_new_protected:Nn \__stex_proof_step_html:nn {
7386     \stex_if_do_html:TF{
7387         \exp_args:Ne \stex_annotation:nn{
7388             shtml:spf#1={
7389                 \seq_if_empty:NF \l__stex_fors_seq {
7390                     \seq_use:Nn \l__stex_fors_seq ,
7391                 }
7392             }
7393             \str_if_empty:NF \l__stex_key_name_str {
7394                 shtml:stepname={\l__stex_key_name_str}
7395             }
7396         }{
7397             \__stex_proof_html:
7398             #2
7399         }
7400     }{ #2 }
7401 }
7402
7403 \cs_new_protected:Nn \__stex_proof_make_step_macro:Nnnnn {
7404     \NewDocumentCommand #1 {s O{} +m} {
7405         \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7406         \stex_keys_set:nn{spfsteps}{##2}
7407         \str_if_empty:NF \l__stex_key_name_str {

```

```

7408     \stex_debug:nn{Here:}{Variable-\l_stex_key_name_str}
7409     \exp_args:NNx\exp_args:Nnx\vardef{v\l_stex_key_name_str}{\l_stex_key_name_str}
7410 }
7411
7412 \begin{spfstepenv}
7413   \str_if_empty:NF \l_stex_key_id_str {
7414     \l_stex_proof_number_as_string:N \@currentlabel
7415     \str_set:Nx \@currentHref{spfstep.\@currentlabel}
7416     \l_stex_do_id:
7417   }
7418
7419   \bool_if:NTF \l_stex_proof_in_spfblock_bool {
7420     \IfBooleanTF ##1 {
7421       \l_stex_proof_step_html:nn{##2}{##3}
7422     }{
7423       \l_stex_proof_step_html:nn{##2}{\l_stex_proof_start_list:n{##3} ##3 \l_stex_proof_end_}
7424       #5
7425     }
7426     \end{spfstepenv}
7427     \group_begin:\stexcommentfont
7428   }{
7429     \l_stex_proof_step_html:nn{##2}{##3}
7430     \end{spfstepenv}
7431   }
7432 }
7433 \stex_deactivate_macro:Nn #1 {sproof~environments}
7434 }
7435
7436 \l_stex_proof_make_step_macro:Nnnnn \assumption {assumption} \l_stex_proof_insert_number: {}
7437 \l_stex_proof_make_step_macro:Nnnnn \conclude {conclusion} {${\Rightarrow}{}$} {} {}
7438 \l_stex_proof_make_step_macro:Nnnnn \spfstep {step} \l_stex_proof_insert_number: {} \l_stex_
7439
7440 \NewDocumentCommand \eqstep {s m} {
7441   \bool_if:NTF \l_stex_proof_in_spfblock_bool {
7442     \group_end:
7443     \IfBooleanTF #1 {
7444       \l_stex_proof_step_html:nn{eqstep}{$=##2$}
7445     }{
7446       \l_stex_proof_step_html:nn{eqstep}{\l_stex_proof_start_list:n{$=$} $##2$ \l_stex_proof_}
7447     }
7448     \group_begin:\stexcommentfont
7449   }{
7450     \l_stex_proof_step_html:nn{eqstep}{$=##2$}
7451   }
7452 }
7453 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
7454
7455 \NewDocumentCommand \yield {+m} {
7456   \stex_annotation:nn{shtml:proofterm={}}{#1}
7457 }
7458 \stex_deactivate_macro:Nn \yield {sproof~environments}
7459
7460 \NewDocumentEnvironment{spfblock}{}{
7461   \bool_set_false:N \l_stex_proof_in_spfblock_bool

```

```

7462 }{
7463   \aftergroup\__stex_proof_inblock_restore:
7464 }
7465 \stex_deactivate_macro:Nn \spfblock {sproof~environments}
7466 \AddToHook{env/spfblock/before} {
7467   \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7468 }
7469
7470
7471 \newcommand\spfjust[1]{
7472   \stex_annotate:nn{\spfjust={}}{ #1 }
7473 }
7474 \stex_deactivate_macro:Nn \spfjust {sproof~environments}

```

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

13.12 Metatheory

```

7475 <@=stex_meta>
7476 \group_begin:
7477   \cs_set:Npn \__stex_modules_persist_module: {}
7478   \cs_set:Npn \stex_check_term:n #1 {}
7479   \cs_set:Npn \__stex_sref_do_aux:n #1 { #1 }
7480   \bool_set_false:N \stex_html_do_output_bool
7481   \bool_set_false:N \c_stex_check_terms_bool
7482   \stex_uri_resolve:Nn \l_stex_current_ns_uri {http://mathhub.info/sTeX/meta}
7483   \stex_module_setup:n{Metatheory}
7484
7485 \symdef{of~type}[args=ii,invisible]{#1}
7486 \notation{of~type}[colon]{#1 \mathbin{\comp{:}}}{#2}
7487
7488 \symdef{apply}[args=ia,prec=0;\infprec x\infprec]{#1\mathopen{\comp{}}}{#2 \mathclose{\comp{}}}
7489 \notation{apply}[lambda]{#1\mathopen{\comp{}}; \argsep{#2}\mathclose{\comp{}}}
7490 \notation{apply}[infixop]{\argsep{#2}\mathbin{#1}}
7491 \notation{apply}[infixrel]{\argsep{#2}\mathrel{#1}}
7492
7493 % structures
7494 \symdef{module~type}[args=i,op=\mathtt{MOD}]{\mathopen{\comp{\mathtt{MOD}}}{#1}\mathclose{\comp{}}}
7495 \symdef{module~type~merge}[args=a,op=\oplus]{\argsep{#1}\mathbin{\oplus}}
7496 \symdef{anonymous~record}[args=a]{\mathopen{\comp{[]}}{#1}\mathclose{\comp{[]}}}
7497 \symdef{record~field}[args=2]{#1\mathbin{.}{#2}}
7498 \symdecl*{record~type}
7499
7500 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
7501 \notation{mathstruct}[angle,prec=nobrackets]{\mathopen{\comp{\langle}}}{#1 \mathclose{\comp{\rangle}}}
7502 \notation{mathstruct}[parens,prec=nobrackets]{\mathopen{\comp{()}}}{#1 \mathclose{\comp{()}}}
7503
7504 % sequences
7505 \symdef{ellipses}[ldots]{\ldots}

```

```

7511 \symdef{sequence~expression}[comma,args=a]{#1}
7512 \symdef{sequence~type}[args=1]{#1^{\comp\ast}}
7513 \symdef{sequence~map}[args=ia]{
7514   \comp{\mathrm{map}}{\mathopen{\comp{()#1\mathpunct{\comp{,}}}}
7515   #2\mathclose{\comp{})}}
7516 }
7517 \iffalse
7518 % binder (\forall, \Pi, \lambda etc.)
7519 \symdef{pibind}[name=dependent function type,prec=nobrackets,
7520   op=(\cdot; \to; \cdot, args=Bi,assoc=pre]
7521   {\argmap{#1}{%
7522     \mathopen{\comp{} ##1 \mathclose{\comp{}}}
7523     }{\mathbin{\comp{\to}}} \mathbin{\comp{\to}} #2}
7524 \notation{pibind}[\forall]{\comp\forall #1\mathpunct{\comp.} #2}
7525 \notation{pibind}[\Pi]{\mathop{\comp\prod}\c_math_subscript_token{#1}#2}
7526
7527 \symdef{mapbind}[name=lambda, mapsto, prec=nobrackets, op=\mapsto, args=Bi,assoc=pre]
7528   { #1 \mathrel{\comp\mapsto} #2}
7529 \notation{mapbind}[\lambda,prec=nobrackets,op=\lambda]
7530   {\comp\lambda #1 \mathpunct{\comp.} #2}
7531 \fi
7532 \symdecl{bind}[args=Bi,assoc=pre]
7533 \notation{bind}[deffun,prec=nobrackets,op=(\cdot; \to; \cdot
7534   {\mathopen{\comp{} #1 \mathclose{\comp{}}}\mathbin{\comp{\to}}} #2}
7535 \notation{bind}[\forall]{\comp\forall #1.\;#2}
7536 \notation{bind}[\Pi]{\mathop{\comp\prod}\c_math_subscript_token{#1}#2}
7537
7538 \symdef{implicit~bind}[args=Bi,assoc=pre]{\mathopen{\comp{} #1 \mathclose{\comp{\}}}\c_math_
7539
7540 \symdecl*{integer~literal}
7541 \notation{integer~literal}{\mathbb Z}
7542
7543 \symdecl*{ordinal}
7544 \notation{ordinal}{\mathit{Ord}}
7545
7546 % propositions
7547 \symdef{prop}[name=proposition]{\mathit{Prop}}
7548 \symdef{judgment~holds}[args=i,role=judgment]{\comp\vdash;#1}
7549
7550 % any object
7551 \symdef{object}{\mathit{Obj}}
7552
7553 % TODO DELETE
7554 \symdef{aseqdots}[args=a,prec=nobrackets]
7555   {#1\comp{,\ldots}}%{##1\comp,##2}
7556 \symdef{aseqfromto}[args=ai,prec=nobrackets]
7557   {#1\comp{,\ldots},#2}%{##1\comp,##2}
7558 \symdef{aseqfromtovia}[args=aai,prec=nobrackets]
7559   {#1\comp{,\ldots},#2\comp{,\ldots},#3}%{##1\comp,##2}
7560
7561
7562 \stex_close_module:
7563 \stex_uri_add_module:NNn \l_stex_metatheory_uri \l_stex_current_ns_uri {Metatheory}
7564 \global \let \l_stex_metatheory_uri \l_stex_metatheory_uri

```

```

7565   \global \let \c_stex_default_metatheory \l_stex_metatheory_uri
7566 \group_end:
```

13.13 MMT Interfaces

```

7567 <@=todo>
7568 \cs_new_protected:Npn \MSC #1 {}

\MMTinclude

7569 \stex_new_stylable_cmd:nnnn{MMTinclude}{m}{
7570   \stex_annotation_invisible:nn{shtml:import={#1}}{}}
7571 }{}
7572 \stex_deactivate_macro:Nn \MMTinclude {module~environments}
7573 \stex_every_module:n {\stex_reactivate_macro:N \MMTinclude}
```

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

```

\MMTrule

7574 \NewDocumentCommand \MMTrule {m m} {
7575   \tl_if_empty:nTF{#2}{\seq_clear:N \l_tmpa_seq}{%
7576     \seq_set_split:Nnn \l_tmpa_seq , {#2}}
7577   }
7578   \int_zero:N \l_tmpa_int
7579   \stex_annotation_invisible:n{
7580     $
7581     \stex_annotation:nn{shtml:rule={scala://#1}}{
7582       \stex_annotation_force_break:n{
7583         \seq_if_empty:NF \l_tmpa_seq {
7584           \seq_map_inline:Nn \l_tmpa_seq {
7585             \int_incr:N \l_tmpa_int
7586             \stex_annotation:nn{
7587               shtml:argmode=i,
7588               shtml:arg={\int_use:N \l_tmpa_int}
7589             }{ ##1 }
7590           }
7591         }
7592       }
7593     }$}
7594   }
7595 }

7596 \stex_deactivate_macro:Nn \MMTrule {module~environments}
7597 \stex_every_module:n{\stex_reactivate_macro:N \MMTrule}
```

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

`mmtinterface` (env.)

```

7598 \NewDocumentEnvironment { mmtinterface } { O{} m m } {
7599   \stex_module_setup_top_nosig:n { #3 }
7600   \str_set_eq:NN \l__todo_mmt_module_str \l_stex_current_module_str
7601   \str_clear:N \l_stex_current_module_str
7602   \stex_keys_set:nn { smodule }{ #1 }
7603   \stex_module_setup:n{ #2 }
7604   \str_set_eq:NN \l__todo_stex_module_str \l_stex_current_module_str
7605   \stex_debug:nn{mmt}{Interface~\l__todo_stex_module_str^~Jfor~\l__todo_mmt_module_str}
```

```

7607 \stex_if_do_html:T {
7608   \exp_args:Nne \begin{stex_annotation_env} {
7609     shtml:theory={\l_stex_current_module_str},
7610     shtml:language={ \l_stex_current_language_str},
7611     shtml:signature={}
7612     \tl_if_empty:NF \l_stex_metatheory_uri ,,
7613       shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
7614   }
7615 }
7616 \stex_annotation_invisible:n{}
7617 \stex_annotation_invisible:nn
7618   {shtml:import=\l_todo_mmt_module_str} {}
7619 }
7620 \stex_module_add_code:x{
7621   \stex_activate_module:n{ \l_todo_mmt_module_str }
7622 }
7623 \stex_module_add_morphism:nonn
7624   {}{\l_todo_mmt_module_str}{import}{}
7625 \stex_reactivate_macro:N \mmtdef
7626 \stex_smsmode_do:
7627 }{
7628   \str_set_eq:NN \l_stex_current_module_str \l_todo_mmt_module_str
7629   \stex_close_module:
7630   \str_set_eq:NN \l_stex_current_module_str \l_todo_stex_module_str
7631   \stex_close_module:
7632   \stex_if_do_html:T { \end{stex_annotation_env} }
7633 }
7634 \stex_sms_allow_env:n{mmtinterface}

\mmtdef

7635 \NewDocumentCommand \mmtdef {m O{} m} {
7636   \stex_keys_set:nn{symdef}{#2}
7637   \str_set:Nx \l_stex_macroname_str { #1 }
7638   \str_if_empty:NT \l_stex_key_name_str {
7639     \str_set:Nx \l_stex_key_name_str { #1 }
7640   }
7641   \stex_symdecl_do:

7642
7643   \str_set_eq:NN \l_stex_current_module_str \l_todo_mmt_module_str
7644   \cs_set_eq:NN \l_todo_old_metagroup_cd \stex_metagroup_do_in:nn
7645   \cs_set_protected:Npn \stex_metagroup_do_in:nn ##1 ##2 {##2}
7646   \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN} {
7647     {\l_stex_macroname_str}
7648     {\l_stex_key_name_str}
7649     {\int_use:N \l_stex_get_symbol_arity_int}
7650     {\l_stex_get_symbol_args_t1}
7651     {}
7652     {}
7653     {}
7654     \stex_invoke_symbol:
7655   }
7656   \cs_set_eq:NN \stex_metagroup_do_in:nn \l_todo_old_metagroup_cd
7657   \str_set_eq:NN \l_stex_current_module_str \l_todo_stex_module_str
7658

```

```

7659 \str_set_eq:NN \l_stex_get_symbol_mod_str \l__todo_mmt_module_str
7660 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
7661 \stex_notation_parse:n{#3}
7662 \_stex_notation_check:
7663 \_stex_notation_add:
7664 \stex_if_do_html:T{
7665   \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7666 }
7667 \stex_smsmode_do:
7668 }
7669 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
7670 \stex_sms_allow_escape:N \mmtdef

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

7671 \seq_if_empty:NT \g_stex_current_file {
7672   \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
7673 }
7674 \_stex_persist_read_now:
7675 \_stex_every_file:
7676 \cs_new_protected:Nn \__todo_newlabel:n {
7677   \exp_args:Ne\__todo_old_newlabel:{\tl_to_str:n{#1}}
7678 }
7679 \AtBeginDocument{
7680   \iow_now:Nn \@auxout {
7681     \ExplSyntaxOn
7682     \let\__todo_old_newlabel:\newlabel
7683     \let\newlabel\__todo_newlabel:n
7684     \ExplSyntaxOff
7685   }
7686 }
7687 
```

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.

```
7688 <*cls>
7689 <@=notesslides>
7690 \ProvidesExplClass{notesslides}{2023/03/19}{3.3.0}{notesslides Class}
7691 \RequirePackage{13keys2e}
7692
7693 \str_const:Nn \c__notesslides_class_str {article}
7694
7695 \keys_define:nn{notesslides / cls}{
7696   class .str_set_x:N = \c__notesslides_class_str,
7697   notes .bool_set:N = \c__notesslides_notes_bool ,
7698   slides .code:n    = { \bool_set_false:N \c__notesslides_notes_bool },
7699   %docopt .str_set_x:N = \c__notesslides_docopt_str,
7700   unknown .code:n   = {
7701     \PassOptionsToClass{\CurrentOption}{beamer}
7702     \PassOptionsToClass{\CurrentOption}{\c__notesslides_class_str}
7703     \PassOptionsToPackage{\CurrentOption}{notesslides}
7704     \PassOptionsToPackage{\CurrentOption}{stex}
7705   }
7706 }
7707 \ProcessKeysOptions{ notesslides / cls }
7708
7709 \RequirePackage{stex}
7710 \stex_if_html_backend:T {
7711   \bool_set_true:N \c__notesslides_notes_bool
7712 }
7713
7714 \bool_if:NTF \c__notesslides_notes_bool {
7715   \PassOptionsToPackage{notes=true}{notesslides}
7716   \message{notesslides.cls:-Formatting-document-in-notes-mode}
```

```

7717 }{
7718   \PassOptionsToPackage{notes=false}{notesslides}
7719   \message{notesslides.cls:~Formatting~document~in~slides~mode}
7720 }
7721
7722 \bool_if:NTF \c_notesslides_notes_bool {
7723   \LoadClass{\c__notesslides_class_str}
7724 }{
7725   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7726   \%newcounter{Item}
7727   \%newcounter{paragraph}
7728   \%newcounter{ subparagraph}
7729   \%newcounter{Hfootnote}
7730 }
7731 \RequirePackage{notesslides}
7732 
```

now we do the same for the `notesslides` package.

```

7733 <*package>
7734 \ProvidesExplPackage{notesslides}{2023/03/19}{3.3.0}{notesslides Package}
7735 \RequirePackage{l3keys2e}
7736
7737 \keys_define:nn{notesslides / pkg}{
7738   notes          .bool_set:N  = \c_notesslides_notes_bool ,
7739   slides         .code:n    = { \bool_set_false:N \c_notesslides_notes_bool },
7740   sectocframes  .bool_set:N  = \c_notesslides_sectocframes_bool ,
7741   topsect        .str_set_x:N = \c_notesslides_topsect_str,
7742   unknown        .code:n    = {
7743     \PassOptionsToPackage{\CurrentOption}{stex}
7744     \PassOptionsToPackage{\CurrentOption}{tikzinput}
7745   }
7746 }
7747 \ProcessKeysOptions{ notesslides / pkg }
7748
7749 \RequirePackage{stex}
7750 \stex_if_html_backend:T {
7751   \bool_set_true:N\c_notesslides_notes_bool
7752 }
7753
7754 \cs_set:Npn \sectiontitleemph #1 {
7755   \textbf{\Large #1}
7756 }
7757
7758 \newif\ifnotes
7759 \bool_if:NTF \c_notesslides_notes_bool {
7760   \notestrue
7761   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7762   \RequirePackage[noamsthm,hyperref]{beamerarticle}
7763   \RequirePackage{mdframed}
7764   \str_if_empty:NTF \c_notesslides_topsect_str{
7765     \%setsectionlevel{section}
7766   }{
7767     \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7768   }

```

```

7769 }{
7770   \notesfalse
7771
7772   \cs_new_protected:Nn \__notesslides_do_sectocframes: {
7773     \cs_set_protected:Nn \__notesslides_do_label:n {
7774       \str_case:nnF{##1} {
7775         {part} {
7776           \tl_set:Nx\l__notesslides_num{\the part}
7777           \tl_set:cx{@ @ label}{%
7778             \cs_if_exist:NTF\parttitlename{\exp_not:N\parttitlename}{\exp_not:N\partname}{-}}
7779         }
7780         {chapter} {
7781           \tl_set:Nx\l__notesslides_num{\the chapter}
7782           \tl_set:cx{@ @ label}{%
7783             \cs_if_exist:NTF\chapertitlename{\exp_not:N\chapertitlename}{\exp_not:N\chaptername}{-}}
7784         }
7785         {section} {
7786           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesection.}
7787           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7788         }
7789         {subsection} {
7790           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsection.}
7791           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7792         }
7793         {subsubsection} {
7794           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsubsection.}
7795           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7796         }
7797         {paragraph} {
7798           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubparagraph.}
7799           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7800         }
7801       }{
7802         \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsubsubsection.}
7803         \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7804       }
7805     }
7806     \cs_set_protected:Nn \_sfragment_do_level:nn {
7807       \tl_if_exist:cT{c@##1}{\stepcounter{##1}}
7808       \addcontentsline{toc}{##1}{\protect\numberline{\use:c{the##1}}##2}
7809       \__notesslides_do_label:n{##1}
7810       \pdfbookmark[\int_use:N \l_stex_docheader_sect]{\l__notesslides_num##2}{##1.\l__note}
7811       \begin{frame}[noframenumbering]
7812         \vfill\centering
7813         \sectiontitleemph{%
7814           \use:c{@ @ label} ##2
7815         }
7816         \end{frame}
7817         \int_incr:N \l_stex_docheader_sect
7818         \tl_set:Nn \stex_current_section_level{##1}
7819       }
7820     }
7821
7822   \AtBeginDocument{

```

```

7823 \str_if_empty:NTF \c_notesslides_topsect_str {
7824   \setsectionlevel{section}
7825 } {
7826   \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7827   \exp_args:No \str_if_eq:nnTF \c_notesslides_topsect_str {chapter} {
7828     \__notesslides_define_chapter:
7829   }{
7830     \exp_args:No \str_if_eq:nnT \c_notesslides_topsect_str {part} {
7831       \__notesslides_define_chapter:
7832       \__notesslides_define_part:
7833     }
7834   }
7835 }
7836 }
7837
7838 \bool_if:NT \c_notesslides_sectocframes_bool {
7839   \__notesslides_do_sectocframes:
7840 }
7841 }

7842 \cs_new_protected:Nn \__notesslides_define_chapter: {
7843   \cs_if_exist:N \chaptername {
7844     \cs_set_protected:Npn \chaptername {Chapter}
7845   }
7846   \cs_if_exist:N \chapter {
7847     \cs_set_protected:Npn \chapter {INVALID}
7848   }
7849   \cs_if_exist:N \c@chapter {
7850     \newcounter{chapter}\counterwithin*{section}{chapter}
7851   }
7852 }
7853 }

7854 \cs_new_protected:Nn \__notesslides_define_part: {
7855   \cs_if_exist:N \partname {
7856     \cs_set_protected:Npn \partname {Part}
7857   }
7858   \cs_if_exist:N \part {
7859     \cs_set_protected:Npn \part {INVALID}
7860   }
7861   \cs_if_exist:N \c@part {
7862     \newcounter{part}\counterwithin*{chapter}{part}
7863   }
7864 }
7865 }

```

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

```

7866 \def \c_notesslides_document_str{document}
7867 \newcommand\afterprematurestop{}
7868 \def\prematurestop@endsfragment{
7869   \unless\ifx\@currenvir\c_notesslides_document_str
7870     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7871       \expandafter\prematurestop@endsfragment
7872     \fi
7873 }

```

```

7874 \providecommand\prematurestop{
7875   \stex_if_html_backend:F{
7876     \message{Stopping-sTeX-processing-prematurely}
7877     \prematurestop@endsfragment
7878   \afterprematurestop
7879   \end{document}
7880 }
7881 }

```

(End of definition for `\prematurestop`. This function is documented on page 98.)

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.

```

7882 \bool_if:NT \c_notesslides_notes_bool {
7883   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamertheme#2}}
7884 }
7885 \NewDocumentCommand \libusetheme {O{} m} {
7886   \libusepackage[#1]{beamertheme#2}
7887 }

```

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

```

7888 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7889 \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.

```

7890 \ifnotes
7891
7892 \newlength{\slideframewidth}
7893 \setlength{\slideframewidth}{1.5pt}

```

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

```

7894 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7895   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7896     \bool_set_true:N #1
7897   }{
7898     \bool_set_false:N #1
7899   }
7900 }
7901
7902 \stex_keys_define:nnnn{notesslides / frame}{
7903   \str_clear:N \l__notesslides_frame_label_str
7904   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7905   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7906   \bool_set_true:N \l__notesslides_frame_fragile_bool
7907   \bool_set_true:N \l__notesslides_frame_shrink_bool
7908   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7909   \bool_set_true:N \l__notesslides_frame_t_bool
7910 }
7911   label .str_set_x:N = \l__notesslides_frame_label_str,
7912   allowframebreaks .code:n = {

```

```

7913     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7914 },
7915 allowdisplaybreaks .code:n      = {
7916     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7917 },
7918 fragile          .code:n      = {
7919     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7920 },
7921 shrink           .code:n      = {
7922     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7923 },
7924 squeeze          .code:n      = {
7925     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7926 },
7927 t                .code:n      = {
7928     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7929 },
7930 unknown         .code:n      = {}
7931 }{}
```

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

```

7932 \cs_new_protected:Nn \__notesslides_setup_itemize: {
7933     \def\itemize@level{outer}
7934     \def\itemize@outer{outer}
7935     \def\itemize@inner{inner}
7936     \%newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
7937     \ renewenvironment{itemize} {
7938         \ifx\itemize@level\itemize@outer
7939             \def\itemize@label{$\rhd$}
7940         \fi
7941         \ifx\itemize@level\itemize@inner
7942             \def\itemize@label{$\scriptstyle\rhd$}
7943         \fi
7944         \begin{list}
7945             {\itemize@label}
7946             {\setlength{\labelsep}{.3em}
7947                 \setlength{\labelwidth}{.5em}
7948                 \setlength{\leftmargin}{1.5em}
7949             }
7950             \edef\itemize@level{\itemize@inner}
7951         }{
7952             \end{list}
7953         }
7954     }
```

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

```

7955 \stex_if_html_backend:TF {
7956     \cs_new_protected:Nn \__notesslides_frame_box_begin: {
7957         \vbox\bgroup
7958         \begin{stex_annotation_env}[shtml:frame={}]
7959             \mdf@patchamsthm\notesslidesfont
7960         }
7961     \cs_new_protected:Nn \__notesslides_frame_box_end: {
7962         %^A \notesslides@slidelabel
```

```

7963     \medskip\par\noindent\tiny\notesslidesfooter
7964     \end{stex_annotation_env}\egroup
7965   }
7966 }{
7967   \cs_new_protected:Nn \__notesslides_frame_box_begin: {
7968     \begin{mdframed}[
7969       linewidth=\slideframewidth,
7970       skipabove=1ex,
7971       skipbelow=1ex,
7972       userdefinedwidth=\slidewidth,
7973       align=center
7974     ]\notesslidesfont
7975   }
7976   \cs_new_protected:Nn \__notesslides_frame_box_end: {
7977     \medskip\par\noindent\tiny\notesslidesfooter%^^A\notesslides@slidelabel
7978     \end{mdframed}
7979   }
7980 }

```

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

```

7981 \renewenvironment{frame}[1][]{
7982   \stex_keys_set:nn{notesslides / frame}{#1}
7983   \stepcounter{framenumber}
7984   \renewcommand{\newpage}{\addtocounter{framenumber}{1}}
7985   \def\@currentlabel{\theframenumber}
7986   \str_if_empty:NF \l__notesslides_frame_label_str {
7987     \label{\l__notesslides_frame_label_str}
7988   }
7989   \__notesslides_setup_itemize:
7990   \__notesslides_frame_box_begin:
7991 }{
7992   \__notesslides_frame_box_end:
7993 }

```

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

```

\frametitle
7994 \renewcommand{\frametitle}[1]{
7995   \stexdoctitle { #1 }
7996   \notesslidestitlenameph{#1}\medskip
7997 }

```

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

```

\pause
7998 \newcommand{\pause}{}

```

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

We redefine the `columns` and `column` environments:

```

7999 \renewenvironment{columns}[1][]{
8000   \par\noindent
8001   \begin{minipage}
8002     \slidewidth\centering\leavevmode
8003   % \stex_if_html_backend:T{

```

```

8004 %     \cs_if_exist:NT \rustex_if:T {
8005 %         \rustex_if:T {\par
8006 %             \rustex_direct_HTML:n{<table><tr><td>}
8007 %         }
8008 %     }
8009 % }
8010 }{
8011 % \stex_if_html_backend:T{
8012 %     \cs_if_exist:NT \rustex_if:T {
8013 %         \rustex_if:T {\par
8014 %             \rustex_direct_HTML:n{</td></tr></table>}
8015 %         }
8016 %     }
8017 % }
8018     \end{minipage}\par\noindent
8019 }
8020 \newsavebox\columnbox
8021 \renewenvironment<>{column}[2] []{
8022     \begin{lrbox}{\columnbox}
8023 % \stex_if_html_backend:T{
8024 %     \cs_if_exist:NT \rustex_if:T {
8025 %         \rustex_if:T {\par
8026 %             \rustex_direct_HTML:n{</td><td>}
8027 %         }
8028 %     }
8029 % }
8030     \begin{minipage}{#2}
8031 }{
8032     \end{minipage}
8033 % \stex_if_html_backend:T{
8034 %     \cs_if_exist:NT \rustex_if:T {
8035 %         \rustex_if:T {\par
8036 %             \rustex_direct_HTML:n{</td><td>}
8037 %         }
8038 %     }
8039 % }
8040     \end{lrbox}\usebox\columnbox
8041 }
8042 \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.

```

8043 \bool_if:NTF \c_notesslides_notes_bool {
8044     \renewenvironment{note}{\ignorespaces}{}
8045 }{
8046     \renewenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup\egroup}
8047 }

```

For other environments we introduce variants prefixed with `n`, which are excluded in `slides` mode.

```

8048 \cs_new_protected:Nn \__notesslides_notes_env:nnnn {
8049   \bool_if:NTF \c_notesslides_notes_bool {
8050     \newenvironment{#1}#2{#3}{#4}
8051   }{
8052     \newenvironment{#1}#2{
8053       \cs_set:Npn \__notesslides_eat: #####1 \end #####2 {
8054         \str_if_eq:nnTF{#1}{#####2} {
8055           \end{#1}
8056         }{
8057           \__notesslides_eat:
8058         }
8059       }
8060       \__notesslides_eat:
8061       \%setbox\l_tmpa_box\vbox\bgroup#3
8062     }{
8063       %#4\egroup
8064     }
8065   }
8066 }

8067 \__notesslides_notes_env:nnnn{nparagraph}{[1][]}{\begin{sparagraph}{#1}}{\end{sparagraph}}
8068 \__notesslides_notes_env:nnnn{nfragment}{[2][]}{\begin{sfragment}{#1}{#2}}{\end{sfragment}}
8069 \__notesslides_notes_env:nnnn{ndefinition}{[1][]}{\begin{sdefinition}{#1}}{\end{sdefinition}}
8070 \__notesslides_notes_env:nnnn{nassertion}{[1][]}{\begin{sassertion}{#1}}{\end{sassertion}}
8071 \__notesslides_notes_env:nnnn{nproof}{[2][]}{\begin{sproof}{#1}{#2}}{\end{sproof}}
8072 \__notesslides_notes_env:nnnn{nexample}{[1][]}{\begin{sexample}{#1}}{\end{sexample}}
8073 \__notesslides_notes_env:nnnn{nexample}{[1][]}{\begin{sexample}{#1}}{\end{sexample}}
8074
8075 \RequirePackage{graphicx}
8076
8077 \NewDocumentCommand\frameimage{s O{} m}{
8078   \IfBooleanTF #1 {
8079     \begin{frame}[plain]
8080   }{
8081     \begin{frame}
8082   }
8083   \bool_if:NTF \c_notesslides_notes_bool {
8084     \slidewidth=\dimexpr\slidewidth-(2\slideframewidth)\relax
8085   }{
8086     \slidewidth=\textwidth\relax
8087   }
8088   \def\Gin@ewidth{}\setkeys{Gin}{#2}
8089   \tl_if_empty:NTF \Gin@ewidth {
8090     \mhgraphics[width=\slidewidth,#2]{#3}
8091   }{
8092     \mhgraphics[#2]{#3}
8093   }
8094   \end{frame}
8095 }

```

hacking inputref:

\inputref*

```

8096 \cs_set_eq:NN \__notesslides_inputref:\inputref
8097 \cs_set_protected:Npn \inputref{@ifstar\ninputref\__notesslides_inputref:}

```

```

8098 \bool_if:NTF \c_notesslides_notes_bool {
8099   \newcommand\nininputref[2][]{%
8100     \__notesslides_inputref:[#1]{#2}%
8101   }%
8102 }{%
8103   \newcommand\nininputref[2]{}%
8104 }

```

(End of definition for `\inputref*`. This function is documented on page 97.)

14.1.4 Styling Across Notes/Slides

```

8105 \def\notesslidestitleemph#1{%
8106   {\Large\bf\sf#1}%
8107   \vskip0.1\baselineskip
8108   \leaders\vrule width \textwidth
8109   \vskip0.4pt%
8110   \nointerlineskip
8111 }%
8112
8113 \def\notesslidesfooter{}%
8114
8115 \let\notesslidesfont\sffamily

```

14.1.5 Beamer Compatibility

All of this should be removed and made part of a template

```

8116
8117 \bool_if:NT \c_notesslides_notes_bool {%
8118   \def\author{\@dblarg\ns@author}%
8119   \long\def\ns@author[#1]#2{%
8120     \tl_if_empty:nTF{#1}{%
8121       \def\beamer@shortauthor{#2}%
8122     }{%
8123       \def\beamer@shortauthor{#1}%
8124     }%
8125   \def\@author{#2}%
8126 }%
8127 \def\title{\@dblarg\ns@title}%
8128 \long\def\ns@title[#1]#2{%
8129   \tl_if_empty:nTF{#1}{%
8130     \def\beamer@shorttitle{#2}%
8131   }{%
8132     \def\beamer@shorttitle{#1}%
8133   }%
8134   \def\@title{#2}%
8135   \stexdoctitle{#2}%
8136 }%
8137 \def\insertshortauthor{%
8138   \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shortauthor\beamer@shortauthor\egroup
8139 }%
8140 \def\insertshorttitle{%
8141   \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shorttitle\beamer@shorttitle\egroup
8142 }%
8143 \stex_if_html_backend:TF{%

```

```

8144     \def\insertframenumber{\stex_annotate:nn{shtml:framenumber={}}{}}
8145   }{
8146     \def\insertframenumber{\@arabic\c@framenumber}
8147   }
8148   \def\insertshortdate{\today}
8149 }

```

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.

```

8150 \gdef\printexcursions{}
8151 \newcommand\excursionref[2]{% label, text
8152   \bool_if:NT \c_notesslides_notes_bool {
8153     \begin{sparagraph}[title=Excursion]
8154       #2 \sref[fallback=the appendix]{#1}.
8155     \end{sparagraph}
8156   }
8157 }
8158 \newcommand\activate@excursion[2][]{%
8159   \tl_gput_right:Nn\printexcursions{\inputref[#1]{#2}}
8160 }
8161 \newcommand\excursion[4][]{% repos, label, path, text
8162   \bool_if:NT \c_notesslides_notes_bool {
8163     \activate@excursion[#1]{#3}
8164     \excursionref{#2}{#4}
8165   }
8166 }

```

(End of definition for `\excursion`. This function is documented on page 99.)

\excursiongroup

```

8167 \keys_define:nn{notesslides / excursiongroup }{
8168   id      .str_set_x:N = \l__notesslides_excursion_id_str,
8169   intro    .tl_set:N     = \l__notesslides_excursion_intro_tl,
8170   archive  .str_set_x:N = \l__notesslides_excursion_mrepos_str
8171 }
8172 \cs_new_protected:Nn \__notesslides_excursion_args:n {
8173   \tl_clear:N \l__notesslides_excursion_intro_tl
8174   \str_clear:N \l__notesslides_excursion_id_str
8175   \str_clear:N \l__notesslides_excursion_mrepos_str
8176   \keys_set:nn {notesslides / excursiongroup }{ #1 }
8177 }
8178 \newcommand\excursiongroup[1][]{%
8179   \__notesslides_excursion_args:n{ #1 }
8180   \tl_if_empty:NF\printexcursions
8181   {\IfInputref{}{\begin{note}
8182     \begin{sfragment}{Excursions}% TODO pass on id
8183     \ifdefempty{\l__notesslides_excursion_intro_tl}{}{
8184       \exp_args:NNe \use:nn \inputref{[\l__notesslides_excursion_mrepos_str]}{
8185         \l__notesslides_excursion_intro_tl
8186       }
8187     }
8188   }

```

```

8188     \printexcursions%
8189     \end{sfragment}
8190     \end{note}}}
8191 }
8192 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
(End of definition for \excursiongroup. This function is documented on page 99.)
8193 \prop_new:N \g__notesslides_variables_prop
8194 \cs_set_protected:Npn \setSGvar #1 #2 {
8195     \prop_gput:Nnn \g__notesslides_variables_prop {#1}{#2}
8196 }
8197 \cs_set_protected:Npn \useSGvar #1 {
8198     \prop_item:Nn \g__notesslides_variables_prop {#1}
8199 }
8200 \cs_set_protected:Npn \ifSGvar #1 #2 #3 {
8201     \prop_get:NnNF \g__notesslides_variables_prop {#1} \l__notesslides_tmp {
8202         \PackageError{document-structure}
8203         {The sTeX Global variable #1 is undefined}
8204         {set it with \protect\setSGvar}\TODO better error
8205     }
8206     \tl_if_eq:NnT \l__notesslides_tmp {#2}{ #3 }
8207 }
8208
8209
8210 </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.

```

8211 <*package>
8212 <@@=problems>
8213 \ProvidesExplPackage{problem}{2023/03/19}{3.3.0}{Semantic Markup for Problems}
8214 \RequirePackage{l3keys2e}
8215
8216 \keys_define:nn { problem / pkg }{
8217     notes      .default:n   = { true },
8218     notes      .bool_set:N = \c__problems_notes_bool,
8219     gnotes     .default:n   = { true },
8220     gnotes     .bool_set:N = \c__problems_gnotes_bool,
8221     hints      .default:n   = { true },
8222     hints      .bool_set:N = \c__problems_hints_bool,
8223     solutions   .default:n   = { true },
8224     solutions   .bool_set:N = \c__problems_solutions_bool,
8225     pts        .default:n   = { true },
8226     pts        .bool_set:N = \c__problems_pts_bool,
8227     min        .default:n   = { true },
8228     min        .bool_set:N = \c__problems_min_bool,
8229     %boxed     .default:n   = { true },
8230     %boxed     .bool_set:N = \c__problems_boxed_bool,

```

```

8231   test      .default:n    = { true },
8232   test      .bool_set:N   = \c__problems_test_bool,
8233   unknown   .code:n     = {
8234     \PassOptionsToPackage{\CurrentOption}{stex}
8235   }
8236 }
8237 \newif\ifsolutions
8238
8239 \ProcessKeysOptions{ problem / pkg }
8240 \bool_if:NTF \c__problems_solutions_bool {
8241   \solutionstrue
8242 }{
8243   \solutionsfalse
8244 }
8245 \RequirePackage{stex}

```

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

```

8246 \AddToHook{begindocument}{
8247   \ExplSyntaxOn\makeatletter
8248   \input{problem-english.ldf}
8249   \ltx@ifpackageloaded{babel}{
8250     \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bblobloaded}
8251       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8252         \input{problem-ngerman.ldf}
8253       }
8254       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8255         \input{problem-finnish.ldf}
8256       }
8257       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8258         \input{problem-french.ldf}
8259       }
8260       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8261         \input{problem-russian.ldf}
8262       }
8263   }{}
8264   \makeatother\ExplSyntaxOff
8265 }

```

(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.

```

8266 \stex_keys_define:nnnn{ problem }{
8267   \tl_set:Nn \l_stex_key_pts_tl 0
8268   \tl_set:Nn \l_stex_key_min_tl 0
8269   \str_clear:N \l_stex_key_name_str
8270   \str_clear:N \l_stex_key_mhrepos_str
8271 }{
8272   pts      .tl_set:N      = \l_stex_key_pts_tl,
8273   min     .tl_set:N      = \l_stex_key_min_tl,

```

```

8274     name .str_set:N = \l_stex_key_name_str,
8275     archive .str_set:N = \l_stex_key_mhrepos_str,
8276     creators .code:n = {}
8277     %imports .tl_set:N = \l__problems_prob_imports_tl,
8278     %refnum .int_set:N = \l__problems_prob_refnum_int,
8279 }{id,title,style,uses}

```

Then we set up a counter for problems.

```
\numberproblemsin
8280 \newcounter{sproblem}[section]
8281 \newcommand\numberproblemsin[1]{
8282   \@addtoreset{sproblem}{#1}
8283   \def\thesproblem{\arabic{#1}.\arabic{sproblem}}
8284 }
8285 \numberproblemsin{section}
8286 \%def\theplainsproblem{\arabic{sproblem}}
8287 \%def\thesproblem{\thesection.\theplainsproblem}
```

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

```
sproblem (env.)
8288 \newcounter{pts}
8289 \newcounter{min}
8290 \stex_new_stylable_env:nnnnnnn {problem} {0{}}{}{
8291   \cs_if_exist:NTF \l_problem_inputproblem_keys_tl {
8292     \tl_put_left:Nn \l_problem_inputproblem_keys_tl {#1,}
8293     \exp_args:Nno \stex_keys_set:nn{problem}{
8294       \l_problem_inputproblem_keys_tl
8295     }
8296   }{
8297     \stex_keys_set:nn{problem}{#1}
8298   }
8299   \refstepcounter{sproblem}
8300   \str_if_empty:NT \l_stex_key_name_str {
8301     \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8302     \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8303   }
8304
8305   \exp_args:No \stex_module_setup:n \l_stex_key_name_str
8306
8307   \stex_if_do_html:T {
8308     \exp_args:Nne \begin{stex_annotation_env} {
8309       shtml:problem={\l_stex_current_module_str},
8310       shtml:language={ \l_stex_current_language_str},
8311       shtml:signature={\l_stex_key_sig_str}
8312       \tl_if_empty:NF \l_stex_metatheory_uri {
8313         shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
8314       }
8315     }
8316     \stex_annotation_invisible:n{}
8317     \tl_if_empty:NF \l_stex_key_title_tl {
8318       \exp_args:No \stexdoctitle \l_stex_key_title_tl
8319     }
8320 }
```

```

8321 \stex_if_smsmode:F {
8322   \str_set_eq:NN \thismoduleuri \l_stex_current_module_str
8323   \tl_set_eq:NN \thismodulename \l_stex_key_name_str
8324   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
8325   \stex_style_apply:
8326   \addtocounter{pts}{\l_stex_key_pts_tl}
8327   \addtocounter{min}{\l_stex_key_min_tl}
8328   \stex_do_id:
8329   \__problems_record_problem:
8330 }
8331 \stex_reactivate_macro:N \solution
8332 \stex_reactivate_macro:N \mcb
8333 \stex_reactivate_macro:N \fillinsol
8334 \stex_smsmode_do:
8335 }{
8336   \stex_close_module:
8337   \stex_if_smsmode:F \stex_style_apply:
8338   \stex_if_do_html:T{ \end{stex_annotation_env} }
8339 }{
8340   \par\noindent\problemheader
8341   \bool_if:NT \c__problems_pts_bool {
8342     \marginpar{\l_stex_key_pts_tl{}~\problem@kw@points\smallskip}
8343   }
8344   \bool_if:NT \c__problems_min_bool {
8345     \marginpar{\l_stex_key_min_tl{}~\problem@kw@minutes\smallskip}
8346   }
8347   \par
8348   \stex_ignore_spaces_and_pars:
8349 }{
8350   \par\bigskip
8351 % \bool_if:NT \c__problems_test_bool \pagebreak
8352 }{s}
8353 \stex_sms_allow_env:n{sproblem}
8354
8355 \tl_set:Nn \problemheader {
8356   \textbf{\sproblemname{\~}\thesproblem}
8357   \tl_if_empty:NF \thistitle {
8358     {\~}(\thistitle)
8359   }
8360 }
8361 }
8362
8363 \cs_new_protected:Nn \__problems_record_problem: {
8364   \exp_args:NNN \iow_now:Nn \@auxout {
8365     \problem@restore {\thesproblem}{\l_stex_key_pts_tl}{\l_stex_key_min_tl}
8366   }
8367 }
8368
8369 \cs_new_protected:Npn \problem@restore #1 #2 #3 {}

\includeproblem
8370 \stex_keys_define:nnnn{ includeproblem }{
8371   \str_clear:N \l_stex_key_mhrepos_str
8372 }

```

```

8373     archive .str_set:N      = \l_stex_key_mhrepos_str,
8374     unknown .code:n = {}
8375 }{}
8376
8377 \NewDocumentCommand\includeproblem{O{} m}{
8378     \group_begin:
8379     \tl_set:Nn \l_problem_inputproblem_keys_tl {#1}
8380     \stex_keys_set:nn{includeproblem}{#1}
8381     \exp_args:Nno \use:nn{\inputref[]}\l_stex_key_mhrepos_str]{#2}
8382     \group_end:
8383 }
8384

```

(End of definition for `\includeproblem`. This function is documented on page 105.)

`solution (env.)`

```

8385 \int_new:N \g_problem_id_counter
8386
8387 \cs_new_protected:Nn \__problems_solution_start:n {
8388     \str_set:Nn \l_stex_key_title_tl {#1}
8389     \str_set:Nn \l_stex_key_id_str {#1}
8390     \str_if_empty:NT \l_stex_key_id_str {
8391         \int_gincr:N \g_problem_id_counter
8392         \str_set:Nx \l_stex_key_id_str {
8393             SOLUTION_\int_use:N \g_problem_id_counter
8394         }
8395     }
8396     \stex_if_do_html:T{
8397         \begin{stex_annotation_env}[
8398             shtml:solution=\l_stex_key_id_str
8399         ]
8400     }
8401     \stex_style_apply:
8402 }
8403
8404 \stex_new_stylable_env:nnnnnnn { solution }{ 0{} }{
8405     \stex_if_do_html:TF{
8406         \__problems_solution_start:n{#1}
8407     }{
8408         \ifsolutions
8409             \__problems_solution_start:n{#1}
8410         \else
8411             \setbox\l_tmpa_box\vbox\bgroup
8412         \fi
8413     }
8414 }{
8415     \stex_if_do_html:TF{
8416         \stex_style_apply:
8417         \end{stex_annotation_env}
8418 }{
8419     \ifsolutions
8420         \stex_style_apply:
8421         \stex_if_do_html:T{
8422             \end{stex_annotation_env}

```

```

8423     }
8424     \else
8425         \egroup
8426     \fi
8427 }
8428 }{
8429     \par\smallskip\hrule\smallskip
8430     \noindent\emph{\problem@kw@solution\str_if_empty:NF \l_stex_key_title_tl{%
8431         \~{}\l_stex_key_title_tl
8432     } :~}}
8433 }{
8434     \par\smallskip\hrule
8435 }{}}
8436
8437 \stex_deactivate_macro:Nn \solution {sproblem~environments}

\startsolution
\stopsolution
8438 \cs_new_protected:Npn \startsolution{
8439     \global\solutionstrue
8440 }
8441 \cs_new_protected:Npn \stopsolution{
8442     \global\solutionsfalse
8443 }

(End of definition for \startsolution and \stopsolution. These functions are documented on page
101.)
```

`hint (env.)`

```

8444 \cs_new_protected:Nn \__problems_hint_start:n {
8445     \str_set:Nn \l_stex_key_title_tl {\#1}
8446     \str_set:Nn \l_stex_key_id_str {\#1}
8447     \str_if_empty:NT \l_stex_key_id_str {
8448         \int_gincr:N \g_problem_id_counter
8449         \str_set:Nx \l_stex_key_id_str {
8450             HINT_\int_use:N \g_problem_id_counter
8451         }
8452     }
8453     \stex_if_do_html:T{
8454         \begin{stex_annotation_env}%
8455             \shtml:problemhint=\l_stex_key_id_str
8456         %
8457     }
8458     \stex_style_apply:
8459 }
8460
8461 \stex_new_stylable_env:nnnnnnn { hint }{ 0{} }{
8462     \stex_if_do_html:TF{
8463         \__problems_hint_start:n{\#1}
8464     }{
8465         \bool_if:NTF \c__problems_hints_bool {
8466             \__problems_hint_start:n{\#1}
8467         }{
8468             \setbox\l_tmpa_box\vbox\bgroup
8469         }
```

```

8470     }
8471 }{
8472     \stex_if_do_html:TF{
8473         \stex_style_apply:
8474         \end{stex_annotation_env}
8475 }{
8476     \bool_if:NTF \c__problems_hints_bool {
8477         \stex_style_apply:
8478         \stex_if_do_html:T{
8479             \end{stex_annotation_env}
8480         }
8481     }{
8482         \egroup
8483     }
8484 }
8485 }{
8486     \par\smallskip\hrule\smallskip
8487     \noindent\emph{Hint}\str_if_empty:N \l_stex_key_title_tl{
8488         {~}\l_stex_key_title_tl
8489     } :~}
8490 }{
8491     \par\smallskip\hrule
8492 }{}}

exnote (env.)
8493 \cs_new_protected:Nn \__problems_exnote_start:n {
8494     \str_set:Nn \l_stex_key_title_tl {\#1}
8495     \str_set:Nn \l_stex_key_id_str {\#1}
8496     \str_if_empty:NT \l_stex_key_id_str {
8497         \int_gincr:N \g_problem_id_counter
8498         \str_set:Nx \l_stex_key_id_str {
8499             EXNOTE_\int_use:N \g_problem_id_counter
8500         }
8501     }
8502     \stex_if_do_html:T{
8503         \begin{stex_annotation_env} {
8504             shtml:problemnote=\l_stex_key_id_str
8505         }
8506     }
8507     \stex_style_apply:
8508 }
8509
8510 \stex_new_stylable_env:nnnnnnn { exnote }{ 0{} }{
8511     \stex_if_do_html:TF{
8512         \__problems_exnote_start:n{\#1}
8513     }{
8514         \bool_if:NTF \c__problems_notes_bool {
8515             \__problems_exnote_start:n{\#1}
8516         }{
8517             \setbox\l_tmpa_box\vbox\bgroup
8518         }
8519     }
8520 }{
8521     \stex_if_do_html:TF{

```

```

8522     \stex_style_apply:
8523     \end{stex_annotation_env}
8524 }{
8525     \bool_if:NTF \c__problems_notes_bool {
8526         \stex_style_apply:
8527         \stex_if_do_html:T{
8528             \end{stex_annotation_env}
8529         }
8530     }{
8531         \egroup
8532     }
8533 }
8534 }{
8535     \par\smallskip\hrule\smallskip
8536     \noindent\emph{Note}\str_if_empty:N \l_stex_key_title_tl{
8537         \l_stex_key_title_tl
8538     } :~}
8539 }{
8540     \par\smallskip\hrule
8541 }{}}

gnote (env.)
8542 \cs_new_protected:Nn \__problems_gnote_start:n {
8543     \str_set:Nn \l_stex_key_title_tl {\#1}
8544     \str_set:Nn \l_stex_key_id_str {\#1}
8545     \str_if_empty:NT \l_stex_key_id_str {
8546         \int_gincr:N \g_problem_id_counter
8547         \str_set:Nx \l_stex_key_id_str {
8548             GNOTE_\int_use:N \g_problem_id_counter
8549         }
8550     }
8551     \stex_if_do_html:T{
8552         \begin{stex_annotation_env} {
8553             \shtml:problemgnote=\l_stex_key_id_str
8554         }
8555     }
8556     \stex_style_apply:
8557 }

8558 \stex_new_stylable_env:nnnnnnn { gnote }{ 0{} }{
8559     \stex_if_do_html:TF{
8560         \__problems_gnote_start:n{\#1}
8561     }{
8562         \bool_if:NTF \c__problems_gnotes_bool {
8563             \__problems_gnote_start:n{\#1}
8564         }{
8565             \setbox\l_tmpa_box\vbox\bgroup
8566         }
8567     }
8568 }
8569 }{
8570     \stex_if_do_html:TF{
8571         \stex_style_apply:
8572         \end{stex_annotation_env}
8573 }{

```

```

8574     \bool_if:NTF \c__problems_gnotes_bool {
8575         \stex_style_apply:
8576         \stex_if_do_html:T{
8577             \end{stex_annotation_env}
8578         }
8579     }{
8580         \egroup
8581     }
8582 }
8583 }{
8584     \par\smallskip\hrule\smallskip
8585     \noindent\emph{Grading}\str_if_empty:N \l_stex_key_title_tl{
8586         \l_stex_key_title_tl
8587     } :~
8588 }{
8589     \par\smallskip\hrule
8590 }{}
```

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

```

8591 \def\pts#1{
8592     \bool_if:NT \c__problems_pts_bool {
8593         \stex_annotation:nn{shtml:problempoints}{\marginpar{\#1~\problem@kw@points}}
8594     }
8595 }
8596 \def\min#1{
8597     \bool_if:NT \c__problems_min_bool {
8598         \stex_annotation:nn{shtml:problemminutes}{\marginpar{\#1~\problem@kw@minutes}}
8599     }
8600 }
```

mcb (env.)

```

8601 \newenvironment{mcb}{\par
8602     \stex_if_do_html:T{
8603         \begin{stex_annotation_env}{shtml:multiple-choice-block={}}
8604     }
8605     \stex_deactivate_macro:Nn \mcb {sproblem~environments}
8606     \stex_deactivate_macro:Nn \solution {sproblem~environments}
8607     \stex_reactivate_macro:N \mcc
8608     \begin{enumerate}
8609     }{
8610         \end{enumerate}
8611         \stex_if_do_html:T{
8612             \end{stex_annotation_env}
8613         }
8614     }
8615 \stex_deactivate_macro:Nn \mcb {sproblem~environments}
```

we define the keys for the `mcc` macro

```

8616 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8617     \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8618         \bool_set_true:N #1
8619     }{
8620         \bool_set_false:N #1
8621     }
```

```

8622 }
8623 \stex_keys_define:nnnn{mcc}{
8624   \tl_clear:N \l_stex_key_feedback_tl
8625   \bool_set_false:N \l_stex_key_T_bool
8626   \tl_clear:N \l_stex_key_Ttext_tl
8627   \tl_clear:N \l_stex_key_Ftext_tl
8628 }{
8629   feedback .tl_set:N      = \l_stex_key_feedback_tl ,
8630   T        .default:n    = { false } ,
8631   T        .bool_set:N   = \l_stex_key_T_bool ,
8632   F        .default:n    = { false } ,
8633   F        .code:n       = {\bool_set_false:N \l_stex_key_T_bool} ,
8634   Ttext    .tl_set:N     = \l_stex_key_Ttext_tl ,
8635   Ftext    .tl_set:N     = \l_stex_key_Ftext_tl ,
8636 }{id}
8637

\mcc

8638 \tl_set:Nn \problem_mcc_box_tl {
8639   \ltx@ifpackageloaded{amssymb}{$\square$}){
8640     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
8641   }
8642 }
8643 \newcommand\mcc[2][]{
8644   \stex_keys_set:nn{mcc}{#1}\par
8645   \tl_set:Nn \l_tmpb_tl {~~~
8646     \bool_if:NTF \l_stex_key_T_bool {
8647       \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
8648     }{
8649       \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
8650     }
8651     \tl_if_empty:NF \l_stex_key_feedback_tl {
8652       \\\emph{\l_stex_key_feedback_tl}
8653     }
8654   }
8655   \tl_set:Nn \l_tmpa_tl {
8656     #2
8657     \stex_if_do_html:TF{
8658       \stex_annotation:nn{shtml:mcc-solution={}}{\l_tmpb_tl}
8659     }{
8660       \if solutions \l_tmpb_tl \fi
8661     }
8662   }
8663   \item[\problem_mcc_box_tl]{\l_tmpa_tl}
8664   \stex_if_do_html:TF{
8665     \stex_annotation:nn{shtml:mcc=}
8666     \bool_if:NTF \l_stex_key_T_bool {true}{false}
8667   }{\l_tmpa_tl}
8668 }{\l_tmpa_tl}
8669 }
8670 \stex_deactivate_macro:Nn \mcc {mcb~environments}

```

(End of definition for `\mcc`. This function is documented on page 102.)

```

\fillinsol
8671 \newcommand\fillinsol[2][]{%
8672   \quad
8673   \mode_if_math:TF{%
8674     \hbox{\_\_problems_fillinsol:nn{\#1}{\#2}}%
8675   }{%
8676     \_\_problems_fillinsol:nn{\#1}{\#2}%
8677   }
8678   \quad
8679 }
8680 \cs_new_protected:Nn \_\_problems_fillinsol:nn {
8681   \stex_if_do_html:TF{%
8682     \stex_annotation:n{html:fillinsol={}}{ \stex_annotation_force_break:n{\#2} }%
8683   }{%
8684     \if solutions
8685       \textcolor{red}{\fbox{\#2}}%
8686     \else
8687       \fbox{%
8688         \tl_if_empty:nTF{\#1}{%
8689           \phantom{\huge{\#2}}%
8690         }{%
8691           \hspace{\#1}%
8692         }%
8693       }%
8694     \fi
8695   }%
8696 }%
8697 \stex_deactivate_macro:Nn \fillinsol {sproblem~environments}

```

(End of definition for `\fillinsol`. This function is documented on page 104.)

`\testemptypage`

```

8698 \newcommand\testemptypage[1][]{%
8699   \bool_if:NT \c__problems_test_bool {\vfill\begin{center}\hwexam@kw@testemptypage\end{center}}%
8700 }

```

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

`\testspace`

```

8701 \newcommand\testspace[1]{\bool_if:NT \c__problems_test_bool {\vspace*{\#1}}}

```

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

`\testnewpage`

```

8702 \newcommand\testnewpage{\bool_if:NT \c__problems_test_bool {\newpage}}

```

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

```

8703 
```

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.

```
8704 <*package>
8705 \ProvidesExplPackage{hwexam}{2023/03/19}{3.3.0}{homework assignments and exams}
8706 \RequirePackage{l3keys2e}
8707
8708 \keys_define:nn {hwexam / pkg} {
8709   multiple .default:n = { false },
8710   multiple .bool_set:N = \c_hwexam_multiple_bool,
8711   unknown .code:n = {
8712     \PassOptionsToPackage{\CurrentOption}{problem}
8713   }
8714 }
8715 \ProcessKeysOptions{ hwexam /pkg }
8716 \RequirePackage{problem}
```

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

```
8717 \AddToHook{begindocument}{
8718   \ExplSyntaxOn\makeatletter
8719   \input{hwexam-english.ldf}
8720   \ltx@ifpackageloaded[babel]{
8721     \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
8722     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8723       \input{hwexam-ngerman.ldf}
8724     }
8725     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8726       \input{hwexam-finnish.ldf}
8727     }
8728     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8729       \input{hwexam-french.ldf}
8730     }
8731     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8732       \input{hwexam-russian.ldf}
8733     }
8734   }{}}
8735   \makeatother\ExplSyntaxOff
8736 }
```

(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.)
8737 \stex_keys_define:nnnn{ assignment }{
8738   \tl_clear:N \l_stex_key_number_tl
8739 \tl_clear:N \l_stex_key_given_tl
8740 \tl_clear:N \l_stex_key_due_tl
8741 }{
8742   number .tl_set:N = \l_stex_key_number_tl,
8743   given .tl_set:N = \l_stex_key_given_tl,
8744   due .tl_set:N = \l_stex_key_due_tl,
8745   unknown .code:n = {}
8746 }{id,title,style}
8747
8748 \newcounter{assignment}
8749 \stex_new_stylable_env:nnnnnnn {assignment}{0}{}{
8750   \cs_if_exist:NTF \l_hwexam_includeassignment_keys_tl {
8751     \tl_put_left:Nn \l_hwexam_includeassignment_keys_tl {#1,}
8752     \exp_args:Nno \stex_keys_set:nn{assignment}{}{
8753       \l_hwexam_includeassignment_keys_tl
8754     }
8755   }{
8756     \stex_keys_set:nn{assignment}{#1}
8757   }
8758 \tl_if_empty:NF \l_stex_key_number_tl {
8759 \global\setcounter{assignment}{\int_eval:n{\l_stex_key_number_tl-1}}
8760 }
8761 \global\refstepcounter{assignment}
8762 \setcounter{sproblem}{0}
8763 \def\thesproblem{\theassignment.\arabic{sproblem}}
8764 \stex_style_apply:
8765 \_stex_do_id:
8766 }{
8767 \stex_style_apply:
8768 }{
8769 \par\begin{center}
8770 \textbf{\Large\assignmentautorefname~\theassignment}
8771 \tl_if_empty:NF \l_stex_key_title_tl {
8772 {\~}---\l_stex_key_title_tl
8773 }
8774 }\par\smallskip
8775 \textbf{
8776 \tl_if_empty:NF \l_stex_key_given_tl {
8777 \hwexam@kw@given :~\l_stex_key_given_tl\quad
8778 }
8779 \tl_if_empty:NF \l_stex_key_due_tl {
8780 \hwexam@kw@due :~\l_stex_key_due_tl\quad
8781 }
8782 }
8783 \end{center}
8784 \par\bigskip
8785 }{
8786 \par\pagebreak
8787 }{}}

\includeassignment

```

```

8788 \NewDocumentCommand\includeassignment{O{} m}{
8789   \group_begin:
8790   \tl_set:Nn \l_hwexam_includeassignment_keys_tl {#1}
8791   \stex_keys_set:nn{includeproblem}{#1}
8792   \exp_args:Nno \use:nn{\inputref[]}\l_stex_key_mhrepos_str]{#2}
8793   \group_end:
8794 }
```

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

Restoring information about problems:

```

8795 \prop_new:N \c_@@_problems_prop
8796 \tl_set:Nn \c_@@_total_mins_tl {0}
8797 \tl_set:Nn \c_@@_total_pts_tl {0}
8798 \int_new:N \c_@@_total_problems_int
8799 \cs_set_protected:Npn \problem@restore #1 #2 #3 {
8800   \int_gincr:N \c_@@_total_problems_int
8801   \prop_gput:Nnn \c_@@_problems_prop {#1}{#2}{#3}
8802 \tl_gset:Nx \c_@@_total_pts_tl { \int_eval:n { \c_@@_total_pts_tl + #2 } }
8803 \tl_gset:Nx \c_@@_total_mins_tl { \int_eval:n { \c_@@_total_mins_tl + #2 } }
8804 }
```

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

```

8805 \newcommand\correction@table{
8806   \int_compare:nNnT \c_@@_total_problems_int = 0 {
8807     \int_incr:N \c_@@_total_problems_int
8808     \prop_put:Nnn \c_@@_problems_prop {\sim}{\sim}{\sim}
8809   }
8810   \tl_clear:N \l_tmpa_tl
8811   \tl_clear:N \l_tmpb_tl
8812   \tl_clear:N \l_tmpc_tl
8813   \prop_map_inline:Nn \c_@@_problems_prop {
8814     \tl_put_right:Nn \l_tmpa_tl { ##1 & }
8815     \tl_put_right:Nx \l_tmpb_tl { \use_i:nn ##2 & }
8816     \tl_put_right:Nn \l_tmpc_tl { & }
8817   }
8818   \resizebox{\textwidth}{!}{%
8819     \exp_args:Nne \begin{tabular}{|l|*{\int_use:N \c_@@_total_problems_int}{c|}c||l|}\hline
8820       &\exp_args:Ne \multicolumn{\int_eval:n{ \c_@@_total_problems_int + 1}}{c||}{}
8821       {\footnotesize\hwexam@kw@forgrading} &\hline
8822       \hwexam@kw@probs & \l_tmpa_tl \hwexam@kw@sum & \hwexam@kw@grade\hline
8823       \hwexam@kw@pts & \l_tmpb_tl \c_@@_total_pts_tl & \hline
8824       \hwexam@kw@reached & \l_tmpc_tl & [.7cm]\hline
8825     \end{tabular}}}
```

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

`\testheading`

```

8826 \def\hwexamheader{\input{hwexam-default.header}}
8827
8828 \def\hwexamminutes{
8829   \tl_if_empty:NTF \hwexam@duration {
8830     \hwexam@min}~\hwexam@minutes@kw
8831   }{
8832   \hwexam@duration
```

```

8833 }
8834 }
8835
8836 \stex_keys_define:nnnn{ hwexam / testheading }{
8837 \tl_clear:N \hwexam@min
8838 \tl_clear:N \hwexam@duration
8839 \tl_clear:N \hwexam@reqpts
8840 \tl_clear:N \hwexam@tools
8841 }{
8842 min .tl_set:N = \hwexam@min,
8843 duration .tl_set:N = \hwexam@duration,
8844 reqpts .tl_set:N = \hwexam@reqpts,
8845 tools .tl_set:N = \hwexam@tools
8846 }{}}
8847
8848 \newenvironment{testheading}[1][]{%
8849 \stex_keys_set:nn { hwexam / testheading}{#1}%
8850
8851 \tl_set_eq:NN \hwexam@totalpts \c_@@_total_pts_tl
8852 \tl_set_eq:NN \hwexam@totalmin \c_@@_total_mins_tl
8853 \tl_set:Nx \hwexam@checktime {\int_eval:n { \hwexam@min - \hwexam@totalmin }}%
8854
8855 \newif\if@bonuspoints
8856 \tl_if_empty:NTF \hwexam@reqpts {
8857 \@bonuspointsfalse
8858 }{
8859 \tl_set:Nx \hwexam@bonuspts {
8860 \int_eval:n{\hwexam@totalpts - \hwexam@reqpts}
8861 }
8862 \@bonuspointstrue
8863 }
8864
8865 \makeatletter\hwexamheader\makeatother
8866 }{
8867 \newpage
8868 }

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

8869 ⟨/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

```

8870 <@=tikzinput>
8871 <*package>
8872
8873 %%%%%%%%%%%%%% tikzinput.dtx %%%%%%%%%%%%%%
8874
8875 \ProvidesExplPackage{tikzinput}{2023/03/19}{3.3.0}{tikzinput package}
8876 \RequirePackage{l3keys2e}
8877
8878 \keys_define:nn { tikzinput } {
8879   image .bool_set:N = \c_tikzinput_image_bool,
8880   image .default:n = false ,
8881   unknown .code:n = {}
8882 }
8883
8884 \ProcessKeysOptions { tikzinput }
8885
8886 \bool_if:NTF \c_tikzinput_image_bool {
8887   \RequirePackage{graphicx}
8888
8889   \providecommand\usetikzlibrary[]{}
8890   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
8891 }
8892   \RequirePackage{tikz}
8893   \RequirePackage{standalone}
8894
8895 \newcommand \tikzinput [2] [] {
8896   \setkeys{Gin}{#1}
8897   \ifx \Gin@ewidth \Gin@exclamation
8898     \ifx \Gin@eheight \Gin@exclamation
8899       \input { #2 }
8900     \else
8901       \resizebox{!}{\Gin@eheight }{
8902         \input { #2 }
8903       }
8904     \fi
8905   \else
8906     \ifx \Gin@eheight \Gin@exclamation
8907       \resizebox{ \Gin@ewidth }{! }{
8908         \input { #2 }
8909       }
8910     \else
8911       \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
8912         \input { #2 }
8913       }
8914

```

```
8914         \fi
8915     \fi
8916 }
8917 }
8918 \newcommand \ctikzinput [2] [] {
8919     \begin{center}
8920         \tikzinput [#1] {#2}
8921     \end{center}
8922 }
8923 }
8924 </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 | |
|--|---|
| \\$ | 2083, 2086, 2090 |
| \; | <u>7489, 7520, 7533, 7535, 7548</u> |
| @@ commands: | |
| \c_@@_problems_prop | 8795, 8801, 8808, 8813 |
| \c_@@_total_mins_tl .. | <u>8796, 8803, 8852</u> |
| \c_@@_total_problems_int | .. <u>8798, 8800, 8806, 8807, 8819, 8820</u> |
| \c_@@_total_pts_tl | 8797, 8802, 8823, 8851 |
| \\ | <u>52, 76, 1054, 7778, 7783, 8138, 8141, 8652, 8821, 8822, 8823, 8824</u> |
| \{ | 7538 |
| \} | 7538 |
| _ | 43, 639, 7810, 8699 |
| _comp | 3943, 4149, 4490, 5186, 6005, 6029, 6113 |
| _customthiscomp | 6532, 6533, 6545 |
| _defcomp | 6037, 6971, 6980 |
| _stex_html_do_output_bool | 278, 279, 282, 287, 291, 322, 325, 2152, 7480 |
| _thiscomp | 6515, 6526, 6527 |
| _varcomp | 4471, 4770, 4779, 4936, 4974, 5220, 5662, 5961, 5974, 5987, 6033 |
| \ | 2082, 2085, 2089 |
| A | |
| \activateexcursion | 99 |
| \addbibresource | <u>70, 1952</u> |
| \addcontentsline | 7808 |
| \addmhbibresource | <u>70, 1946</u> |
| \addtocounter | 7984, 8326, 8327 |
| \AddToHook | 1036, 1039, 7284, 7351, 7354, 7466, 8246, 8717 |
| \aftergroup | <u>133, 5174, 7349, 7463</u> |
| \afterprematurestop | <u>98, 7867, 7878</u> |
| \apply | 78 |
| \arabic | 8283, 8286, 8763 |
| \arg | 41, 83, 5399 |
| \argarraymap | 82, 4210, 4710 |
| \argmap | 82, 4209, 4351, <u>4686, 7521</u> |
| \argsep | 35, 82, 4208, 4346, <u>4670, 7489, 7490, 7491, 7497</u> |
| \assign | <u>59, 119, 2988, 3344, 3471</u> |
| assignment (env.) | <u>106, 8737</u> |
| B | |
| \assignmentautorefname | 8770 |
| \assignMorphism | <u>119, 2989, 3473</u> |
| \assumption | 7262, <u>7367</u> |
| \ast | 7512 |
| \AtBeginDocument | 1006, 1259, 1445, 1448, 1494, 7679, 7822 |
| \AtEndDocument | 647, 1495 |
| \AtEndOfPackageFile | 2033, 2047, 2060 |
| \author | 8118 |
| \autoref | <u>74, 1639</u> |
| B | |
| \backmatter | 1459, 1460, 1461 |
| \baselineskip | 8107 |
| \beameritemnestingprefix | 8192 |
| \begin ... | <u>86, 125, 126, 131, 191, 1302, 1382, 2044, 2057, 2075, 2232, 2254, 2271, 2521, 2779, 2825, 4733, 6866, 7176, 7209, 7231, 7269, 7340, 7412, 7608, 7811, 7944, 7958, 7968, 8001, 8022, 8030, 8068, 8069, 8070, 8071, 8072, 8073, 8079, 8081, 8153, 8181, 8182, 8308, 8397, 8454, 8503, 8552, 8603, 8608, 8699, 8769, 8819, 8920</u> |
| \begingroup | 358, 1924 |
| \bf | 8106 |
| \bgroup | |
| 3786, 4816, 5028, 7957, 8046, 8061, 8138, 8141, 8411, 8468, 8517, 8566 | |
| \bigskip | 8350, 8784 |
| blindfragment (env.) | <u>72</u> |
| bool commands: | |
| \bool_if:NTF | 196, 618, 670, 671, 677, 1144, 1154, 1156, 1215, 1236, 2135, 2390, 2740, 2773, 2957, 3453, 3659, 4831, 4934, 5184, 5218, 5760, 5854, 5860, 5995, 6961, 7237, 7285, 7323, 7342, 7345, 7347, 7352, 7355, 7363, 7405, 7419, 7441, 7467, 7714, 7722, 7759, 7838, 7882, 8043, 8049, 8083, 8098, 8117, 8152, 8162, 8240, 8341, 8344, 8351, 8465, 8476, 8514, 8525, 8563, 8574, 8592, 8597, 8646, 8666, 8699, 8701, 8702, 8886 |
| \bool_if:nTF | 282 |
| \bool_if_exist:NTF | <u>255, 271</u> |

```

\bbool_lazy_any:nTF ..... 4038, 4078
\bbool_lazy_any_p:n ..... 773
\bbool_new:N ..... 278, 2132,
2375, 2798, 4742, 5154, 5729, 5851
\bbool_not_p:n ..... 772, 776
\bbool_set_false:N ..... 181, 287, 325, 1141, 1165,
2133, 2152, 2376, 2753, 2820, 3426,
4602, 4745, 5158, 5730, 5855, 7095,
7244, 7292, 7320, 7461, 7480, 7481,
7698, 7739, 7898, 8620, 8625, 8633
\bbool_set_true:N ..... 186, 257, 273, 279, 291,
322, 607, 613, 1138, 2151, 2737,
2817, 3438, 3448, 3656, 3787, 4836,
5076, 5155, 5384, 5393, 5509, 5520,
5618, 5703, 5761, 5772, 5805, 5834,
5880, 6076, 6487, 6518, 6535, 6580,
7107, 7274, 7711, 7751, 7896, 7904,
7905, 7906, 7907, 7908, 7909, 8618
\bbool_while_do:Nn ..... 1139
\bbool_while_do:nn ..... 771, 1984, 7111, 7123, 7135, 7152, 7164
\l_tmpa_bool .. 3426, 3438, 3448, 3453
box commands:
\box_clear:N ..... 2155
\c_empty_box ..... 3968, 3975
\l_tmpa_box 2150, 2155, 3671, 4201,
8046, 8061, 8411, 8468, 8517, 8566
boxed ..... 100

C
\catcode ..... 43, 52, 358, 639, 1054,
1692, 2081, 2082, 2083, 2084, 2085,
2086, 2088, 2089, 2090, 2432, 2433
\cdot ..... 7520, 7533
\centering ..... 7812, 8002
\chapter ..... 22, 72, 7847, 7848
\chaptername ..... 7783, 7844, 7845
\chapertitlename ..... 7783
\circ ..... 49
\clearpage ..... 1454, 1464, 1475, 1486
clist commands:
\clist_clear:N 93, 405, 3699, 4127,
5077, 6369, 6468, 6783, 6788, 7369
\clist_count:N ..... 6349, 6359
\clist_count:n ..... 4718, 6451
\clist_get:NN ..... 461, 512
\clist_if_empty:NTF ..... 180, 457,
508, 3803, 4299, 6348, 6394, 6651, 6855
\clist_if_in:NnTF ..... 65, 68, 99, 6949, 8251, 8254,
8257, 8260, 8722, 8725, 8728, 8731
\clist_if_in:nnTF ..... 6477
\clist_item:Nn ..... 4308
\clist_item:nn ..... 4729
\clist_map_function:NN ..... 5078, 6397
\clist_map_function:nN ..... 2690, 2715, 3590, 3610, 3631
\clist_map_inline:Nn ..... 94, 103, 183, 2842, 3805, 4854,
5047, 6176, 6226, 6652, 6765, 6801
\clist_map_inline:nn ..... 369, 418, 5550, 5649, 5678, 6174, 7051
\clist_pop:NN ..... 4310
\clist_put_right:Nn ..... 95, 5091, 5095, 6376, 6495, 6497
\clist_set:Nn ..... 37, 91, 4955, 8250, 8721
\clist_set_eq:NN ..... 89, 6871
\l_tmpa_clist 8250, 8251, 8254, 8257,
8260, 8721, 8722, 8725, 8728, 8731
\clstinputmhlisting ..... 73, 2032
\cmhgraphics ..... 73, 2032
\cmhtikzinput ..... 108, 2032
\columnbox ..... 8020, 8022, 8040
\comp ..... 27, 28, 31, 41, 80,
83, 93, 121, 3943, 4149, 4322, 4471,
4490, 4532, 4538, 4540, 4770, 4779,
4974, 5203, 5204, 5538, 5662, 5882,
5887, 5902, 5961, 5974, 5987, 6005,
6007, 6014, 6113, 6336, 6344, 6527,
6541, 6542, 6971, 6972, 6980, 7486,
7488, 7495, 7497, 7499, 7500, 7505,
7507, 7512, 7514, 7515, 7522, 7523,
7524, 7525, 7528, 7530, 7534, 7535,
7536, 7538, 7548, 7555, 7557, 7559
\compemph ..... 93, 6014
\conclude ..... 7261, 7367
\conclusion 45–47, 89, 91, 6940, 7060, 7076
\copymod ..... 57, 58, 3600, 3602, 3604
\copymodule ..... 3490, 3492
\counterwithin ..... 7851, 7863
cs commands:
\cs:w ..... 444, 447, 479, 640, 2165,
2394, 2395, 2396, 2403, 2407, 2413,
5282, 6217, 6267, 6269, 6889, 6902
\cs_argument_spec:N ..... 3992
\cs_end: .. 444, 447, 479, 640, 2165,
2394, 2395, 2396, 2405, 2409, 2413,
5282, 6217, 6267, 6269, 6889, 6902
\cs_generate_from_arg_count:NNnn
..... 3796,
4598, 4845, 5038, 5122, 5323, 5355
\cs_generate_variant:Nn ..... 86, 156, 230, 243, 557, 581,
590, 602, 626, 686, 715, 870, 875,
879, 963, 1125, 1150, 1552, 1880,

```

2197, 2555, 2559, 2564, 2582, 2631,
 2653, 2666, 3121, 5179, 5196, 5513
`\cs_gset:Npn` 2428
`\cs_if_eq:NNTF` 139,
 885, 939, 1421, 2221, 2232, 2235,
 2254, 2257, 3994, 4671, 5594, 5596
`\cs_if_exist:NTF` 308,
 1093, 1327, 1331, 1369, 1372, 1396,
 1400, 1429, 1435, 1449, 1459, 1472,
 1483, 1638, 1786, 1810, 1830, 1836,
 1840, 2103, 2643, 2648, 3970, 3977,
 3989, 4420, 4428, 4869, 4876, 5063,
 5109, 5273, 5299, 5643, 5882, 5887,
 6433, 6730, 6733, 6737, 6740, 6745,
 6748, 6752, 6755, 7778, 7783, 7786,
 7790, 7794, 7798, 7802, 7844, 7847,
 7850, 7856, 7859, 7862, 8004, 8012,
 8024, 8034, 8138, 8141, 8291, 8750
`\cs_new:Nn` 208,
 431, 545, 551, 558, 567, 583, 592,
 764, 884, 2206, 2389, 2437, 2948,
 3766, 4097, 4110, 4314, 4381, 4398,
 4403, 4547, 5118, 5331, 5428, 6154,
 6209, 6405, 6411, 6414, 6424, 6846
`\cs_new:Npn` 543, 620, 621, 629, 630,
 880, 2889, 2920, 4103, 4116, 5574, 5947
`\cs_new_nopar:Nn` 365, 380
`\cs_new_protected:Nn` 42, 51, 59, 64,
 74, 80, 83, 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, 692, 708, 717, 732, 790, 797,
 827, 862, 872, 876, 898, 915, 928,
 934, 938, 965, 969, 985, 1001, 1007,
 1020, 1085, 1092, 1115, 1126, 1135,
 1152, 1162, 1171, 1198, 1245, 1253,
 1299, 1307, 1311, 1321, 1367, 1380,
 1390, 1427, 1505, 1515, 1541, 1581,
 1615, 1627, 1637, 1645, 1685, 1743,
 1754, 1785, 1791, 1795, 1809, 1816,
 1858, 1883, 1899, 1923, 1934, 1946,
 1973, 2003, 2077, 2093, 2116, 2120,
 2124, 2139, 2143, 2149, 2160, 2168,
 2198, 2220, 2226, 2244, 2267, 2276,
 2292, 2299, 2312, 2325, 2340, 2348,
 2357, 2378, 2400, 2421, 2443, 2447,
 2455, 2467, 2552, 2556, 2560, 2569,
 2577, 2583, 2601, 2608, 2620, 2634,
 2641, 2654, 2668, 2685, 2694, 2705,
 2710, 2719, 2730, 2735, 2759, 2777,
 2791, 2799, 2840, 2849, 2861, 2865,
 2877, 2885, 2893, 2908, 2921, 2967,
 2973, 2994, 3037, 3070, 3137, 3156,
 3176, 3182, 3187, 3214, 3245, 3270,
 3296, 3346, 3390, 3457, 3540, 3565,
 3575, 3670, 3678, 3739, 3751, 3770,
 3812, 3820, 3874, 3883, 3965, 3974,
 3979, 3986, 4007, 4024, 4037, 4048,
 4050, 4063, 4074, 4166, 4182, 4233,
 4245, 4272, 4293, 4298, 4319, 4330,
 4339, 4449, 4480, 4522, 4551, 4558,
 4579, 4593, 4629, 4634, 4639, 4670,
 4743, 4787, 4801, 4814, 4864, 4882,
 4895, 4901, 4911, 4919, 4982, 4996,
 5009, 5010, 5012, 5056, 5089, 5108,
 5120, 5132, 5161, 5183, 5198, 5217,
 5242, 5247, 5252, 5319, 5335, 5341,
 5376, 5382, 5390, 5445, 5506, 5515,
 5531, 5544, 5578, 5589, 5610, 5642,
 5674, 5720, 5723, 5733, 5746, 5759,
 5782, 5783, 5784, 5788, 5814, 5817,
 5843, 5853, 5941, 5993, 6014, 6108,
 6133, 6140, 6157, 6163, 6212, 6272,
 6280, 6287, 6292, 6346, 6368, 6375,
 6379, 6442, 6467, 6493, 6525, 6531,
 6550, 6562, 6573, 6591, 6601, 6641,
 6650, 6658, 6667, 6677, 6688, 6702,
 6726, 6808, 6860, 6906, 6919, 6957,
 7028, 7044, 7120, 7208, 7215, 7219,
 7230, 7245, 7362, 7385, 7403, 7676,
 7772, 7843, 7855, 7894, 7932, 7956,
 7961, 7967, 7976, 8048, 8172, 8363,
 8387, 8444, 8493, 8542, 8616, 8680
`\cs_new_protected:Npn` 205, 337, 340, 344,
 345, 428, 688, 891, 894, 1241, 1325,
 1394, 1410, 1496, 1498, 1829, 1835,
 1839, 1854, 2212, 2295, 2371, 2854,
 2914, 2952, 2980, 3011, 3025, 3108,
 3123, 3257, 3583, 4418, 4564, 4571,
 4621, 4677, 4686, 4711, 5062, 5075,
 5140, 5146, 5150, 5180, 5230, 5271,
 5297, 5422, 5558, 5872, 5892, 5899,
 6029, 6033, 6037, 6043, 6047, 6051,
 6055, 6059, 6063, 6067, 6071, 6075,
 6313, 6335, 6340, 6431, 6450, 6463,
 6515, 6762, 6799, 7109, 7133, 7150,
 7162, 7205, 7568, 8369, 8438, 8441
`\cs_parameter_spec:N` 547, 561
`\cs_prefix_spec:N` 559
`\cs_set:Npn` 448, 483, 488,
 636, 637, 819, 820, 821, 822, 943,
 950, 1094, 1097, 1110, 2174, 2423,
 2456, 2670, 2672, 2675, 2688, 2713,
 2738, 2752, 3797, 4010, 4203, 4204,
 4208, 4209, 4210, 4322, 4352, 4582,
 4598, 4692, 4717, 4846, 4884, 5039,

| | |
|-----------------|--|
| \definiens | 42, 52, 59, 89–91, 6930, 7007, 7025, 7026 |
| \defnotation | 31, 40, 90, 6913, 6943, 6979, 6982 |
| \detokenize | 246, 351, 355, 1658, 8251, 8254, 8257, 8260, 8722, 8725, 8728, 8731 |
| \dimexpr | 8084 |
| do commands: | |
| _do_comp:nNn | 4358, 6014, 6030, 6034, 6038 |
| \dobracket | 82 |
| \dobrackets | 82, 5847, 5900 |
| \dowithbrackets | 5899 |
| \due | 106 |
| \duration | 106 |
| E | |
| \edef | 110, 2081, 2082, 2083, 6527, 7950 |
| \egroup | 3810, 4861, 5054, 7964, 8046, 8063, 8138, 8141, 8425, 8482, 8531, 8580 |
| \eject | 8699 |
| \ellipses | 85, 86, 4696, 4697, 5090, 5092, 5650, 5688 |
| \else | 295, 307, 1887, 1970, 2208, 5877, 6056, 8192, 8410, 8424, 8686, 8900, 8905, 8910 |
| \emph | 12, 14, 92, 6064, 7192, 7280, 7297, 8430, 8487, 8536, 8585, 8652 |
| \end | 126, 131, 1308, 1362, 2044, 2057, 2075, 2235, 2257, 2280, 2540, 2795, 2873, 4735, 6883, 7190, 7216, 7278, 7295, 7346, 7348, 7426, 7430, 7632, 7816, 7870, 7879, 7952, 7964, 7978, 8018, 8032, 8040, 8053, 8055, 8068, 8069, 8070, 8071, 8072, 8073, 8094, 8155, 8189, 8190, 8338, 8417, 8422, 8474, 8479, 8523, 8528, 8572, 8577, 8610, 8612, 8699, 8783, 8825, 8922 |
| \endcsname | 267, 295, 296, 355, 2079, 2080, 2081, 2082, 2083, 2088, 2089, 2090, 2393, 2656 |
| \endgroup | 360, 362, 1931 |
| \endinput | 1766, 2282 |
| \ensuremath | 5762, 5773 |
| environments: | |
| assignment | 106, 8737 |
| blindfragment | 72 |
| exnote | 1, 8493 |
| extstructure | 86, 6172 |
| extstructure* | 86 |
| frame | 1, 7894 |
| gnote | 1, 8542 |

hint 1, 8444
 mathstructure 86, 6086
 mcb 1, 8601
 mminterface 7598
 nassertion 1
 ndefinition 1
 nexample 1
 note 1
 nparagraph 1, 1
 nsproof 1
 problem 1
 sassertion 89
 sdefinition 89
 sexample 89
 sfragment 72
 smodule 76, 2512
 solution 1, 8385
 sparagraph 89
 sproblem 8288
 sproof 91, 7208
 stex_annotation_env 131
 subproof 7301
 testheading 106
 \eq 27, 37
 \eqstep 7263, 7367
 \equal 26
 \escapechar 52, 1054
 \everyeof 2161
 \excursion 99, 8150
 \excursiongroup 99, 8167
 \excursionref 99, 8151, 8164
 exnote (env.) 1, 8493
 exp commands:
 \exp_after:wN 355, 444,
 447, 479, 546, 697, 886, 2163, 2207,
 2208, 2209, 2349, 2394, 2395, 2396,
 2403, 2407, 2411, 2412, 2934, 2955,
 3568, 3800, 4099, 4112, 4700, 4728,
 4849, 5042, 5279, 5281, 5289, 5291,
 5337, 5344, 5345, 5346, 5365, 5398,
 5561, 5563, 5665, 5666, 5667, 5680,
 5875, 5938, 5984, 6000, 6216, 6266,
 6269, 6593, 6889, 6902, 6966, 7001
 \exp_args:N 534, 539, 553, 554,
 885, 939, 1027, 1100, 1127, 1649,
 1798, 1811, 1825, 1841, 1844, 1847,
 1849, 2343, 2837, 2843, 2933, 2936,
 3365, 3393, 3399, 3762, 3771, 3928,
 3992, 3994, 4671, 4816, 5013, 5448,
 5594, 5596, 5605, 6177, 6227, 6314,
 6318, 6396, 6477, 6526, 6532, 6766,
 6802, 6895, 7180, 7387, 7677, 8820
 \exp_args:NNe
 ... 53, 197, 200, 594, 1072, 1077,
 1083, 2457, 2896, 3544, 4031, 4217,
 4315, 5480, 5483, 5567, 5614, 5699,
 6497, 6605, 6660, 6728, 8184, 8364
 \exp_args:Nne 206,
 546, 2025, 2096, 2481, 2521, 2578,
 2779, 2857, 4173, 4324, 4363, 4471,
 4490, 4779, 4974, 6115, 6552, 6593,
 6661, 6866, 7231, 7608, 8308, 8819
 \exp_args:NNNo 831, 1175, 3552, 4552, 4554, 4788, 4983
 \exp_args:NNno 266, 683, 831
 \exp_args:Nnno 1725, 1733
 \exp_args:NNNx 1175, 1714
 \exp_args:NNnx
 ... 836, 3072, 4788, 4983, 5459, 5465
 \exp_args:NNo 37,
 47, 65, 68, 95, 99, 164, 172, 695,
 804, 836, 840, 845, 850, 1015, 1178,
 1656, 1987, 1997, 3205, 3231, 3549,
 3555, 3559, 4265, 4278, 4610, 5693
 \exp_args:Nno 251, 253, 266,
 378, 2424, 2959, 2962, 3795, 4031,
 4844, 5037, 5143, 5148, 5232, 6023,
 6486, 7029, 8293, 8381, 8752, 8792
 \exp_args:NNx 54, 109,
 121, 910, 997, 1896, 1944, 2125,
 2144, 6619, 7081, 7409, 8251, 8254,
 8257, 8260, 8722, 8725, 8728, 8731
 \exp_args:Nnx
 ... 246, 1316, 1590, 1658, 1896,
 1944, 2812, 3013, 3018, 3205, 3231,
 3752, 3919, 5121, 5322, 5354, 5364,
 5874, 6147, 6823, 7081, 7409, 7646
 \exp_args:No 141, 161, 162,
 163, 210, 301, 439, 606, 612, 750,
 1025, 1111, 1211, 1263, 1271, 1293,
 1316, 1529, 1714, 1719, 1724, 1756,
 1761, 1763, 2071, 2364, 2461, 2516,
 2553, 2882, 2894, 2931, 3118, 3133,
 3375, 3376, 3566, 3571, 3655, 3757,
 3758, 3759, 4029, 4185, 4543, 4718,
 4794, 4795, 4796, 4807, 4808, 4809,
 4989, 4990, 4991, 5002, 5003, 5004,
 5096, 5165, 5168, 5172, 5213, 5352,
 5357, 5362, 5367, 5369, 5457, 5485,
 5594, 5596, 5623, 5708, 6125, 6135,
 6167, 6240, 6352, 6356, 6363, 6390,
 6425, 6427, 6499, 6501, 6507, 6582,
 6608, 6614, 6616, 6621, 6625, 6631,
 6678, 6727, 6970, 7020, 7767, 7826,
 7827, 7830, 8250, 8305, 8318, 8721
 \exp_args:Nx
 ... 248, 696, 751, 759, 1533, 1639,
 1641, 2213, 2518, 2997, 7895, 8617

| | | |
|-----------------------|--|--|
| \exp_not:N | 55, 210, 212, 560, 945, 952, 1110, 1221, 1224, 1949, 2006, 2161, 2207, 2417, 2813, 3368, 3402, 4184, 4560, 5124, 5127, 5163, 5164, 5167, 5171, 5175, 5332, 5407, 5413, 5615, 5619, 5622, 5700, 5704, 5706, 5876, 6234, 6426, 6500, 6506, 6542, 6581, 6607, 6613, 6622, 6633, 7778, 7783 | |
| \exp_not:n | 210, 543, 1111, 1316, 1529, 2411, 2438, 2461, 2591, 2615, 3374, 3375, 3376, 3757, 3758, 3759, 3948, 4185, 4226, 4227, 4543, 4700, 4728, 4794, 4795, 4796, 4807, 4808, 4809, 4989, 4990, 4991, 5002, 5003, 5004, 5118, 5165, 5168, 5172, 5174, 5176, 5213, 5345, 5352, 5357, 5362, 5367, 5369, 5581, 5623, 5660, 5665, 5708, 6210, 6352, 6390, 6420, 6425, 6427, 6499, 6501, 6503, 6507, 6510, 6536, 6582, 6584, 6608, 6610, 6614, 6616, 6621, 6625, 6631 | |
| \expandafter | 296, 360, 2079, 2080, 2081, 2082, 2083, 2393, 2656, 6056, 7870, 7871 | |
| \ExplSyntaxOff | 2130, 2570, 7684, 8264, 8735 | |
| \ExplSyntaxOn | 2129, 2566, 7681, 8247, 8718 | |
| \extref | 75, 1553 | |
| \extstructure (env.) | 86, 6172 | |
| \extstructure | 6204, 6206 | |
| \extstructure* (env.) | 86 | |
| F | | |
| \fbox | 8685, 8687 | |
| \fi | 298, 313, 1891, 1906, 1914, 1970, 2209, 5185, 5219, 5890, 5994, 6056, 6958, 7531, 7872, 7940, 7943, 8042, 8192, 8412, 8426, 8660, 8694, 8904, 8914, 8915 | |
| fibboxed | 95 | |
| file commands: | | |
| \file_if_exist:nTF | 634, 651, 1164 | |
| \filepath | 128, 129 | |
| \fillinsol | 104, 8333, 8671 | |
| \first | 127 | |
| \fn | 49 | |
| \foo | 10, 49, 78, 131, 133 | |
| \fooname | 10 | |
| \footnotesize | 8821 | |
| \foral | 41, 46 | |
| \forall | 41, 7518, 7524, 7535 | |
| frame (env.) | 1, 7894 | |
| \frameimage | 98, 8077 | |
| frameimages | 95 | |
| \frametitle | 7994 | |
| G | | |
| \gdef | 8150 | |
| \given | 106 | |
| \global | 1243, 1247, 1264, 1269, 2080, 7564, 7565, 8439, 8442, 8759, 8761 | |
| gnote (env.) | 1, 8542 | |
| gnotes | 100, 106 | |
| group commands: | | |
| \group_begin: | 42, 51, 635, 1053, 1692, 2172, 2431, 2468, 2787, 2807, 2831, 3062, 3286, 3327, 3672, 3724, 3886, 3888, 3890, 3892, 3952, 4158, 4465, 4511, 4773, 4928, 4968, 5097, 5101, 5199, 5352, 5508, 5516, 5590, 5611, 5659, 5675, 5878, 5915, 5923, 5934, 5956, 5967, 5980, 6517, 6534, 6890, 6959, 7008, 7061, 7275, 7342, 7364, 7427, 7448, 7476, 8378, 8799 | |
| \group_end: | 46, 54, 641, 1057, 1739, 2195, 2449, 2482, 2796, 2812, 2874, 3067, 3292, 3333, 3674, 3732, 3886, 3888, 3890, 3892, 3957, 3977, 4161, 4476, 4514, 4784, 4940, 4979, 5068, 5086, 5098, 5102, 5143, 5276, 5287, 5304, 5315, 5337, 5365, 5370, 5386, 5419, 5511, 5528, 5590, 5614, 5667, 5699, 5886, 6011, 6434, 6490, 6520, 6543, 6553, 6619, 6899, 6976, 7022, 7074, 7286, 7352, 7356, 7405, 7442, 7467, 7566, 8382, 8793 | |
| \group_insert_after:N | 256, 272 | |
| H | | |
| \have | 7367 | |
| \hbox | 1255, 3058, 3361, 3786, 3940, 3969, 4340, 4816, 5028, 6188, 6772, 7196, 7220, 8138, 8141, 8640, 8674 | |
| hbox commands: | | |
| \hbox_Set:Nn | 3671, 4201 | |
| \hbox_unpack:N | 3968, 3975 | |
| \HCode | 308 | |
| \hfil | 7202 | |
| \hfill | 7202 | |
| hint (env.) | 1, 8444 | |
| hints | 100, 106 | |
| \hline | 8819, 8821, 8822, 8823, 8824 | |
| \href | 1836 | |
| \hrule | 7196, 8429, 8434, 8486, 8491, 8535, 8540, 8584, 8589, 8640 | |
| \hspace | 8691 | |

| | | | |
|-------------------------------|------------------------|-------------------------------------|-------------------------------------|
| \HTML | 10, 21 | \inputreffalse | 1881, 1890 |
| \huge | 8689 | \inputreftrue | 1888, 1925 |
| hwexam commands: | | \insertframenumber | 8144, 8146 |
| \l_hwexam_includeassignment_- | | \insertshortauthor | 8137 |
| keys_t1 | 8750, 8751, 8753, 8790 | \insertshortdate | 8148 |
| \hwexam_kw_* | 8717 | \insertshorttitle | 8140 |
| \c_hwexam_multiple_bool | 8710 | \inset | 40 |
| \hwexamheader | 8826, 8865 | int commands: | |
| \hwexamminutes | 8828 | \int_case:nn | 1368 |
| \hyperlink | 1830, 1831 | \int_case:nnTF | 1326, 1395, 1428 |
| \hypertarget | 1810, 1811 | \int_compare:nNnTF | 248, 2023, 2095, 3545, 4190, 4225, |
| | | 4724, 5306, 5593, 5604, 5653, 5859, | |
| | | 6349, 6359, 6451, 6834, 7142, 8806 | |
| | | \int_compare_p:nNn | 7112, 7124, 7136, 7153, 7165 |
| | | \int_decr:N | 7143, 7171 |
| | | \int_eval:n | 265, 267, 269, 271, 273, |
| | | 5462, 5468, 5858, 6148, 6155, 6282, | |
| | | 8759, 8802, 8803, 8820, 8853, 8860 | |
| | | \int_gincr:N | 1507, |
| | | 5447, 8391, 8448, 8497, 8546, 8800 | |
| | | \int_gset:Nn | 5411 |
| | | \int_gzero:N | 5396 |
| | | \int_incr:N | 1318, 1328, |
| | | 1332, 1356, 1370, 1373, 1376, 1397, | |
| | | 1401, 1432, 1438, 3862, 3863, 3864, | |
| | | 3865, 4640, 4723, 5559, 7117, 7129, | |
| | | 7140, 7157, 7169, 7585, 7817, 8807 | |
| | | \int_new:N | 1276, 1503, 3818, 4006, |
| | | 4710, 5389, 5543, 5847, 8385, 8798 | |
| | | \int_set:Nn | 1412, 1413, 1414, 1415, |
| | | 1416, 1417, 1419, 3042, 3827, 3831, | |
| | | 3835, 3839, 3843, 3847, 3851, 3855, | |
| | | 3859, 3934, 4013, 4054, 4086, 4613, | |
| | | 4718, 4725, 4763, 4887, 5519, 5848, | |
| | | 5881, 7110, 7122, 7134, 7151, 7163 | |
| | | \int_step_function:nN | 5125, 5325 |
| | | \int_step_inline:nn | 3876, 4257, 4264, 4284, 4534 |
| | | \int_use:N | 227, 1302, |
| | | 1382, 1422, 1442, 1508, 3372, 3755, | |
| | | 4334, 4451, 4613, 4792, 4805, 4987, | |
| | | 5000, 5235, 5411, 5448, 5580, 5617, | |
| | | 5702, 5845, 5846, 6825, 7588, 7649, | |
| | | 7810, 8393, 8450, 8499, 8548, 8819 | |
| | | \int_zero:N | 3821, 3822, 4595, 4721, 5549, 7578 |
| | | \c_max_int | 5845, 5846 |
| | | \l_tmpa_int | 4595, 4598, 4613, 4640, 4721, 4723, |
| | | 4724, 4725, 4729, 7110, 7113, 7116, | |
| | | 7117, 7122, 7125, 7128, 7129, 7134, | |
| | | 7137, 7140, 7142, 7143, 7145, 7146, | |

| | |
|---|--|
| 7151, 7154, 7157, 7159, 7163, 7166, 7169, 7171, 7172, 7578, 7585, 7588 | \labelsep 7946 |
| intarray commands: | \labelwidth 7947 |
| \intarray_gset:Nnn 7145, 7159, 7172, 7247 | \lambda 7518, 7529, 7530 |
| \intarray_gzero:N 7246 | \langle 7505 |
| \intarray_item:Nn 7113, 7116, 7125, 7128, 7137, 7146, 7154, 7166 | \Large 7755, 8106, 8770 |
| \intarray_new:Nn 7108 | \LaTeX 18 |
| \interpretmod 3621, 3625 | \large 18 |
| \interpretmodule 3511, 3513 | \ldots 7510, 7555, 7557, 7559 |
| invokation commands: | \leaders 8108 |
| \invokation_macro 113, 115, 121 | \leavevmode 8002 |
| ior commands: | \left 5875 |
| \ior_close:N 658, 664, 1058, 1189 | \leftmargin 7948 |
| \ior_map_inline:Nn 1174 | \let 355, 1450, 1451, 1460, 1461, 1722, 2080, 2162, 2173, 2180, 2191, 2203, 3310, 3969, 3976, 4470, 4489, 4778, 4973, 5202, 5920, 5930, 5944, 6006, 6085, 7564, 7565, 7682, 7683, 8115 |
| \ior_new:N 1170 | \libinput 15, 70, 1973 |
| \ior_open:Nn 652, 660, 1172 | \libusepackage 70, 71, 2021, 7886 |
| \ior_open:NnTF 1052 | \libusetheme 7885 |
| \ior_str_get:NN 1055 | \libusetikzlibrary 70, 108, 2077 |
| \ior_str_map_inline:Nn 654, 661 | \LoadClass 15, 7723, 7725 |
| \g_tmpa_ior 652, 654, 658, 660, 661, 664, 1052, 1055, 1058 | \long 131, 587, 8119, 8128 |
| iow commands: | \lstinputlisting 73, 2056 |
| \iow_close:N 647, 657, 1495 | \lstinputmhlisting 73, 2032 |
| \iow_new:N 616, 1493 | |
| \iow_now:Nn 624, 655, 662, 1521, 1792, 7680, 8364 | M |
| \iow_open:Nn 646, 653, 1494 | \macro 125–127, 132 |
| \g_tmpa_iow 653, 655, 657 | \macroname 27 |
| \isassociative 44 | \magma 48 |
| \iscommutative 44 | \maincomp 27, 28, 31, 49, 80, 87, 93, 3976, 4184, 4193, 4204, 4205, 4226, 4357, 4560, 5204, 5385, 5902, 6081, 6526, 6527, 6532, 6540, 6545 |
| \item 7213, 8663 | \mainmatter 1472, 1473, 1483, 1484 |
| \interpretmod 3623 | \makeatletter 2127, 8247, 8718, 8865 |
| \itshape 7206 | \makeatother 2128, 8264, 8735, 8865 |
| | \maketitle 1268, 1269 |
| J | \mapsto 7527, 7528 |
| \jobname 67, 131, 646, 651, 652, 653, 660, 665, 674, 980, 1494, 1714 | \marginnote 7936 |
| \join 58 | \marginpar 8342, 8345, 8593, 8598 |
| | \mathbb 7541 |
| K | \mathbin 28, 7486, 7490, 7497, 7523, 7534 |
| keys commands: | \mathclose 28, 5887, 7488, 7495, 7499, 7505, 7507, 7515, 7522, 7534, 7538 |
| \l_keys_choice_tl 3689 | \mathhub 68, 128, 160, 30, 1041 |
| \keys_define:nn 27, 378, 7695, 7737, 8167, 8216, 8708, 8878 | \mathop 28, 7525, 7536 |
| \l_keys_key_str 4135, 4138, 6100, 6103 | \mathopen 28, 5882, 7488, 7495, 7499, 7505, 7507, 7514, 7522, 7534, 7538 |
| \l_keys_key_tl 4136, 6101 | \mathord 28 |
| \keys_set:nn 382, 8176 | \mathpunct 28, 4538, 5538, 7514, 7524, 7530 |
| keyval commands: | \mathrel 27, 28, 7491, 7528 |
| \keyval_parse>NNn 6371 | \mathrm 4560, 7514 |
| | mathstructure (env.) 86, 6086 |
| L | |
| \label 74, 111, 122, 1533, 7987 | |

\mathstructure 6130, 6131
 \mathtt 7494, 7495, 7544, 7547, 7551
 mcb (env.) 1, 8601
 \mcb 8332, 8605, 8615
 \mcc 62, 102, 8607, 8638
 \meaning
 1023, 2393, 2656, 4230, 4504, 4589,
 4615, 5142, 5147, 5350, 5360, 5361
 \medskip 7963, 7977, 7996
 \meet 58
 \message 26, 7716, 7719, 7876
 \mhframeimage 98
 \mhgraphics 73, 2032, 8090, 8092
 \mhinput 73, 1881
 \mhtikzinput 108, 2032
 \min 106
 \min 8596
 min 100, 106
 \mmlarg 336
 \mmlintent 336
 \mmtdef 7625, 7635
 \MMTinclude 7569
 mmtinterface (env.) 7598
 \MMTrule 7574
 \mname 77, 86
 mode commands:
 \mode_if_math:TF
 .. 3361, 3969, 3970, 3977, 5244, 8673
 \mode_if_vertical:TF 3968, 3975
 \MSC 7568
 msg commands:
 \msg_error: 132
 \msg_error:nn 81, 1781, 7016, 7069
 \msg_error:nnn 84,
 153, 192, 240, 811, 947, 954, 1975,
 1978, 2104, 3197, 3253, 3424, 3454,
 3982, 4673, 4915, 6160, 6637, 6664
 \msg_error:nnnn 351, 829, 834,
 853, 888, 960, 1949, 2006, 2916,
 3004, 3349, 3867, 4443, 4938, 5193,
 5227, 5473, 5500, 6009, 6274, 6974
 \msg_none:nn 78
 \msg_redirect_module:nnn 100
 \msg_redirect_name:nnn 104
 \msg_set:nnn 75
 \msg_warning:nn 1071
 \msg_warning:nnn 1691, 1733
 \msg_warning:nnnn 433, 1725
 \mult 35, 36, 81, 82
 \multicolumn 8820
 \multiple 106

 N
 nassertion (env.) 1

 \Nat 31
 \ndefinition (env.) 1
 \neginfprefc 36, 80, 81, 4235, 4238, 4247,
 4250, 4263, 5064, 5300, 5311, 5845
 \newcommand 444, 479, 1575,
 1956, 1964, 1969, 2014, 2021, 2051,
 2057, 2065, 2075, 2102, 7471, 7867,
 7936, 7998, 8099, 8103, 8151, 8158,
 8161, 8178, 8281, 8643, 8671, 8698,
 8701, 8702, 8805, 8890, 8895, 8919
 \newcounter ... 7726, 7727, 7728, 7729,
 7851, 7863, 8280, 8288, 8289, 8748
 \NewDocumentCommand
 ... 447, 1539, 1771, 1776, 1823,
 1895, 1943, 2486, 2565, 4927, 5435,
 5914, 5922, 5933, 5955, 5966, 5979,
 6889, 6979, 6984, 6990, 6998, 7007,
 7050, 7060, 7078, 7404, 7440, 7455,
 7574, 7635, 7885, 8077, 8377, 8788
 \NewDocumentEnvironment
 ... 482, 1348, 1357, 7460, 7598
 \newenvironment 1469, 1480,
 7174, 7381, 8050, 8052, 8601, 8848
 \newif ... 296, 319, 1881, 7758, 8237, 8855
 \newlabel 7682, 7683
 \newlength 7888, 7889, 7892
 \newpage 7984, 8702, 8867
 \newsavebox 8020
 \nexample (env.) 1
 \ninputref 8097, 8099, 8103
 \nobreak 7202
 \noindent 1303, 1906,
 1914, 6873, 7192, 7963, 7977, 8000,
 8018, 8340, 8430, 8487, 8536, 8585
 \nointerlineskip 8110
 \notation .. 27, 28, 36, 78, 80, 85, 116,
 119, 125, 2984, 4142, 7486, 7489,
 7490, 7491, 7504, 7506, 7524, 7525,
 7529, 7533, 7535, 7536, 7541, 7544
 note (env.) 1
 notes 95, 100, 106
 \notesfalse 7770
 notesslides commands:
 \c_notesslides_notes_bool
 ... 7697, 7698, 7711, 7714, 7722,
 7738, 7739, 7751, 7759, 7882, 8043,
 8049, 8083, 8098, 8117, 8152, 8162
 \c_notesslides_sectocframes_bool
 ... 7740, 7838
 \c_notesslides_topsect_str 7741,
 7764, 7767, 7823, 7826, 7827, 7830
 notesslides internal commands:
 \c__notesslides_class_str
 ... 7693, 7696, 7702, 7723

| | | | |
|---|--|--|--|
| __notesslides_define_chapter: | 7828, 7831, 7843 | \notesslidesfont | 7959, 7974, 8115 |
| __notesslides_define_part: | 7832, 7855 | \notesslidesfooter | 7963, 7977, 8113 |
| __notesslides_do_label:n 7773, 7809 | | \notesslidestitleemph | 7996, 8105 |
| __notesslides_do_sectocframes: | 7772, 7839 | \notestrue | 7760 |
| __notesslides_do_yes_param:Nn | 7894, | nparagraph (env.) | 1, 1 |
| 7913, 7916, 7919, 7922, 7925, 7928 | | nsproof (env.) | 1 |
| \c__notesslides_docopt_str ... | 7699 | \null | 7202 |
| \c__notesslides_document_str ... | | \num | 133 |
| 7866, 7869 | | \number | 106 |
| __notesslides_eat: . 8053, 8057, 8060 | | \numberline | 7808 |
| __notesslides_excursion_args:n | 8172, 8179 | \numberproblemsin | 8280 |
| \l__notesslides_excursion_id_str | 8168, 8174 | | |
| \l__notesslides_excursion_intro_- tl | 8169, 8173, 8183, 8185 | O | |
| \l__notesslides_excursion_- mhrepos_str | 8170, 8175, 8184 | \OMDoc | 21 |
| \l__notesslides_frame_allowdisplaybreaks_- bool | 7905, 7916 | \omdoc | 10 |
| \l__notesslides_frame_allowframebreaks_- bool | 7904, 7913 | \oplus | 7496, 7497 |
| __notesslides_frame_box_begin: | 7956, 7967, 7990 | | |
| __notesslides_frame_box_end: ... | 7961, 7976, 7992 | P | |
| \l__notesslides_frame_fragile_- bool | 7906, 7919 | \PackageError | 8202 |
| \l__notesslides_frame_label_str | 7903, 7911, 7986, 7987 | \pagebreak | 8351, 8786 |
| \l__notesslides_frame_shrink_- bool | 7907, 7922 | \pagenumbering | 1456, 1466, 1477, 1488 |
| \l__notesslides_frame_squeeze_- bool | 7908, 7925 | \par | 132, 359, 1301, 1305, 7175, 7202, 7245, 7302, 7963, 7977, 8000, 8005, 8013, 8018, 8025, 8035, 8340, 8347, 8350, 8429, 8434, 8486, 8491, 8535, 8540, 8584, 8589, 8601, 8644, 8769, 8774, 8784, 8786 |
| __notesslides_inputref: | 8096, 8097, 8100 | \paragraph | 22, 72 |
| __notesslides_notes_env:nnnn ... | 8048, | \parsep | 7211 |
| 8068, 8069, 8070, 8071, 8072, 8073 | | \part | 22, 72, 7859, 7860 |
| \l__notesslides_num | 7776, 7778, 7781, 7783, 7786, 7787, 7790, 7791, 7794, 7795, 7798, 7799, 7802, 7803, 7810 | \partname | 7778, 7856, 7857 |
| __notesslides_setup_itemize: ... | 7932, 7989 | \parttitlename | 7778 |
| \l__notesslides_tmp | 8201, 8206 | \PassOptionsToClass | 7701, 7702 |
| \g__notesslides_variables_prop | 8193, 8195, 8198, 8201 | \PassOptionsToPackage | |
| | | 10, 7703, 7704, 7715, 7718, 7743, 7744, 7761, 8234, 8712 | |
| | | \pause | 7998 |
| | | \PDF | 10, 21 |
| | | \pdfbookmark | 7810 |
| | | \pdfdest | 1535 |
| | | peek commands: | |
| | | \peek_charcode:NTF | |
| | | ... 3576, 5058, 5071, 5134, 5136, 5258, 5265, 6296, 6302, 6443, 6454 | |
| | | \peek_charcode_remove:NTF | |
| | | ... 5057, 5133, 5249, 5254, 5256, 5263, 5336, 5724, 6301, 6551 | |
| | | \phantom | 8689 |
| | | \Pi | 7518 |
| | | \plus | 34–36, 39, 80–82 |
| | | \prematurestop | 98, 7866 |
| | | \premise 46, 47, 91, 6941, 7078, 7085 | |
| | | prg commands: | |
| | | \prg_new_conditional:Nnn | |
| | | ... 231, 281, 533. | |

```

538, 747, 757, 2134, 2544, 2548,
3649, 3660, 3664, 5479, 5592, 5603
\prg_new_protected_conditional:Nnn
..... 767
\prg_return_false: .. 233, 283, 536,
541, 748, 752, 758, 760, 783, 787,
2135, 2546, 2550, 3650, 3665, 5497,
5501, 5598, 5599, 5600, 5606, 5607
\prg_return_true: ..... 233, 283,
536, 541, 750, 752, 760, 787, 2135,
2546, 2550, 3661, 5495, 5598, 5606
\printexcursion ..... 99
\printexcursions .. 8150, 8159, 8180, 8188
problem (env.) ..... 1
problem commands:
\g_problem_id_counter ..... 8385, 8391, 8393,
8448, 8450, 8497, 8499, 8546, 8548
\l_problem_inputproblem_keys_tl .
..... 8291, 8292, 8294, 8379
\problem_mcc_box_tl ..... 8638, 8663
\problemheader ..... 8340, 8355
problems internal commands:
\c__problems_boxed_bool ..... 8230
\__problems_do_yes_param:Nn .. 8616
\__problems_exnote_start:n .....
..... 8493, 8512, 8515
\__problems_fillinsol:nn .....
..... 8674, 8676, 8680
\__problems_gnote_start:n .....
..... 8542, 8561, 8564
\c__problems_gnotes_bool .....
..... 8220, 8563, 8574
\__problems_hint_start:n .....
..... 8444, 8463, 8466
\c__problems_hints_bool .....
..... 8222, 8465, 8476
\c__problems_min_bool 8228, 8344, 8597
\c__problems_notes_bool .....
..... 8218, 8514, 8525
\l__problems_path_seq .. 8301, 8302
\l__problems_prob_imports_tl . 8277
\l__problems_prob_refnum_int . 8278
\c__problems_pts_bool 8226, 8341, 8592
\__problems_record_problem: .....
..... 8329, 8363
\__problems_solution_start:n ...
..... 8387, 8406, 8409
\c__problems_solutions_bool .....
..... 8224, 8240
\c__problems_test_bool .....
..... 8232, 8351, 8699, 8701, 8702
\ProcessKeysOptions .....
.... 38, 7707, 7747, 8239, 8715, 8884
\ProcessOptions ..... 11
\prod ..... 7525, 7536
\prop ..... 37
prop commands:
\prop_clear:N .. 1173, 2878, 2879, 6642
\prop_const_from_keyval:Nn .. 109, 121
\prop_gclear:N ..... 2317, 2318, 2319, 2429, 5394
\prop_get:Nnn ..... 1207
\prop_get:NnTF ..... 188, 902, 903, 1797, 1901,
1902, 1977, 2932, 2953, 5480, 8201
\prop_gput:Nnn ..... 2457, 2578, 2590, 2592, 2625, 5423,
5459, 5465, 5483, 7046, 8195, 8801
\prop_gset_eq:NN .. 1190, 1221, 1224
\prop_gset_from_keyval:Nn .....
..... 1193, 1218, 2422, 2424, 5407
\prop_if_exist:NTF ..... 900, 1099, 1116, 1600,
1694, 1796, 1861, 1974, 3077, 5406
\prop_if_in:NnTF ..... 137, 172, 804, 6602, 6659, 6689, 6703
\prop_item:Nn ..... 138, 920, 1100, 1213, 1618,
1697, 1863, 1866, 1875, 2039, 2053,
2068, 2858, 3014, 3019, 3080, 3105,
3116, 3131, 6594, 6634, 6662, 8198
\prop_map_break: ..... 2618
\prop_map_break:n 2637, 2673, 2676,
2754, 3038, 4051, 4903, 4906, 7033
\prop_map_function:NN ... 2415, 2945
\prop_map_inline:Nn .. 2425, 2602,
2635, 2679, 2697, 2700, 2722, 2725,
2763, 2909, 2928, 2968, 2996, 3000,
4067, 4896, 6256, 6482, 7030, 8813
\prop_new:N ..... 8193, 8795
\prop_put:Nnn .. 1181, 1182, 1183,
1184, 1185, 2896, 2900, 3368, 3402,
4789, 4984, 6668, 6691, 6705, 8808
\prop_remove:Nn ..... 3464
\prop_set_eq:NN 1088, 1205, 1209, 1609
\prop_to_keyval:N ..... 1191, 1194, 1219, 2394, 2395,
2396, 2403, 2407, 2923, 2926, 5408
\protect ..... 7808, 8204
\protected ..... 113,
131, 357, 571, 572, 1269, 5931, 5945
\providecommand .. 2037, 2044, 7874, 8889
\ProvidesExplClass ..... 5, 7690
\ProvidesExplPackage ..... 19, 7734, 8213, 8705, 8875
\pts ..... 8591
pts ..... 100, 106

```

Q

\quad 7787, 7791, 7795,
 7799, 7803, 8672, 8678, 8777, 8780
 quark commands:
 \quark_new:N 2158, 2441
 quark internal commands:
 \q__stex_smsmode_break
 2158, 2161, 2221

R

\rangleangle 7505
 \realization 3535, 3537
 \realize 58, 59, 3642, 3644, 3646
 \ref 74, 1641
 \refstepcounter 8299, 8761
 \relax 358, 639, 640, 1451, 1461,
 1535, 1692, 2079, 2165, 2203, 2207,
 2432, 2433, 3969, 3976, 8084, 8086
 \renamedecl 59, 119, 2990, 3388, 3472
 \renewcommand 7883, 7984, 7994
 \renewenvironment
 .. 7937, 7981, 7999, 8021, 8044, 8046
 repo commands:
 \repo_prop 128, 129
 \reqpts 106
 \requiremodule 83, 84, 2986, 3337
 \RequirePackage
 .. 4, 13, 18, 23, 24, 146, 148, 197,
 200, 5871, 7691, 7709, 7731, 7735,
 7749, 7762, 7763, 8075, 8214, 8245,
 8706, 8716, 8876, 8887, 8892, 8893
 \resizebox 8818, 8901, 8907, 8911
 \rhd 7939, 7942
 \right 5876
 \Rightarrow 7437
 \rightmargin 7212
 rustex commands:
 \rustex_direct_HTML:n
 8006, 8014, 8026, 8036
 \rustex_if:TF 8004, 8005,
 8012, 8013, 8024, 8025, 8034, 8035
 \rustexBREAK 2131

S

sassertion (env.) 89
 \scriptsize 7936
 \scriptstyle 7942
 sdefinition (env.) 89
 \second 127
 \section 21, 22, 72
 \sectiontitleemph 7754, 7813
 sectocframes 95
 \selectlanguage 130, 141

seq commands:

\seq_clear:N 182, 682, 720,
 1073, 1980, 2175, 2686, 2711, 2736,
 2841, 2880, 4188, 4533, 4694, 5548,
 5612, 5677, 5686, 6122, 6173, 6223,
 6381, 6643, 6644, 6764, 6800, 7575
 \seq_count:N .. 2023, 2095, 3545, 6834
 \seq_gclear:N 2169, 2170, 5395
 \seq_gclear_new:N 983, 984
 \seq_get:NN 1014
 \seq_get_right:NN 158, 8302
 \seq_gpop>NN 1009
 \seq_gpush:Nn 997
 \seq_gput_left:Nn 1544, 1549
 \seq_gput_right:Nn
 1547, 2125, 2144, 5424
 \seq_gset_eq:NN
 996, 1012, 1016, 2868, 2869, 2870, 7672
 \seq_gset_split:Nnn 5413
 \seq_if_empty:NTF .. 169, 718, 736,
 748, 758, 787, 833, 839, 844, 849,
 1008, 1011, 1117, 1948, 2005, 4028,
 6849, 7182, 7233, 7389, 7583, 7671
 \seq_if_empty_p:N 774, 775, 1984
 \seq_if_exist:NTF 1589, 5412
 \seq_if_in:NnTF .. 1542, 1543, 1656,
 1658, 1987, 1997, 2269, 2278, 2655,
 2695, 2720, 2761, 6494, 6605, 6728
 \seq_item:Nn 749,
 759, 2024, 2096, 4301, 4307, 5481, 6835
 \seq_map_break: 1591, 1595, 2673
 \seq_map_break:n 1595, 2676, 4051
 \seq_map_function:NN .. 722, 726, 6393
 \seq_map_inline:Nn
 264, 917, 921, 1594,
 1629, 1951, 2008, 2184, 2678, 4065,
 4695, 4722, 5687, 6186, 6233, 6242,
 6770, 6908, 6935, 7253, 7308, 7584
 \seq_new:N
 ... 225, 1502, 1548, 2115, 2138, 2288
 \seq_pop>NN 719
 \seq_pop_left:NN 168, 702,
 780, 781, 832, 837, 843, 848, 918,
 922, 924, 1991, 3546, 3548, 3554, 3558
 \seq_pop_left:NNTF
 1177, 4275, 4277, 4285
 \seq_pop_right:NN 160, 170,
 737, 792, 794, 799, 801, 803, 1145,
 2363, 3073, 4027, 4553, 4555, 6384
 \seq_push:Nn 725
 \seq_put_left:Nn
 ... 703, 723, 2696, 2721, 6690, 6704
 \seq_put_right:Nn
 164, 190, 228, 740, 795,

| | | | |
|-----------------------------|---|--|------------------------|
| \seq | 805, 808, 980, 1988, 1992, 1998, 2322, 2352, 2365, 2657, 2762, 2844, 2975, 4258, 4265, 4286, 4288, 4535, 4697, 4699, 5579, 5615, 5651, 5658, 5683, 5689, 5693, 5700, 6123, 6178, 6182, 6228, 6660, 6767, 6803, 6817 | | |
| \seq_reverse:N | 6385 | \smacro | 82, 83 |
| \seq_set_eq:NN | 728, 768, 769, 791, 798, 916, 979, 1981, 4704, 4732, 5698 | \small | 7206 |
| \seq_set_split:Nnn | 159, 683, 793, 800, 831, 836, 1137, 1176, 1982, 2362, 3072, 3544, 3552, 4026, 4274, 4278, 4552, 4554, 6383, 7576 | \smallskip | |
| \seq_use:Nn | 195, 198, 201, 765, 795, 808, 836, 1179, 2367, 3074, 3550, 3556, 3560, 4032, 4344, 4538, 4681, 4705, 5414, 5537, 6850, 7183, 7234, 7390 | smodule (env.) | 76, 2512 |
| \l_tmpa_seq | 159, 160, 164, 168, 169, 170, 182, 190, 195, 198, 201, 4533, 4535, 4538, 4552, 4553, 4554, 4555, 4694, 4697, 4699, 4704, 4732, 6381, 6383, 6384, 6385, 6393, 7575, 7576, 7583, 7584 | \smodule | 2987 |
| \seqmap | 86, 5150, 5605 | \Sn | 15, 79, 5902 |
| \setbox | 125, 8046, 8061, 8411, 8468, 8517, 8566 | \sn | 15, 19, 79, 5902 |
| \setcounter | 8759, 8762 | \Sns | 15, 79, 5902 |
| \setkeys | 2041, 2055, 2070, 8088, 8896 | \sns | 15, 79, 5902 |
| \setlength | 7210, 7211, 7212, 7888, 7889, 7893, 7946, 7947, 7948 | solution (env.) | 1, 8385 |
| \setlicensing | 97 | \solution | 8331, 8437, 8606 |
| \setmetatheory | 2467 | solutions | 100, 106 |
| \setnotation | 28, 81, 4418 | \solutionsfalse | 8243, 8442 |
| \setsectionlevel | 22, 72, 111, 1410, 7765, 7767, 7824, 7826 | \solutionstrue | 8241, 8439 |
| \setSGvar | 99, 8194, 8204 | \source | 127 |
| \setslidelogo | 97 | sparagraph (env.) | 89 |
| \setsource | 97 | \spfblock | 7265, 7465 |
| \sexample (env.) | 89 | \spfjust | 7266, 7471, 7474 |
| \sf | 8106 | \spfsketch | 7174 |
| \sffamily | 8115 | \spfsketchenvautorefname | 7192 |
| \sfragment (env.) | 72 | \spfstep | 7260, 7367 |
| \sfragment commands: | | \spfstepautorefname | 7301, 7382 |
| __sfragment_do_level:nn | 1299, 1311, 1327, 1331, 1335, 1336, 1337, 1338, 1339, 7806 | \spfstepenvautorefname | 7382 |
| __sfragment_end: | 1307, 1321, 1353 | sproblem (env.) | 8288 |
| \skipfragment | 72, 1394, 1398, 1402 | \sproblemautorefname | 8356 |
| \slideframewidth | 7892, 7893, 7969, 8084 | sproof (env.) | 91, 7208 |
| \slideheight | 7889 | \sproofautorefname | 7280 |
| \slides | 95 | \sproofend | 7194, 7282, 7299 |
| \slidewidth | | \square | 7195, 8639 |
| | 7888, 7972, 8002, 8084, 8086, 8090 | \sr | 15, 19, 78, 5902 |
| | | \sref | 74, 75, 89, 1553, 8154 |
| | | \sreflabel | 74, 76, 111, 1502 |
| | | \srefsetin | 74, 1575 |
| | | \srefsym | 79, 1823 |
| | | \srefsymuri | 79, 1854 |
| | | \startsolutions | 101, 8438 |
| | | \stepcounter | 1391, 7807, 7983 |
| | | \sTeX | 9, 10, 71 |
| | | \stex | 10, 19, 71 |
| | | stex commands: | |
| | | \l_stex_aB_args_seq | |
| | | 4681, 4695, 4704, 4705, 4722, 4732, 5537, 5548, 5579, 5612, 5615, 5651, 5658, 5677, 5683, 5687, 5698, 5700 | |
| | | \stex_activate_module:n | |
| | | 112, 2333, 2382, 2654, 2654, 2666, 2836, 3179, 3277, 3320, 3520, 6194, 6250, 6775, 7621 | |
| | | \stex_add_definiens:nn | |
| | | 2992, 7020, 7028, 7255, 7310 | |
| | | \stex_add_definiens_inner:nnnnnnnn | |
| | | 7033, 7044 | |

```

\stex_add_module_notation:nnnnn  112
\l_stex_all_modules_seq . . . . .
    . . . . . 112, 2175, 2288,
    2322, 2352, 2655, 2657, 2678, 4065
\l_stex_allow_semantic_bool . . .
    4934, 5076, 5154, 5155, 5158, 5184,
    5218, 5384, 5393, 5509, 5520, 5618,
    5703, 5761, 5772, 5805, 5834, 5995,
    6076, 6487, 6518, 6535, 6580, 6961
\stex_annotate:nn . . . . . 338,
    341, 1282, 1291, 1303, 1745, 1965,
    1966, 3361, 3789, 3792, 3799, 3804,
    3806, 4347, 4353, 4362, 4373, 4384,
    4385, 4388, 4389, 4393, 4838, 4841,
    4848, 4852, 4855, 5031, 5034, 5041,
    5045, 5048, 5079, 5525, 5532, 5626,
    5735, 5748, 5765, 5790, 5798, 5819,
    5827, 6021, 6407, 6415, 6420, 6628,
    6874, 6895, 6972, 7038, 7071, 7083,
    7180, 7222, 7225, 7267, 7321, 7330,
    7334, 7387, 7456, 7472, 7581, 7586,
    8144, 8593, 8598, 8658, 8665, 8682
\stex_annotate:nnn . . . . . 130, 131
\stex_annotate_force_break:n . . .
    . . . . . 327, 328, 334,
    1304, 3360, 3786, 3804, 4835, 4853,
    5029, 5046, 5533, 5631, 5740, 5753,
    5770, 5795, 5803, 5824, 5832, 6419,
    6874, 6896, 7039, 7072, 7582, 8682
\stex_annotate_invisible:n . . .
    . . . . . 131, 329,
    331, 1361, 1383, 2529, 2786, 2830,
    5181, 5437, 7220, 7579, 7616, 8316
\stex_annotate_invisible:nn . . .
    . . . . . 1255, 1422, 1430,
    1436, 1442, 1906, 1914, 3058, 3282,
    3323, 3359, 3393, 3434, 3444, 3771,
    4340, 4816, 5013, 5771, 5804, 5833,
    6188, 6244, 6772, 7054, 7570, 7617
\stex_annotate_invisible:nnn . . 131
\l_stex_argnames_seq . . . . . 114
\stex_args_end: . . . . . 5398,
    5422, 5425, 5947, 5951, 6001, 6967
\stex_assign_do:n . . . . .
    . . . . . 2851, 3341, 3346, 3571
\l_stex_assoc_args_count . . . .
    . . . . . 114, 3818, 3822, 3864, 3865
\l_stex_brackets_dones_bool . . .
    . . . . . 4602, 5851, 5854, 5855, 5879, 5880
\stex_capitalize:n . . . . .
    . . . . . 1293, 5938, 5941, 5984, 7001
\stex_check_term:n . . . . .
    . . . . . 114, 115, 2174, 3353,
    3669, 3670, 3678, 3884, 4320, 7478
\c_stex_check_terms_bool . . . . .
    . . . . . 33, 3656, 3659, 7481
\stex_close_module: 112, 2389, 2389,
    2538, 2793, 7562, 7629, 7631, 8336
\l_stex_current_archive . . . . . 1085
\l_stex_current_archive_prop . . .
    . . . . . 124, 129, 920, 930,
    935, 1088, 1099, 1100, 1109, 1198,
    1600, 1601, 1694, 1697, 1796, 1797,
    1861, 1863, 1866, 1901, 1902, 1974,
    1977, 2039, 2053, 2068, 3077, 3080
\l_stex_current_args_tl . . . . .
    . . . . . 121, 4524, 4528, 5207, 5397, 5398, 5401
\l_stex_current_arity_str . . . .
    . . . . . 121, 4534, 5096, 5123,
    5125, 5206, 5306, 5324, 5325, 5356
\l_stex_current_doc_uri 129, 964,
    966, 967, 1512, 1517, 1519, 1523,
    1589, 1591, 1631, 1648, 1649, 6853
\l_stex_current_domain_str . . . .
    . . . . . 119, 2800, 2803, 2805,
    2813, 2818, 2827, 2836, 2867, 2882,
    2894, 2931, 2943, 3411, 3459, 3480,
    3500, 3520, 3524, 3594, 3614, 3635
\g_stex_current_file . . . . .
    . . . . . 127–129, 158, 929, 931,
    936, 984, 987, 989, 991, 992, 993,
    996, 997, 1012, 1015, 1016, 1604,
    1702, 2341, 3087, 3093, 3094, 3142,
    3161, 3205, 3231, 7671, 7672, 8301
\l_stex_current_language_str . . .
    . . . . . 129, 130, 134, 136, 171, 176, 2523,
    3192, 3195, 3219, 3222, 7610, 8310
\l_stex_current_module . . . . . 113, 115
\l_stex_current_module_str . . .
    . . . . . 112, 133,
    2176, 2287, 2304, 2321, 2322, 2337,
    2350, 2351, 2352, 2353, 2392, 2393,
    2394, 2395, 2396, 2402, 2404, 2408,
    2413, 2415, 2522, 2532, 2545, 2553,
    2557, 2578, 2584, 2590, 2592, 2602,
    2609, 2614, 2623, 2625, 2635, 2659,
    2662, 2781, 2792, 2826, 2837, 2934,
    2937, 3366, 3400, 3427, 3436, 3446,
    3725, 3763, 3772, 3868, 3929, 3938,
    3947, 3953, 4481, 4500, 4501, 6079,
    6117, 6122, 6123, 6143, 6235, 6257,
    6818, 6841, 7029, 7030, 7031, 7032,
    7045, 7046, 7600, 7601, 7604, 7609,
    7628, 7630, 7643, 7657, 8309, 8322
\l_stex_current_ns_uri . . . . . 129,
    970, 971, 2359, 2502, 3143, 3148,
    3149, 3162, 3167, 3168, 7482, 7563
\l_stex_current_redo_tl . . . . .

```

..... 5188, 5201, 5212, 5215,
 5222, 5352, 6352, 6390, 6425, 6575
`\l_stex_current_return_tl`
 121, 3976, 5208, 5274, 5286, 5307, 5357
`\stex_current_section_level`
 1278, 1284, 1293, 1319, 7818
`\l_stex_current_symbol_str`
 121, 4172, 4202,
 4321, 4361, 4372, 4470, 4489, 4552,
 4778, 4930, 4932, 4938, 4973, 5063,
 5066, 5081, 5109, 5110, 5193, 5205,
 5227, 5243, 5272, 5273, 5276, 5282,
 5298, 5299, 5302, 5391, 5473, 5500,
 5517, 5619, 5620, 5663, 5704, 5705,
 5737, 5750, 5767, 5792, 5800, 5821,
 5829, 5882, 5887, 5960, 5973, 5986,
 5996, 6009, 6017, 6021, 6023, 6078,
 6433, 6434, 6962, 6970, 6972, 6974
`\l_stex_current_term_tl` 4935, 5156,
 5210, 5518, 5621, 5622, 5623, 5706,
 5734, 5747, 5762, 5773, 5789, 5806,
 5818, 5835, 6004, 6426, 6432, 6622
`\stex_current_this:` . 5202, 6075, 6085
`\l_stex_current_this_tl`
 6077, 6081, 6095, 6098
`\l_stex_current_type_tl`
 121, 5078, 5209, 6294,
 6349, 6356, 6359, 6363, 6678, 6727
`\stex_deactivate_macro:Nn`
 132, 348, 348, 2574,
 2981, 2982, 2983, 2984, 2985, 2986,
 2987, 3268, 3337, 3471, 3472, 3473,
 3490, 3511, 3535, 3600, 3621, 3642,
 3736, 3961, 4179, 4519, 6130, 6204,
 6266, 6982, 6988, 6996, 7004, 7025,
 7058, 7076, 7085, 7360, 7433, 7453,
 7458, 7465, 7474, 7572, 7596, 7669,
 8437, 8605, 8606, 8615, 8670, 8697
`\stex_debug:nn` .. 110, 64, 64, 101,
 105, 176, 195, 246, 513, 638, 705,
 712, 859, 929, 932, 967, 971, 982,
 1023, 1031, 1059, 1064, 1084, 1087,
 1120, 1163, 1191, 1202, 1212, 1232,
 1248, 1263, 1518, 1583, 1588, 1628,
 1632, 1648, 1651, 1657, 1659, 1663,
 1666, 1672, 1674, 1677, 1686, 1687,
 1713, 1760, 1762, 1764, 1859, 1865,
 1870, 1877, 2227, 2229, 2245, 2247,
 2251, 2268, 2270, 2277, 2279, 2301,
 2314, 2327, 2329, 2351, 2391, 2469,
 2471, 2480, 2584, 2609, 2617, 2621,
 2656, 2922, 2954, 2958, 2961, 3071,
 3076, 3078, 3086, 3093, 3098, 3102,
 3141, 3160, 3177, 3183, 3190, 3206,
 3217, 3232, 3250, 3258, 3260, 3347,
 3354, 3391, 3412, 3428, 3432, 3442,
 3460, 3462, 3885, 3887, 3889, 3891,
 4008, 4025, 4064, 4075, 4077, 4230,
 4262, 4273, 4323, 4500, 4504, 4580,
 4589, 4594, 4615, 4764, 4883, 4923,
 5142, 5147, 5243, 5248, 5253, 5272,
 5275, 5278, 5298, 5350, 5359, 5383,
 5391, 5507, 5858, 6141, 6213, 6293,
 6833, 6836, 6838, 6842, 7018, 7031,
 7045, 7080, 7254, 7309, 7408, 7605
`\c_stex_debug_clist`
 ... 28, 37, 65, 68, 89, 93, 95, 99, 103
`\c_stex_default_metatheory` 2173, 7565
`\l_stex_default_notation`
 ... 4532, 4537, 4540, 4542, 4543,
 4560, 5113, 5287, 5292, 5313, 5647
`\stex_do_default_notation:`
 4522, 5112, 5285, 5646
`\stex_do_default_notation_op:` ..
 4522, 4523, 4558, 5310
`_stex_do_deprecation:n`
 110, 431, 431, 2307, 3712, 3813
`_stex_do_for_list:`
 ... 2991, 6799, 6814, 7178, 7249, 7304
`_stex_do_id:` 122, 436,
 436, 1351, 6878, 6893, 7179, 7258,
 7318, 7327, 7336, 7416, 8328, 8765
`\stex_do_up_to_module:n`
 113, 2552, 2552, 2555, 2562, 3319, 6281
`\l_stex_docheader_sect`
 111, 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, 7810, 7817
`_stex_eat_exclamation_point:` ...
 3767, 5276, 5287, 5723, 5725
`_stex_end:` 420, 428, 2889, 2897
`_stex_every_file:`
 998, 1001, 1006, 1018, 7675
`\stex_every_module:n`
 112, 2289, 2292,
 2576, 3307, 3491, 3512, 3536, 3601,
 3622, 3643, 3737, 3962, 4180, 4520,
 6131, 6205, 6268, 6278, 7573, 7597
`\g_stex_every_module_tl`
 2290, 2293, 2305
`\l_stex_every_symbol_tl` 5157, 5162,
 5164, 5165, 5171, 5172, 5175, 5213
`\stex_execute_in_module:n`
 112, 2306, 2380, 2560,
 2560, 2564, 2595, 2627, 3276, 3519,
 3946, 6121, 6181, 6193, 6232, 6249

```

\stex_fatal_error:n    132, 80, 80, 1118
\stex_fatal_error:nnn   .....
.... 132, 80, 83, 86, 1129, 2029, 2098
\l_stex_feature_name_str .....
..... 119, 2833, 2837,
2866, 2937, 2942, 2958, 2959, 2961,
2962, 3005, 3350, 3366, 3400, 3404
\stex_file_in_smsmode:nn .....
... 125, 2149, 2168, 2197, 2343, 3246
\stex_file_resolve:Nn   .....
127,
128, 687, 692, 708, 715, 877, 978,
987, 989, 992, 1047, 1075, 1077,
1603, 1621, 1711, 3094, 3188, 3215
\stex_file_set:Nn .. 127, 128, 681,
681, 686, 699, 710, 873, 920, 1015, 1072
\stex_file_split_off_ext:NN .....
.... 127, 790, 790, 895, 2341, 3087
\stex_file_split_off_lang:NN ...
. 127, 790, 797, 892, 2342, 3088, 8301
\stex_file_use:N .....
127,
128, 705, 712, 764, 764, 841, 846,
851, 905, 911, 926, 929, 982, 993,
997, 1052, 1072, 1078, 1083, 1127,
1130, 1163, 1164, 1166, 1200, 1604,
1605, 1617, 1696, 1702, 1707, 1712,
1714, 1862, 1869, 1874, 1886, 1889,
1913, 1918, 1929, 1985, 1995, 2035,
2039, 2049, 2053, 2063, 2068, 2344,
3090, 3093, 3094, 3096, 3117, 3132,
3142, 3161, 3189, 3205, 3216, 3231
\stex_filestack_pop: .....
... 127, 1007, 1007, 1032, 1039, 2194
\stex_filestack_push:n .....
.... 127, 985, 985, 1025, 1027, 2178
\l_stex_fors_seq .....
2841,
2844, 6800, 6803, 6817, 6834, 6835,
6849, 6850, 6908, 6935, 7182, 7183,
7233, 7234, 7253, 7308, 7389, 7390
\stex_get_env:Nn .....
.... 132, 50, 51, 59, 87, 300, 604,
610, 974, 976, 1043, 1045, 1050, 3653
\stex_get_in_morphism:n .....
121, 2843, 2994, 2994, 3340, 3384, 3566
\_stex_get_mathstructure:n .....
..... 2802, 6158, 6163
\stex_get_mathstructure:n .....
..... 6157, 6157, 6175, 6224, 6763
\l_stex_get_structure_module_str ...
.. 2803, 6159, 6164, 6168, 6182, 6233
\_stex_get_symbol:n . 3980, 3986, 6165
\stex_get_symbol:n .....
.... 114, 115, 121, 1824, 3979,
3979, 4144, 4419, 4924, 5231, 5917,
5925, 5936, 6802, 6960, 7011, 7064
\l_stex_get_symbol_args_tl .....
.... 114, 116,
224, 3043, 3373, 3756, 3875, 3877,
3878, 4014, 4055, 4087, 4098, 4100,
4113, 4528, 4793, 4806, 4888, 4988,
5001, 5236, 5401, 6146, 6826, 7650
\l_stex_get_symbol_arity_int ...
.... 114, 116, 224,
3042, 3372, 3755, 3797, 3821, 3827,
3831, 3835, 3839, 3843, 3847, 3851,
3855, 3859, 3862, 3863, 3864, 3865,
3876, 3934, 4006, 4013, 4054, 4086,
4190, 4225, 4257, 4264, 4284, 4334,
4451, 4763, 4792, 4805, 4846, 4887,
4987, 5000, 5039, 5235, 6825, 7649
\l_stex_get_symbol_def_tl .....
.... 114, 2862, 3044,
3348, 4015, 4056, 4088, 4889, 5237
\l_stex_get_symbol_invoke_cs ...
.... 114, 3047, 3377,
4018, 4059, 4091, 4892, 5240, 6167
\l_stex_get_symbol_macro_str ...
.... 121, 3041, 3371
\l_stex_get_symbol_mod_str . 114,
1826, 2845, 2855, 3039, 3355, 3359,
3369, 3394, 3403, 3938, 3944, 3987,
4011, 4052, 4084, 4150, 4154, 4169,
4332, 4421, 4425, 4429, 4433, 4440,
4444, 4501, 4508, 4885, 5233, 5997,
6804, 6963, 7013, 7066, 7659, 7665
\l_stex_get_symbol_name_str .....
114, 1826, 2845, 2856, 2995, 2999,
3003, 3040, 3347, 3349, 3355, 3359,
3369, 3391, 3394, 3403, 3404, 3939,
3944, 3981, 3988, 4012, 4053, 4085,
4150, 4154, 4168, 4169, 4332, 4421,
4425, 4429, 4433, 4440, 4444, 4458,
4463, 4466, 4502, 4508, 4886, 4912,
4914, 4920, 4922, 5234, 5927, 5938,
5960, 5971, 5973, 5984, 5986, 5997,
5998, 5999, 6000, 6166, 6804, 6963,
6964, 6965, 6966, 6993, 7001, 7009,
7013, 7054, 7062, 7066, 7660, 7665
\stex_get_symbol_or_var:n .....
.... 115, 4882, 4919
\l_stex_get_symbol_return_tl ...
.... 114, 3046, 3376, 4017,
4058, 4090, 4765, 4891, 4954, 5239
\l_stex_get_symbol_type_tl .....
114, 3045, 3375, 4016, 4057, 4089,
4890, 5238, 6168, 6176, 6226, 6765
\stex_get_var:n .....
115, 4457,
4882, 4911, 5958, 5969, 5982, 7052
\c_stex_home_file .....
128, 1041

```

```

\stex_if_check_terms: .....
..... 114, 3649, 3660, 3664
\stex_if_check_terms:TF 114, 3648,
3669, 4146, 4460, 4767, 4963, 4966
\stex_if_check_terms_p: ... 114, 3648
\stex_if_do_html: .....
281
\stex_if_do_html:TF .....
130,
278, 290, 1254, 1281, 1290, 1534,
2520, 2540, 2778, 2794, 2824, 2872,
3057, 3281, 3322, 3358, 3392, 3746,
3930, 3942, 4148, 4507, 4761, 4769,
4815, 4962, 6187, 6243, 6771, 6829,
7053, 7250, 7268, 7278, 7295, 7305,
7339, 7346, 7348, 7386, 7607, 7632,
7664, 8307, 8338, 8396, 8405, 8415,
8421, 8453, 8462, 8472, 8478, 8502,
8511, 8521, 8527, 8551, 8560, 8570,
8576, 8602, 8611, 8657, 8664, 8681
\stex_if_do_html_p: .....
130
\stex_if_file_absolute:N 127, 747, 757
\stex_if_file_absolute:NTF .....
..... 127, 745, 991, 1076
\stex_if_file_absolute_p:N . 127, 745
\stex_if_file_starts_with:NN 127, 767
\stex_if_file_starts_with:NNTF ..
..... 127, 767, 1199
\stex_if_html_backend:TF .....
..... 130, 295, 320,
327, 336, 619, 1254, 1298, 1360,
1379, 1426, 1936, 1963, 3648, 5185,
5219, 5731, 5787, 5816, 5994, 6020,
6872, 6882, 6958, 7710, 7750, 7875,
7955, 8003, 8011, 8023, 8033, 8143
\stex_if_html_backend_p: ... 130, 295
\stex_if_in_module: .....
2544
\stex_if_in_module:TF .....
..... 112, 2296, 2544, 2560
\stex_if_in_module_p: ... 112, 2544
\stex_if_module_exists:n .....
2548
\stex_if_module_exists:nTF .....
112, 2313, 2326, 2548, 3145, 3148,
3164, 3167, 3252, 6177, 6227, 6766
\stex_if_module_exists_p:n 112, 2548
\stex_if_smsmode: .....
2134
\stex_if_smsmode:TF .....
..... 125, 437, 1246, 1520, 2132,
2199, 2332, 2531, 2539, 3061, 3204,
3230, 3285, 3326, 3478, 3485, 3498,
3506, 3522, 3530, 3592, 3612, 3633,
3723, 3951, 4157, 4510, 6865, 6881,
6894, 6907, 6934, 7037, 8321, 8337
\stex_if_smsmode_p: ..... 125, 2132
\stex_ignore_spaces_and_pars: ...
. 132, 357, 357, 360, 2570, 2571, 8348
\l_stex_import_archive_str . 120,
2472, 2477, 2809, 3054, 3079, 3083,
3103, 3104, 3105, 3273, 3300, 3316
\stex_import_module_uri:nn .....
..... 120, 2470, 2806,
3052, 3070, 3070, 3271, 3297, 3314
\l_stex_import_name_str .....
120, 2474, 2479, 2811, 3056, 3064,
3073, 3275, 3289, 3302, 3318, 3330
\l_stex_import_ns_str .....
2480,
2483, 2813, 3059, 3063, 3146, 3149,
3152, 3165, 3168, 3171, 3179, 3277,
3280, 3283, 3288, 3320, 3324, 3329
\l_stex_import_path_str .....
..... 120, 2473, 2478,
2810, 3055, 3074, 3085, 3089, 3093,
3094, 3095, 3098, 3274, 3301, 3317
\stex_import_require_module:nnn .
..... 120, 2476, 2808,
3053, 3108, 3108, 3121, 3272, 3315
\stex_import_require_module_-
safe:nnn .....
3123, 3299
\l_stex_import_uri_str .....
..... 120, 3084, 3105, 3111,
3116, 3126, 3131, 3139, 3141, 3145,
3146, 3152, 3158, 3160, 3164, 3165,
3171, 3190, 3197, 3217, 3252, 3253
\stex_in_archive:nn 124, 1092, 1092,
1884, 1935, 1960, 2018, 2071, 2109
\stex_in_invisible_html_bool ...
..... 3787, 4836, 5729, 5730, 5760
\l_stex_in_meta_bool 2375, 2376, 2381
\l_stex_inpararray_bool .....
5860
\stex_input_with_hooks:n .....
... 1020, 1886, 1889, 1910, 1927, 1929
\_stex_invoke_notation:w .....
..... 5148, 5265, 5266, 5271, 5308
\stex_invoke_outer_field: .....
6151
\stex_invoke_sequence: .....
..... 4992, 5005, 5056, 5056, 5597
\stex_invoke_sequence_in: .....
122
\stex_invoke_sequence_range: ... 122
\stex_invoke_structure: .....
..... 6119, 6167, 6287
\stex_invoke_symbol .....
..... 121
\stex_invoke_symbol: .....
..... 115, 121, 3760,
4797, 4810, 5242, 5242, 6827, 7654
\l_stex_invoke_symbol:nnnnnnnnN ...
..... 113, 121,
2613, 3995, 4009, 4010, 5183, 5183,
5196, 5232, 6693, 6707, 6715, 6720
\l_stex_invoke_symbol:nnnnnnnnN\l_-
stex_current_symbol_str .....
121

```

```

\stex_invoke_text_symbol: 3926, 3974
\stex_invoke_variable:nnnnn . 5217
\stex_invoke_variable:nnnnnnN ...
..... 121, 4803, 4998, 5217, 5595
\stex_is_sequentialized:n ...
..... 5589, 5616, 5691, 5701
\stex_iterate_break: 113, 2669, 2672
\stex_iterate_break:n . 113, 2669,
2675, 2752, 3433, 3443, 3458, 4083
\stex_iterate_morphisms:nn ...
..... 2735, 2735, 3411, 3427
\stex_iterate_notations:nn ...
..... 116, 2710, 2710, 2931, 6727
\stex_iterate_symbols:n ...
..... 113, 2668, 2668, 4076
\stex_iterate_symbols:nn ...
..... 113, 2685, 2685, 2894, 3461, 6135, 6678
\l_stex_key_archive_str ...
..... 122, 385, 388, 1567, 1585,
1599, 1608, 1609, 1686, 1693, 1707
\l_stex_key_argnames_clist ... 114
\l_stex_key_args_str ...
..... 114, 3681, 3686, 3695, 3729,
3773, 3823, 3828, 3832, 3836, 3840,
3844, 3848, 3852, 3856, 3860, 3879,
4485, 4818, 4949, 4950, 5015, 6784
\l_stex_key_argtypes_clist ...
..... 3699, 3704, 3803, 3805,
3892, 4851, 4854, 5044, 5047, 6788
\l_stex_key_assoc_str 3684, 3689,
3777, 3778, 4822, 4823, 5019, 5020
\l_stex_key_continues_tl . 7091, 7100
\l_stex_key_def_tl 115, 3697, 3707,
3728, 3757, 3791, 3792, 3888, 3899,
3903, 3923, 4484, 4794, 4807, 4840,
4841, 4989, 5002, 5033, 5034, 6786
\l_stex_key_deprecate_str ...
..... 411, 413, 432, 433
\l_stex_key_due_tl ...
..... 8740, 8744, 8779, 8780
\l_stex_key_feedback_tl ...
..... 8624, 8629, 8651, 8652
\l_stex_key_file_str ...
..... 122, 386, 389, 1568,
1584, 1587, 1604, 1619, 1647, 1662,
1673, 1686, 1698, 1702, 1708, 1780
\l_stex_key_for_clist 2842, 6783,
6791, 6801, 7088, 7097, 7369, 7374
\l_stex_key_for_str ...
..... 6871
\l_stex_key_from_tl ... 7089, 7098
\l_stex_key_Ftext_tl 8627, 8635, 8649
\l_stex_key_functions_tl . 7093, 7101
\l_stex_key_given_tl ...
..... 8739, 8743, 8776, 8777
\l_stex_key_hide_bool 7095, 7104, 7237
\l_stex_key_id_str ...
..... 122, 393, 395, 438,
439, 6852, 6853, 6920, 6922, 7315,
7324, 7413, 8389, 8390, 8392, 8398,
8446, 8447, 8449, 8455, 8495, 8496,
8498, 8504, 8544, 8545, 8547, 8553
\l_stex_key_intent_args_clist ...
..... 4127, 4133, 4299, 4308, 4310
\l_stex_key_intent_str ...
..... 3936, 4126, 4132, 4222, 4302
\l_stex_key_macroname_str ...
.. 6782, 6792, 6809, 6811, 6820, 6824
\l_stex_key_method_tl ...
.. 7094, 7103, 7224, 7225, 7371, 7375
\l_stex_key_mhrepos_str ...
.. 8270, 8275, 8371, 8373, 8381, 8792
\l_stex_key_min_tl ...
..... 8268, 8273, 8327, 8345, 8365
\l_stex_key_name_str ...
..... 114, 115, 3694, 3701, 3725, 3726,
3740, 3741, 3754, 3763, 3772, 3813,
3868, 3897, 3901, 3910, 3911, 3913,
3921, 3929, 3939, 3947, 3953, 3954,
4458, 4481, 4482, 4500, 4502, 4753,
4754, 4764, 4771, 4774, 4789, 4791,
4804, 4817, 4866, 4869, 4870, 4874,
4876, 4877, 4946, 4947, 4969, 4984,
4986, 4999, 5014, 6781, 6790, 6810,
6811, 6815, 6818, 6824, 6832, 6841,
6870, 7370, 7377, 7393, 7394, 7407,
7408, 7409, 7638, 7639, 7648, 7660,
8269, 8274, 8300, 8302, 8305, 8323
\l_stex_key_number_tl ...
..... 8738, 8742, 8758, 8759
\l_stex_key_op_tl 3935, 4125, 4131,
4183, 4184, 4185, 4192, 4193, 4200,
4205, 4336, 4370, 4374, 4432, 4437,
4453, 4873, 4875, 4878, 4956, 4958
\l_stex_key_post_tl .. 5906, 5910,
5927, 5938, 5971, 5984, 6993, 7001
\l_stex_key_pre_tl .. 5905, 5909,
5927, 5938, 5971, 5984, 6993, 7001
\l_stex_key_prec_str .. 116, 3937,
4124, 4130, 4234, 4237, 4240, 4246,
4249, 4252, 4255, 4261, 4273, 4274
\l_stex_key_proofend_tl ...
..... 7090, 7099, 7201, 7202
\l_stex_key_proot_tl ... 5907
\l_stex_key_pts_tl ...
..... 8267, 8272, 8326, 8342, 8365
\l_stex_key_reorder_str 3683, 3687,
3780, 3781, 4828, 4829, 5025, 5026

```

```

\l_stex_key_return_tl . . . . .
    ... 3698, 3703, 3759, 3794, 3797,
        3890, 4765, 4796, 4809, 4843, 4846,
        4954, 4991, 5004, 5036, 5039, 6787
\l_stex_key_role_str . . . . .
    ... 3682, 3690, 3783, 3784,
        3915, 4825, 4826, 5022, 5023, 6821
\l_stex_key_root_tl . . . . . 5911
\l_stex_key_short_tl . . . . .
    ... 1313, 1316, 1343, 1345
\l_stex_key_sig_str . . . . .
    ... 112, 2302, 2344, 2492, 2507, 2524, 8311
\l_stex_key_style_clist . . . . .
    ... 123, 405, 407,
        457, 461, 508, 512, 6855, 6856, 6949
\l_stex_key_T_bool . . . . .
    ... 8625, 8631, 8633, 8646, 8666
\l_stex_key_term_tl . . . . .
    ... 7092, 7102, 7221, 7222, 7372, 7376
\l_stex_key_title_str . . . . . 6793
\l_stex_key_title_tl . . . . . 399,
    401, 1569, 1747, 1748, 2514, 6869,
    6874, 8317, 8318, 8324, 8388, 8430,
    8431, 8445, 8487, 8488, 8494, 8536,
    8537, 8543, 8585, 8586, 8771, 8772
\l_stex_key_Ttext_tl 8626, 8634, 8647
\l_stex_key_type_tl . . . . . 115,
    3696, 3706, 3727, 3758, 3788, 3789,
    3886, 3898, 3902, 4483, 4795, 4808,
    4837, 4838, 5030, 5031, 6785, 6794
\l_stex_key_variant_str . . . . .
    ... 116, 4123, 4129, 4136,
        4138, 4167, 4220, 4333, 4342, 4374,
        4468, 4487, 4776, 4867, 4875, 4971
\stex_keys_define:nnnn . . . . .
    ... 122, 365, 365, 384, 392, 398,
        404, 410, 416, 1342, 1553, 1559,
        2491, 3680, 3693, 3896, 4122, 4478,
        4744, 5904, 6094, 6780, 7087, 7368,
        7902, 8266, 8370, 8623, 8737, 8836
\stex_keys_set:nn . . . . . 122, 123,
    380, 380, 1349, 1571, 1772, 1777,
    2513, 3715, 3907, 3908, 4143, 4456,
    4497, 4751, 4944, 5916, 5924, 5935,
    5957, 5968, 5981, 6109, 6862, 6891,
    6985, 6991, 6999, 7177, 7248, 7303,
    7406, 7602, 7636, 7982, 8293, 8297,
    8380, 8644, 8752, 8756, 8791, 8849
\stex_kpsewhich:Nn 132, 42, 42, 53, 60
\c_stex_language_abbrevs_prop . .
    ... 129, 109
\stex_language_from_file: . . . .
    ... 130, 157, 157, 1003
\c_stex_languages_clist . 29, 180, 183
\c_stex_languages_prop . . . . .
    ... 129, 109, 137, 138, 172, 188, 804
\g_stex_last_feature_str . . . . .
    ... 118, 2792, 6135
\stex_macro_body:N . 131, 543, 545, 563
\stex_macro_definition:N 131, 558, 558
\l_stex_macroname_str . . . . .
    ... 115, 217, 2782,
        2783, 3713, 3717, 3719, 3753, 3774,
        3775, 3909, 3920, 4498, 4752, 4790,
        4802, 4819, 4820, 4945, 4985, 4997,
        5016, 5017, 6120, 6820, 7637, 7647
\_stex_main_archive: . . . . . 1198, 1237
\c_stex_main_archive_prop . 124, 1198
\c_stex_main_file . . . . .
    ... 127, 128, 973, 1012, 1078, 7672
\stex_map_args:N . . . . .
    ... 3798, 4097, 4097, 4199, 4295,
        4366, 4530, 4847, 5040, 5403, 6149
\stex_map_notation_args:N . . . .
    ... 4097, 4110, 4399
\stex_map_uri:Nnnnn 128, 810, 817, 2359
\c_stex_mathhub_file 128, 917, 1041,
    1117, 1127, 1130, 1140, 1199, 1617,
    1696, 1707, 1862, 1874, 1886, 1889,
    1913, 1918, 1929, 1981, 2035, 2039,
    2049, 2053, 2063, 2068, 3117, 3132
\c_stex_mathhub_main_manifest_-
    prop . . . . . 1205, 1206
\stex_mathml_arg:nn . . . . . 131
\stex_mathml_intent:nn . . . . . 131
\_stex_maybe_brackets:nn . . . .
    ... 4583, 4600, 5064, 5300, 5311, 5853
\stex_metagroup_do_in:nn . . . .
    ... 113, 133, 231, 236,
        243, 2553, 3365, 3399, 7644, 7645, 7656
\stex_metagroup_new:n . . . .
    ... 133, 225, 226, 230, 2304, 2353, 2837
\l_stex_metatheory_uri . . . .
    ... 2173, 2379, 2382, 2466,
        2482, 2496, 2498, 2525, 2526, 7563,
        7564, 7565, 7612, 7613, 8312, 8313
\c_stex_module_ . . . . . 113, 115
\stex_module_add_code:n . . . .
    ... 113, 2556, 2556, 2559, 2561, 2571, 7620
\stex_module_add_morphism:nnn .. 112
\stex_module_add_morphism:nnnn .. .
    ... 120, 2577, 2577,
        2582, 2941, 3279, 6191, 6247, 7623
\stex_module_add_notation:nnnnn .
    ... 115, 2620,
        2620, 2631, 2933, 2936, 4331, 4450
\stex_module_add_symbol:nnnnnnN .
    ... 112, 2583

```

```

\stex_module_add_symbol:nnnnnnN
    .... 113, 2583, 2955, 2959, 2962,
    3752, 3919, 6115, 6147, 6823, 7646
\stex_module_setup:n .. 112, 2295,
    2295, 2518, 2788, 7483, 7603, 8305
\_stex_module_setup_top_nosig:n ..
    ..... 2303, 2312, 2372, 7599
\l_stex_morphism_morphisms_seq ..
    ..... 119, 2870, 2880, 2975
\l_stex_morphism_renames_prop ...
    ..... 119, 2869, 2879,
    2926, 2932, 2945, 2953, 3000, 3402
\l_stex_morphism_symbols_prop ...
    ..... 119, 2858,
    2868, 2878, 2896, 2900, 2909, 2923,
    2928, 2996, 3014, 3019, 3368, 3464
\stex_new_statement:nn .... 6799
\stex_new_statement:nmm .....
    ..... 6860, 6926, 6932, 6947, 6948
\stex_new_stylable_cmd:nnnn .....
    ..... 123, 443, 443,
    3051, 3264, 3313, 3339, 3383, 3409,
    3587, 3607, 3628, 3714, 3906, 4142,
    4455, 4496, 4750, 4943, 7175, 7569
\stex_new_stylable_env:nnnnnnn .. .
    ..... 123, 443,
    478, 2512, 3474, 3496, 3516, 6086,
    6172, 6221, 6861, 7272, 7290, 7302,
    8290, 8404, 8461, 8510, 8559, 8749
\_stex_next_symbol:n .....
    .. 5160, 5161, 5179, 6498, 6620, 6980
\l_stex_notation_ .. .....
    ..... 116
\l_stex_notation_add: .....
    .. 3941, 4147, 4319, 4330, 4506, 7663
\l_stex_notation_args_tl .. 4111,
    4189, 4223, 4294, 4300, 4306, 4504
\l_stex_notation_check: 4146, 4319,
    4319, 4460, 4505, 4767, 4966, 7662
\l_stex_notation_do_html:n .. 3944,
    4150, 4319, 4339, 4508, 4771, 7665
\l_stex_notation_downprec .. 5519,
    5847, 5848, 5858, 5859, 5879, 5881
\l_stex_notation_macrocode_cs ...
    ..... 115–117, 4173, 4218,
    4230, 4324, 4335, 4364, 4424, 4452,
    4471, 4490, 4779, 4868, 4871, 4974
\l_stex_notation_make_args: 4174,
    4319, 4325, 4398, 4472, 4491, 4780
\l_stex_notation_parse:n .....
    ..... 3940, 4145, 4182, 4182,
    4459, 4503, 4766, 4957, 4960, 7661
\l_stex_notation_parse_and_then:nw
    ..... 116, 118
\_stex_notation_set_default:n ...
    ..... 4153, 4418, 4439, 4449, 4463
\l_stex_notation_top:nw .....
    ..... 118
\l_stex_persist:n .....
    ..... 112, 131, 616, 620, 621, 623, 626,
    629, 630, 636, 637, 1192, 1217, 2401
\c_stex_persist_mode_bool .....
    ..... 131, 31, 607, 670, 1236
\l_stex_persist_read_now: .....
    ..... 669, 1216, 7674
\c_stex_persist_write_mode_bool .
    ..... 131, 32, 613, 618, 671, 677, 1215, 2390
\l_stex_pseudogroup:nn .....
    ..... 132,
    133, 205, 205, 286, 1096, 2658, 5392
\l_stex_pseudogroup_restore:N .....
    ..... 132, 133, 205, 208, 1109, 2662
\l_stex_pseudogroup_with:nn .....
    ..... 132, 215, 215, 818,
    940, 2381, 2669, 2687, 2712, 4009,
    4679, 4688, 4713, 5879, 5893, 6015
\c_stex_pwd_file .....
    ..... 128, 973, 993, 1199, 1200, 1714, 1869
\l_stex_reactivate_macro:N 132, 348,
    354, 2576, 2988, 2989, 2990, 3307,
    3309, 3492, 3513, 3537, 3602, 3623,
    3644, 3737, 3962, 4180, 4520, 6131,
    6206, 6269, 6912, 6913, 6914, 6915,
    6916, 6930, 6939, 6940, 6941, 6942,
    6943, 6944, 6945, 7259, 7260, 7261,
    7262, 7263, 7264, 7265, 7266, 7573,
    7597, 7625, 8331, 8332, 8333, 8607
\l_stex_ref_new_doc_target:n .....
    ..... 111, 122, 439, 1502, 1515, 1539
\l_stex_ref_new_id:n .. 1505, 1516, 6921
\l_stex_ref_new_sym_target:n .....
    ..... 111, 1785, 1809, 1818, 6909, 6936, 6970
\l_stex_ref_new_sym_target:nn 111, 1816
\l_stex_ref_new_symbol:n .....
    ..... 111, 1785, 2426, 3762, 3928
\l_stex_ref_url_str .. 1512, 1533, 1535
\l_stex_renamedecl_do:nn .....
    ..... 3385, 3390, 3584
\l_stex_require_archive:n .....
    ..... 120, 124, 1086, 1115, 1115,
    1125, 1608, 1873, 3104, 3114, 3129
\l_stex_resolve_path_pair:Nnn .....
    ..... 124, 1858, 1858, 1880
\l_stex_return_args:nn .....
    ..... 3766, 3798, 4847, 5040
\l_stex_return_notation_tl .....
    ..... 5200, 5377, 5378, 6336,
    6344, 6506, 6507, 6587, 6613, 6614
\l_stex_set_current_archive:n .....
    ..... 124, 1085, 1085, 1105, 1211, 3248

```

```

\stex_set_current_namespace: . . .
    ..... 129, 969, 969, 1004
\stex_set_document_uri: . . .
    ..... 129, 965, 965, 1002
\stex_set_language:n . . .
    ..... 130, 135, 135, 156, 173, 2505
\stex_set_notation_macro:nnnn .. .
    115, 116, 2620, 2637, 2641, 2641, 2653
\stex_sms_allow:N . . .
    125, 2116, 2116, 2127, 2128, 2129, 2130, 2131
\stex_sms_allow_env:n . . .
    126, 2116, 2124, 2543, 3494, 3515, 3539,
    6129, 6203, 6265, 6886, 7634, 8353
\stex_sms_allow_escape:N . . .
    126, 2116, 2120, 2490, 2575, 3306,
    3344, 3388, 3604, 3625, 3646, 3738,
    3963, 4181, 4521, 6902, 7026, 7670
\stex_sms_allow_import:Nn . . .
    ..... 126, 2137, 2139, 3308
\stex_sms_allow_import_env:nn . . .
    ..... 126, 2137, 2143
\g_stex_sms_import_code . . .
    ..... 126, 2148, 2171, 2189, 3298
\stex_smsmode_do: . . .
    126, 2162, 2164, 2198, 2203, 2488, 2536,
    2572, 3266, 3335, 3342, 3386, 3483,
    3503, 3527, 3598, 3619, 3640, 3734,
    3959, 4163, 4516, 6088, 6197, 6254,
    6879, 6900, 7023, 7626, 7667, 8334
\stex_split_slash: . . .
    ..... 5947, 5951, 6000, 6966
\stex_sref_do_aux:n . . .
    ..... 1791, 1798, 1802, 1805, 7479
\stex_str_if_ends_with:nn .. 126, 533
\stex_str_if_ends_with:nnTF . .
    ..... 126, 533, 986, 3418, 3441, 4066
\stex_str_if_ends_with_p:nn . .
    ..... 126, 533, 4041, 4081
\stex_str_if_starts_with:nn 126, 538
\stex_str_if_starts_with:nnTF . .
    ..... 126, 419, 538, 2895, 6314, 6318, 7029
\stex_str_if_starts_with_p:nn . .
    ..... 126, 538
\stex_structural_feature_-
    module:nn ... 118, 2777, 2777, 6125
\stex_structural_feature_module_-
    end: 118, 2777, 2791, 6090, 6199, 6261
\stex_structural_feature_-
    morphism:nnn ..... 2798
\stex_structural_feature_-
    morphism:nnnn ..... 120
\stex_structural_feature_-
    morphism:nnnnn .... 119, 2799,
    3476, 3497, 3518, 3588, 3608, 3629
\stex_structural_feature_-
    morphism_check_total: . . .
    ..... 2908, 3505, 3529, 3617, 3638
\stex_structural_feature_-
    morphism_end: . 119, 2798, 2865,
    3488, 3509, 3533, 3597, 3618, 3639
\stex_style_apply: . . .
    ..... 123, 124, 443, 448, 483,
    488, 2534, 2539, 3066, 3291, 3332,
    3481, 3486, 3501, 3507, 3525, 3531,
    3595, 3615, 3636, 3731, 3956, 4160,
    4475, 4513, 4783, 4978, 6876, 6882,
    7187, 7257, 7277, 7294, 7314, 7332,
    7335, 7344, 8325, 8337, 8401, 8416,
    8420, 8458, 8473, 8477, 8507, 8522,
    8526, 8556, 8571, 8575, 8764, 8767
\stex_suppress_html:n . . .
    ..... 130, 285, 285, 2149, 4213
\_stex_symdecl_check_terms: . . .
    ... 115, 3744, 3883, 3883, 3918, 4758
\stex_symdecl_do: . . .
    ..... 114, 115, 3743, 3812,
    3812, 3917, 4757, 4952, 6822, 7641
\stex_symdecl_html: . . .
    ..... 3747, 3770, 3931, 6829
\stex_symdecl_top:n . . .
    ..... 115, 3721, 3739, 3739, 4499
\stex_symdef_styledefs: . 4480, 4512
\stex_term_arg:nnn . . .
    ..... 5510, 5514, 5521, 5531
\stex_term_arg:nnnn . . .
    ..... 4414, 4644, 4652,
    5514, 5515, 5546, 5580, 5617, 5702
\stex_term_arg_aB:nnnn . . .
    ..... 4657, 4665, 4672, 5536, 5544
\stex_term_do_aB_clist: .. 4679,
    4680, 4689, 4693, 4714, 4719, 5541,
    5553, 5582, 5632, 5661, 5709, 5712
\stex_term_oma:nnn . . .
    ..... 4596, 5402, 5787, 5788, 5814
\stex_term_oma_or_omb:nnn . . .
    ..... 4596, 4601, 4649, 4662
\stex_term_omb:nnn . . .
    ..... 4649, 4662, 5430, 5431, 5816, 5817, 5843
\stex_term_oms:nnn . . .
    ..... 4374, 5187, 5189, 5723,
    5733, 5782, 5786, 5918, 5928, 5939
\stex_term_oms_or_omv:nnn . . .
    ..... 3976, 4584, 5065, 5187,
    5189, 5221, 5223, 5301, 5312, 5385,
    5723, 5786, 6353, 6391, 6429, 6485
\stex_term_omv:nnn . . .
    ..... 4936, 5221, 5223,
    5723, 5746, 5783, 5962, 5975, 5988

```

```

\stex_undefined: ..... 40
\stex_uri_add_module:Nn .. .
..... 129, 938, 938, 963, 1517, 7563
\stex_uri_from_current_file:Nn ..
..... 129, 928, 928, 966
\stex_uri_from_current_file_-_
nolang:Nn .. .
129, 928, 934, 970
\stex_uri_from_repo_file .. .
129
\stex_uri_from_repo_file:NNNn ..
..... 128, 129, 891, 894, 930, 1622
\stex_uri_from_repo_file_-_
nolang:NNNn .. .
129, 891, 891, 935
\stex_uri_resolve:Nn .. .
128,
876, 876, 879, 2482, 2498, 2502, 7482
\stex_uri_set:Nn 128, 826, 872, 875, 926
\stex_uri_use:N 128, 859, 880, 884,
932, 967, 971, 1512, 1518, 1519,
1523, 1589, 1591, 1624, 1631, 1648,
1649, 2382, 2526, 3143, 3148, 3149,
3162, 3167, 3168, 6853, 7613, 8313
\stex_vardecl_notation_macro: ..
... 115, 4461, 4768, 4864, 4864, 4967
\stex_variable:nnnnnnnN . 4743, 4884
\l_stex_variables_prop .. .
... 115, 4741, 4789, 4896, 4984
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:nn .. .
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 .. .
66, 69, 74
\l_stex_debug_cl .. .
89, 91, 94
\stex_debug_env_str .. .
87, 88, 91
\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 .. .
..... 1460, 1463, 1481
\stex_doc_orig_frontmatter .. .
..... 1450, 1453, 1470
\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:nnnn .. .
5551, 5558
\stex_expr_aB_simple_arg:nnnn .. .
..... 5569, 5578
\stex_expr_add_prop_arg:nnw .. .
..... 5398, 5422, 5425
\stex_expr_arg:n .. .
5399, 5435
\l_stex_expr_arg_counter_int .. .
..... 5389, 5396, 5411, 5447, 5448
\stex_expr_arg_do:nn .. .
..... 5449, 5455, 5471, 5506, 5513
\stex_expr_arg_inner:nn .. .
..... 5438, 5441, 5445, 5451
\stex_expr_assoc_make_seq:nnn .. .
..... 5613, 5642, 5721
\stex_expr_assoc_seq:nnnnnn .. .
..... 5562, 5610
\stex_expr_check:n .. .
5479
\stex_expr_check:nTF .. .
5448, 5454
\stex_expr_check_b:nn .. .
5403, 5428
\l_stex_expr_count_int .. .
... 5543, 5549, 5559, 5580, 5617, 5702
\l_stex_expr_cs .. .
5644, 5647, 5667
\l_stex_expr_customs_prop .. .
..... 5394, 5406, 5407,
5408, 5423, 5459, 5465, 5480, 5483
\l_stex_expr_customs_seq .. .
... 5395, 5412, 5413, 5414, 5424, 5481
\stex_expr_do_aB_clist: .. .
..... 5536, 5541, 5582, 5661
\stex_expr_do_ab_next:nnn .. .
..... 5402, 5404, 5430, 5431
\stex_expr_do_headerterm:nn .. .
..... 5711, 5743, 5756, 5759, 5784
\stex_expr_do_ref:nNn .. .
5918,
5926, 5937, 5959, 5970, 5983, 5993
\stex_expr_do_seqmap:nnnnnn .. .
..... 5567, 5674
\stex_expr_end: .. .
5564, 5574

```

```

\__stex_expr_gobble:nnnnnnnn . .
    ..... 5564, 5574
\l__stex_expr_iarg_tl 5654, 5656, 5667
\__stex_expr_invoke_custom:n . .
    ..... 5249, 5263, 5390
\__stex_expr_invoke_math: 5244, 5252
\__stex_expr_invoke_op_custom:n . .
    ..... 5249, 5256, 5382
\__stex_expr_invoke_op_notation:w . .
    ..... 5258, 5259, 5297
\__stex_expr_invoke_return: . .
    ..... 5338, 5341
\__stex_expr_invoke_return_-
    maybe:n ..... 5280, 5290, 5319
\__stex_expr_invoke_return_next: . .
    ..... 5326, 5335
\__stex_expr_invoke_text: 5244, 5247
\__stex_expr_is_seqmap:n ..... 5603
\__stex_expr_is_seqmap:nTF . .
    ..... 5566
\__stex_expr_is_varseq:n ..... 5592
\__stex_expr_is_varseq:nTF 5560, 5679
\l__stex_expr_left_bracket_str . .
    ..... 5849, 5875, 5883, 5893, 5894
\l__stex_expr_old_seq . .
    ..... 5686, 5689, 5693, 5698
\l__stex_expr_reset_tl . .
    ..... 5160, 5163, 5167, 5168, 5174
\__stex_expr_ret_cs . .
    ..... 5323, 5328, 5355, 5360, 5365
\__stex_expr_return_arg:n 5325, 5331
\l__stex_expr_return_args_tl . .
    .. 5320, 5332, 5337, 5346, 5362, 5367
\__stex_expr_return_notation:n . .
    ..... 5343, 5376
\l__stex_expr_return_this_tl . .
    ..... 5321, 5337,
        5342, 5346, 5350, 5351, 5361, 5369
\l__stex_expr_right_bracket_str . .
    ..... 5850, 5876, 5888, 5893, 5895
\__stex_expr_setup:nnnnnn . .
    ..... 5186, 5198, 5220
\__stex_expr_varseq_in_map:nnnnnnnn . .
    ..... 5681, 5720
\__stex_features_add_definiens:nn . .
    ..... 2849, 2992
\__stex_features_break: .. 2850, 2854
\__stex_features_check_break:nnnnnnnn . .
    ..... 3013, 3018, 3027, 3030, 3037
\__stex_features_clean:nnw 2889, 2897
\__stex_features_do_decls: 2881, 2893
\__stex_features_do_elaboration: . .
    ..... 2871, 2921
\__stex_features_do_for_list: . .
    ..... 2840, 2991
\__stex_features_do_morph:nnnn . .
    ..... 2969, 2973
\__stex_features_do_morphisms:n . .
    ..... 2882, 2967, 2977
\__stex_features_elab_check: . .
    ..... 2929, 2952
\l__stex_features_feature_str . .
    ..... 2832, 2944
\__stex_features_get_check:nnnn . .
    ..... 2997, 3025
\l__stex_features_implicit_bool . .
    ..... 2798, 2817, 2820, 2957
\__stex_features_reactivate: . .
    ..... 2835, 2980
\__stex_features_rename:nn 2945, 2948
\__stex_features_rename_all: . 2885
\__stex_features_renamed_-
    check:nnnn ..... 3001, 3011
\__stex_features_set_definiens_-
    macros: ..... 2850, 2854
\__stex_features_set_definiens_-
    macros_i:nnnnnn ..... 2857, 2861
\__stex_features_setup: .. 2834, 2877
\__stex_features_split_qm:w . .
    ..... 2920, 2938
\l__stex_features_tmp . .
    ..... 2932, 2934, 2953, 2954, 2955
\__stex_features_total_check: . .
    ..... 2910, 2914
\__stex_groups_do: . .
    ..... 256, 263, 272
\__stex_groups_do_in:nn 238, 245, 266
\__stex_groups_exists:n . .
    ..... 231
\__stex_groups_exists:nTF . .
    ..... 237
\l__stex_groups_ids_seq 225, 228, 264
\__stex_groups_tmp . .
    ..... 247, 254, 260
\l__stex_importmodule_archive_-
    str 3110, 3115, 3125, 3130, 3247, 3248
\__stex_importmodule_check_-
    file:nn ..... 3191,
        3192, 3193, 3194, 3195, 3196, 3218,
        3219, 3220, 3221, 3222, 3223, 3257
\__stex_importmodule_get_from_-
    file:nnm ..... 3151, 3187
\__stex_importmodule_get_from_-
    file_safe:nnn ..... 3170, 3214
\__stex_importmodule_get_-
    module:nnn ..... 3112, 3118, 3176
\__stex_importmodule_get_module_-
    safe:nnn ..... 3127, 3133, 3182
\__stex_importmodule_get_module_-
    uri:nnn ..... 3137, 3178
\__stex_importmodule_get_module_-
    uri_safe:nnn ..... 3156, 3184

```

```

\__stex_importmodule_import_-
    module:nn ..... 3265, 3270, 3310
\__stex_importmodule_import_-
    module_presms:nn ..... 3296, 3310
\__stex_importmodule_load_file:n
    ..... 3208, 3211, 3235, 3240, 3245
\l__stex_importmodule_path_seq ..
    ..... 3087, 3088, 3090
\__stex_importmodule_seq .....
    ..... 3072, 3073, 3074
\l__stex_importmodule_seq .....
    .. 3094, 3096, 3188, 3189, 3215, 3216
\l__stex_importmodule_str .....
    ..... 3117, 3118, 3132,
        3133, 3189, 3205, 3216, 3231, 3234,
            3239, 3246, 3250, 3258, 3259, 3261
\__stex_inputs_bibresource:n ...
    ..... 1946, 1958, 1960
\l__stex_inputs_gin_repo_str ...
    ..... 2032, 2062, 2066, 2071
\l__stex_inputs_id_seq .....
    ..... 1982, 1984, 1991
\l__stex_inputs_id_str ... 1977, 1982
\__stex_inputs_inputref:nn 1934, 1944
\__stex_inputs_inputref_html:nn .
    ..... 1899, 1937
\__stex_inputs_inputref_pdf:nn ..
    ..... 1923, 1938
\__stex_inputs_libinput:n .....
    ..... 2003, 2016, 2018
\l__stex_inputs_libinput_files_-
    seq ..... 1948,
        1951, 1980, 1987, 1988, 1997, 1998,
            2005, 2008, 2023, 2024, 2095, 2096
\__stex_inputs_mhinput:nn 1883, 1896
\l__stex_inputs_path_seq .....
    ..... 1981, 1985, 1992, 1995
\l__stex_inputs_path_str .....
    ..... 1985, 1986, 1987, 1988,
        1991, 1992, 1995, 1996, 1997, 1998
\l__stex_inputs_tmp_str .. 2024, 2026
\__stex_inputs_up_archive:nn ...
    ..... 1947, 1973, 2004, 2022, 2094
\__stex_inputs_usetikzlibrary:n .
    ..... 2093, 2107, 2109
\__stex_inputs_usetikzlibrary_-
    i:nn ..... 2077, 2096
\l__stex_iterate_continue_bool ..
    ..... 2737, 2740, 2753, 2773
\__stex_iterate_it_decl_check:nnnn
    ..... 2698, 2705
\__stex_iterate_it_decl_i:n .....
    ..... 2690, 2694, 2707
\__stex_iterate_it_not_check:nnnn
    ..... 2726, 2730
\__stex_iterate_it_not_i:n .....
    ..... 2715, 2719, 2732
\__stex_iterate_iterate_morphism:nn
    ..... 2743, 2747, 2756, 2759
\l__stex_iterate_mods_seq .....
    ..... 2686, 2695, 2696,
        2711, 2720, 2721, 2736, 2761, 2762
\__stex_iterate_morphism_cs:nnnn
    ..... 2738, 2764
\__stex_iterate_not_cs:nnnnn ...
    ..... 2712, 2713, 2723
\__stex_iterate_sym_cs:nnnnnnnn
    .. 2669, 2670, 2680, 2687, 2688, 2701
\l__stex_iterate_todo_tl .....
    ..... 2742, 2746, 2760, 2773
\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
\__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 .....
    1043, 1045, 1047, 1050, 1051, 1055,
        1056, 1061, 1067, 1070, 1075, 1078,
            1208, 1209, 1211, 1218, 1222, 1225
\l__stex_mathhub_tl .....
    ..... 1145
\l__stex_mathhub_val .....
    .. 1179, 1181, 1182, 1183, 1184, 1185
\__stex_module_setup_end: 2350, 2371

```

```

\__stex_module_setup_get_uri_-
    str:n ..... 2300, 2357
\__stex_module_setup_load_meta: .
    ..... 2309, 2378
\__stex_module_setup_load_sig: ..
    ..... 2330, 2340
\l__stex_module_setup_ns_str ...
    ..... 2301, 2303, 2358, 2360, 2367
\l__stex_module_setup_seg .....
    ..... 2363, 2364, 2365
\l__stex_module_setup_seq .....
    ..... 2362, 2363, 2365, 2367
\__stex_module_setup_setup_-
    nested:n ..... 2296, 2348
\__stex_module_setup_setup_top:n
    ..... 2296, 2299
\__stex_module_setup_setup_top_-
    sig:n ..... 2303, 2325
\l__stex_module_setup_sigfile ...
    ..... 2341, 2342, 2344
\__stex_module_setup_split_-
    module:n ..... 2350, 2371
\__stex_modules_activate_-
    i:nnnnnnnn ..... 2604, 2608
\__stex_modules_activate_not:nn .
    ..... 2628, 2634
\__stex_modules_activate_sym:n ..
    ..... 2596, 2601
\__stex_modules_export:n . 2567, 2569
\__stex_modules_persist_module: .
    ..... 2390, 2400, 7477
\__stex_modules_persist_nots_-
    i:nn ..... 2416, 2437
\l__stex_modules_restore_mod_str
    ..... 2430, 2458
\__stex_modules_restore_module:nnnn
    ..... 2402, 2421
\__stex_modules_restore_nots:n ..
    ..... 2434, 2443
\__stex_modules_restore_nots_i:n
    ..... 2444, 2447, 2464
\__stex_modules_restore_nots_-
    ii:nnnnn ..... 2451, 2455
\__stex_modules_set_metatheory:nn
    ..... 2467, 2487
\__stex_modules_tl .....
    ..... 2423, 2424
\l__stex_modules_tl .....
    ..... 2456, 2461
\l__stex_morphisms_ass_tl .....
    ..... 3543, 3556, 3560, 3570, 3571
\__stex_morphisms_do_morph_-
    assign:nnn .....
    ..... 3416, 3419, 3457
\__stex_morphisms_do_parsed_-
    assign: ..... 3562, 3565
\__stex_morphisms_do_parsed_-
    newname: ..... 3568, 3575
\__stex_morphisms_do_parsed_-
    newname:w ..... 3577, 3579, 3583
\__stex_morphisms_end: ... 3568, 3583
\l__stex_morphisms_morphism_dom_-
    str ... 3410, 3423, 3435, 3445, 3459
\l__stex_morphisms_name_str 3541,
    3548, 3549, 3553, 3554, 3555, 3566
\l__stex_morphisms_newname_str ...
    ..... 3542, 3558, 3559, 3567, 3568
\l__stex_morphisms_next_tl .....
    ..... 3546, 3550, 3552
\__stex_morphisms_parse_assign:n
    ..... 3540, 3590, 3610, 3631
\l__stex_morphisms_seq .....
    ..... 3544, 3545, 3546, 3548,
    3550, 3552, 3554, 3556, 3558, 3560
\__stex_notations_add:nnnnn .....
    ..... 4630, 4635, 4639
\__stex_notations_add_last:nnnnn
    ..... 4623, 4634
\__stex_notations_add_missing_-
    args:nn ..... 4199, 4314
\__stex_notations_add_next:nnnnn
    ..... 4625, 4629
\l__stex_notations_after_tl .....
    ..... 4609, 4636
\__stex_notations_args_end: .....
    ..... 4100, 4103, 4106,
    4113, 4116, 4119, 4618, 4621, 4631
\l__stex_notations_args_tl .....
    ..... 4525, 4529, 4542
\__stex_notations_augment_arg:nn
    ..... 4530, 4547
\__stex_notations_check_aB_-
    arg:Nn ..... 4670, 4678, 4687, 4712
\l__stex_notations_clist_count_-
    int ..... 4710, 4718, 4724
\l__stex_notations_code_tl .....
    ... 4566, 4581, 4589, 4597, 4611,
    4612, 4615, 4643, 4651, 4656, 4664
\__stex_notations_complex:nnnnnn
    ..... 4575, 4593
\__stex_notations_const_precs: ...
    ..... 4191, 4233
\l__stex_notations_cs .....
    ..... 4582, 4587, 4598, 4607
\__stex_notations_do_argname:nn .
    ..... 4295, 4298
\__stex_notations_do_argnames: ...
    ..... 4269, 4293
\__stex_notations_fun_precs: ...
    ..... 4196, 4245

```

```

\__stex_notations_make_arg:nnnn . . . . . 4399, 4403
\__stex_notations_make_arg_- html:nn . . . . . 4366, 4381
\__stex_notations_make_name: . . . . . 4527, 4551, 4559
\__stex_notations_map_args_i:w . . . . . 4099, 4103, 4106
\__stex_notations_map_args_ii:w . . . . . 4112, 4116, 4119
\__stex_notations_map_cs: . . . . . 4352, 4354,
     4690, 4692, 4700, 4715, 4717, 4728
\l__stex_notations_missing_str . . . . . 4197, 4315
\l__stex_notations_missing_tl . . . . . 4198, 4227, 4316
\l__stex_notations_name_str . . . . . 4553, 4554, 4555, 4560
\l__stex_notations_opprec_tl 4221,
     4235, 4238, 4240, 4247, 4250, 4252,
     4256, 4263, 4276, 4282, 4288, 4343
\__stex_notations_parse_notation_- args:nnnnw . . . . . 4618, 4621, 4631
\__stex_notations_parse_precs: . . . . . 4267, 4272
\l__stex_notations_pre_tl . . . . . 4596, 4611, 4648, 4661
\l__stex_notations_precs_seq . . . . . 4188, 4258,
     4265, 4286, 4288, 4301, 4307, 4344
\__stex_notations_process_- notation:nnnnnn . . . . . 4565, 4571
\l__stex_notations_seq . . . . . 4274, 4275, 4277, 4278, 4285
\__stex_notations_simple:nnnnn . . . . . 4573, 4579
\l__stex_notations_str . . . . . 4275, 4276, 4277, 4279, 4285, 4286
\__stex_notations_styledefs: . . . . . 4159, 4166
\__stex_path_: . . . . . 688, 697
\l__stex_path_a_seq . . . . . 768, 774, 780
\l__stex_path_a_tl . . . . . 780, 782
\__stex_path_auth:n . . . . . 128, 810, 813, 814,
     815, 818, 819, 864, 880, 885, 939, 941
\l__stex_path_auth_str . . . . . 832, 840, 845, 850
\l__stex_path_b_seq . . . . . 769, 775, 781, 787
\l__stex_path_b_tl . . . . . 781, 782
\__stex_path_canonicalize:N . . . . . 700, 711, 717
\__stex_path_colonslash . . . . . 826, 831, 836
\l__stex_path_do_hooks_pre_tl . . . . .
     1021, 1022, 1034, 1037
\__stex_path_dodots:n . . . . . 722, 726, 732
\l__stex_path_file . . . . . 899, 905, 911, 916
\__stex_path_from_repo_file:NNNNn . . . . .
     892, 895, 898
\l__stex_path_maybein_str . . . . .
     749, 750, 751
\l__stex_path_mod . . . . . 843, 846, 851
\__stex_path_module:n . . . . . 128,
     814, 818, 821, 866, 880, 941, 943, 945
\l__stex_path_name . . . . . 848, 851
\__stex_path_name:n . . . . . 128,
     815, 818, 822, 867, 880, 941, 950, 952
\l__stex_path_path . . . . .
     837, 838, 841, 846, 851, 920, 921
\__stex_path_path:n . . . . .
     128, 813, 818, 820, 865, 880, 941
\__stex_path_relativize:N . . . . . 908, 915
\l__stex_path_return_tl . . . . .
     770, 776, 783, 786, 788
\l__stex_path_seq . . . . . 720, 723, 728, 736,
     737, 740, 793, 794, 795, 800, 801,
     803, 805, 808, 916, 918, 922, 924, 926
\g__stex_path_stack . . . . .
     983, 997, 1008, 1009, 1011, 1014
\l__stex_path_str . . . . .
     690, 693, 695, 696, 697, 699, 702,
     709, 710, 719, 721, 725, 737, 792,
     793, 794, 799, 800, 801, 803, 804,
     805, 974, 976, 978, 1009, 1014, 1015
\l__stex_path_tl . . . . . 918, 922, 924
\l__stex_path_uri . . . . .
     901, 902, 903, 908, 926
\__stex_path_uri_set:NnN . . . . . 827, 873, 877
\__stex_path_uri_set:Nnnnn . . . . .
     840, 845, 850, 862, 870, 904, 910
\__stex_path_uri_use:w . . . . . 880, 886
\l__stex_path_win_drive . . . . .
     689, 694, 701, 703
\__stex_path_win_take:w . . . . . 688, 697
\__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: . . . . . 7150, 7331
\__stex_proof_begin_proof:nn . . . .
     7245, 7273, 7291

```

\l__stex_proof_counter_intarray
..... 7108, 7113,
7116, 7125, 7128, 7137, 7145, 7146,
7154, 7159, 7166, 7172, 7246, 7247
\l__stex_proof_end_list:
... 7215, 7286, 7347, 7357, 7423, 7446
\l__stex_proof_html: . 7219, 7241, 7397
\l__stex_proof_html_env:n
..... 7230, 7251, 7306
\l__stex_proof_in_spfblock_bool
... 7244, 7274, 7285, 7292, 7320,
7323, 7342, 7345, 7347, 7352, 7355,
7363, 7405, 7419, 7441, 7461, 7467
\l__stex_proof_inblock_restore:
..... 7349, 7362, 7463
\l__stex_proof_inc_counter:
..... 7133, 7345, 7436, 7438
\l__stex_proof_inc_counter_bool 7107
\l__stex_proof_insert_number:
..... 7109, 7329, 7436, 7438
\l__stex_proof_make_step_macro:Nnnnn
..... 7403, 7436, 7437, 7438
\l__stex_proof_number_as_string:N
..... 7120, 7316, 7325, 7414
\l__stex_proof_proof_box_tl 7090, 7194
\l__stex_proof_remove_counter:
..... 7162, 7356
\l__stex_proof_start_list:n
..... 7208, 7274, 7329, 7423, 7446
\l__stex_proof_step_html:nn 7385,
7421, 7423, 7429, 7444, 7446, 7450
\l__stex.refs.add_doc_ref:nn
..... 1519, 1541, 1552
\l__stex.refs.default_archive_-
str 1561, 1567, 1576
\l__stex.refs.default_file_str
..... 1562, 1568, 1577
\l__stex.refs.default_title_tl
..... 1563, 1569, 1578
\l__stex.refs.do_autoref:n
..... 1637, 1649, 1660,
1664, 1675, 1689, 1715, 1726, 1734
\l__stex.refs.do_internal_link:nn
..... 1829, 1841, 1847
\l__stex.refs.do_return:nmm
..... 1743, 1757, 1765
\l__stex.refs.do_sref:nn .. 1645, 1774
\l__stex.refs.do_sref_in:n
..... 1653, 1668, 1679, 1685, 1783
\l__stex.refs.do_url_link:nn
..... 1835, 1849
\l__stex.refs.file
... 1603, 1605, 1621, 1623, 1711, 1712
\l__stex.refs_file_str
1695, 1701, 1706, 1711, 1712, 1713,
1714, 1719, 1723, 1725, 1733, 1745
\l__stex.refs_files_seq
..... 1502, 1542, 1547, 1594, 1656
\l__stex.refs_find_uri:n
..... 1581, 1773, 1778
\l__stex.refs_find_uri_in_-
file:nnn 1590, 1595, 1627
\l__stex.refs_find_uri_in_prop_-
file:N 1601, 1610, 1615
\l__stex.refs_id_str
..... 1721, 1756, 1762, 1763
\l__stex.refs_iow
..... 1493, 1494, 1495, 1521
\l__stex.refs_new_symbol:n 1787, 1795
\l__stex.refs_prop 1609, 1610
\l__stex.refs_restore_target:nnnnn
..... 1722, 1754
\l__stex.refs_return_tl
..... 1720, 1724, 1730, 1744
\l__stex.refs_set_keys_b:n
..... 1565, 1652, 1667, 1678, 1779
\l__stex.refs_str 1508, 1510, 1512,
1517, 1519, 1524, 1797, 1799, 6922
\l__stex.refs_sym_aux:nn
..... 1826, 1839, 1844, 1855
\l__stex.refs_unnamed_counter_-
int 1503, 1507, 1508
\l__stex.refs_uri
..... 1517, 1518, 1622, 1624
\l__stex.refs_uri_str
... 1582, 1593, 1605, 1616, 1621,
1624, 1631, 1646, 1656, 1657, 1658,
1659, 1660, 1663, 1664, 1666, 1672,
1674, 1675, 1677, 1687, 1688, 1689,
1716, 1727, 1735, 1755, 1760, 1761
\l__stex_seqs_add: 4964, 4982
\l__stex_seqs_args_tl ... 5040, 5042
\l__stex_seqs_check_terms: 4963, 5010
\l__stex_seqs_clist
..... 5077, 5084, 5091, 5095
\l__stex_seqs_cs
..... 5038, 5042, 5098, 5102,
5110, 5113, 5122, 5129, 5142, 5143
\l__stex_seqs_do_all:w ... 5136, 5146
\l__stex_seqs_do_first: ... 5059, 5120
\l__stex_seqs_do_first_arg:n
..... 5118, 5125
\l__stex_seqs_do_first_next:
..... 5127, 5132
\l__stex_seqs_do_one:w ... 5134, 5140
\l__stex_seqs_do_op:w ... 5058, 5062
\l__stex_seqs_doop_arg:n .. 5078, 5089

```

\__stex_seqs_doop_range:w  5071, 5075
\__stex_seqs_first_args_tl . . . . .
. . . . . 5124, 5142, 5143, 5147, 5148
\__stex_seqs_get_index_notation:n
. . . . . 5070, 5108, 5141
\__stex_seqs_html: . . . . . 4962, 5012
\__stex_seqs_macro: . . . . . 4965, 4996
\__stex_seqs_make_args: . . . 4975, 5009
\__stex_seqs_range_clist . . . .
. . . . . 4955, 4990, 5003
\g__stex_smsemode_allowed_escape_tl . . .
. . . . . 2114, 2121, 2250
\g__stex_smsemode_allowed_import_env_seq 2138, 2144, 2184, 2233, 2236
\g__stex_smsemode_allowed_import_tl . . .
. . . . . 2137, 2140, 2181, 2228
\g__stex_smsemode_allowed_tl . . .
. . . . . 2113, 2117, 2246
\g__stex_smsemode_allowedenvs_seq . . .
. . . . . 2115, 2125, 2255, 2258
\g__stex_smsemode_bool . . .
. . . . . 2132, 2133, 2135, 2151
\__stex_smsemode_check_begin:Nn . . .
. . . . . 2233, 2255, 2267
\__stex_smsemode_check_cs:NNn . . .
. . . . . 2206, 2214
\__stex_smsemode_check_end:Nn . . .
. . . . . 2236, 2258, 2276
\__stex_smsemode_do:w . . .
. . . . . 2200, 2212, 2214, 2216,
2238, 2248, 2260, 2273, 2280, 2283
\__stex_smsemode_do_aux:N . 2214, 2220
\__stex_smsemode_do_aux_curr:N . . .
. . . . . 2180, 2191, 2222
\__stex_smsemode_do_aux_imports:N . . .
. . . . . 2180, 2226
\__stex_smsemode_do_aux_normal:N . . .
. . . . . 2191, 2244
\__stex_smsemode_importmodules_seq . . .
. . . . . 2169
\__stex_smsemode_in_smsemode:n . . .
. . . . . 2149, 2179, 2189, 2190
\__stex_smsemode_sigmodules_seq 2170
\__stex_smsemode_smsemode_do: . . .
. . . . . 2162, 2198
\__stex_smsemode_start_smsemode:n . . .
. . . . . 2160, 2187, 2192
\__stex_statements_do_defref:nn . . .
. . . . . 6957, 6986, 6992, 7000
\__stex_statements_force_id: . . .
. . . . . 6816, 6919, 6927, 6950
\__stex_statements_html_keyvals:nn . . .
. . . . . 6846, 6867, 6895
\__stex_statements_setup:nn . . .
. . . . . 6808, 6928, 6933, 6947, 6951, 6954
\__stex_statements_setup_def: . . .
. . . . . 6906, 6929, 6952
\__stex_statements_uri_str . . .
. . . . . 6831, 6835, 6836, 6841, 6842, 7012,
7015, 7018, 7020, 7065, 7068, 7071
\__stex_structures_assigned_seq . . .
. . . . . 6494, 6605, 6644, 6660
\__stex_structures_begin:nn . . .
. . . . . 6087, 6108, 6185, 6240
\__stex_structures_check_def:nnnnnnnn . . .
. . . . . 6259, 6272
\__stex_structures_clist . . .
. . . . . 6468, 6489, 6495, 6497
\__stex_structures_comp_cs . . .
. . . . . 6537, 6542
\__stex_structures_cs . . .
. . . . . 6470, 6476, 6483
\__stex_structures_current_type: . . .
. . . . . 6424, 6555, 6626
\__stex_structures_current_type_tl . . .
. . . . . 6351, 6387, 6427, 6432
\__stex_structures_do_assign:nn . . .
. . . . . 6371, 6375
\__stex_structures_do_assign_list:n . . .
. . . . . 6347, 6368
\__stex_structures_do_decl:nnnnnnnn . . .
. . . . . 6682, 6702
\__stex_structures_do_decl_nomacro:nnnnnnnn . . .
. . . . . 6680, 6688
\__stex_structures_do_externals: . . .
. . . . . 6091, 6133, 6200, 6262
\__stex_structures_end: . . .
. . . . . 6315, 6319, 6335, 6340
\__stex_structures_exstruct_name_str . . .
. . . . . 6235, 6240, 6284
\__stex_structures_extend_structure:nn . . .
. . . . . 6212, 6235
\__stex_structures_extend_structure_i:NnnnnnnnN . 6209, 6217
\__stex_structures_external_decl:nnnn . . .
. . . . . 6136, 6140
\__stex_structures_extmod_str . . .
. . . . . 6210, 6214
\__stex_structures_extname_count . . .
. . . . . 6278, 6282, 6284
\__stex_structures_field_name_str . . .
. . . . . 6592, 6596, 6597, 6628
\__stex_structures_fields_clist . . .
. . . . . 6348,
6369, 6376, 6394, 6397, 6651, 6652
\__stex_structures_get_field_name:n . . .
. . . . . 6591, 6603

```

```

\l__stex_structures_imports_seq .
    ..... 6173, 6178, 6186,
    6223, 6228, 6242, 6764, 6767, 6770
\l__stex_structures_invocation_-
    type:n ..... 6299, 6346
\l__stex_structures_invoke_-
    field:n ..... 6569, 6601
\l__stex_structures_invoke_maybe_-
    field:nn ..... 6558, 6562
\l__stex_structures_invoke_this:n
    ..... 6308, 6550
\l__stex_structures_invoke_top:n .
    ..... 6289,
    6292, 6316, 6320, 6323, 6326, 6328
\l__stex_structures_make_mod:n ...
    ..... 6393, 6405
\l__stex_structures_make_oml:n ...
    ..... 6397, 6411
\l__stex_structures_make_oml:nn ...
    ..... 6412, 6414
\l__stex_structures_make_prop: ...
    ..... 6436, 6563, 6641
\l__stex_structures_make_prop_-
    assign: ..... 6437, 6566, 6650
\l__stex_structures_make_prop_-
    assign:nn ..... 6653, 6658
\l__stex_structures_make_prop_-
    assign_replace:nnnn ... 6661, 6667
\l__stex_structures_make_type:n ..
    ..... 6356, 6361, 6363, 6379
\l__stex_structures_maybe_-
    notation:w ..... 6303, 6305, 6431
\l__stex_structures_merge:nw ...
    ..... 6297, 6313
\l__stex_structures_more_-
    nextsymbol_tl ... 6604, 6606, 6631
\l__stex_structures_name_str 6079,
    6096, 6101, 6103, 6110, 6111, 6116,
    6117, 6122, 6123, 6126, 6142, 6148
\l__stex_structures_new_extstruct_-
    name: ..... 6222, 6280
\l__stex_structures_present: ...
    ..... 6442, 6567
\l__stex_structures_present:nn ...
    ..... 6446, 6452, 6457, 6464, 6467
\l__stex_structures_present_-
    entry:nn ..... 6472, 6478, 6493
\l__stex_structures_present_i:w ..
    ..... 6438, 6444, 6450
\l__stex_structures_present_ii:nw
    ..... 6455, 6463
\l__stex_structures_prop .....
    6482, 6594, 6602, 6634, 6642, 6659,
    6662, 6668, 6689, 6691, 6703, 6705
\l__stex_structures_prop_do_-
    decls: ..... 6646, 6677
\l__stex_structures_prop_do_-
    notations: ..... 6647, 6726
\l__stex_structures_redo_tl ...
    .. 6621, 6645, 6671, 6718, 6729, 6744
\l__stex_structures_replace_-
    this_tl ..... 6134
\l__stex_structures_seq .....
    ..... 6643, 6690, 6704, 6728
\l__stex_structures_set_comp_tl ...
    ..... 6288, 6337, 6341, 6499, 6616
\l__stex_structures_set_custom_-
    comp:n ..... 6342, 6531
\l__stex_structures_set_customcomp:
    ..... 6315, 6340
\l__stex_structures_set_this:n ...
    ..... 6564, 6573
\l__stex_structures_set_thiscomp:
    ..... 6288, 6525
\l__stex_structures_set_thisnotation:
    ..... 6319, 6335
\l__stex_structures_shift_-
    argls:nn ..... 6149, 6154
\l__stex_structures_this_tl ...
    6300, 6500, 6501, 6504, 6519, 6576,
    6579, 6586, 6607, 6608, 6611, 6625
\l__stex_symdecl_add_decl: 3745, 3751
\l__stex_symdecl_args_tl . 3798, 3800
\l__stex_symdecl_cs .....
    .. 3796, 3800, 3990, 3992, 3994, 4020
\l__stex_symdecl_do_args: . 3815, 3874
\l__stex_symdecl_env_str .....
    ..... 3653, 3654, 3655
\l__stex_symdecl_get_from_one_-
    string:n ..... 4029, 4074
\l__stex_symdecl_get_symbol_from_-
    cs: ..... 3996, 4007
\l__stex_symdecl_get_symbol_from_-
    modules:nn ..... 4031, 4063
\l__stex_symdecl_get_symbol_from_-
    string:n ... 3997, 3999, 4002, 4024
\l__stex_symdecl_name ... 4027, 4033
\l__stex_symdecl_parse_arity: ...
    ..... 3814, 3820
\l__stex_symdecl_seq .....
    ..... 4026, 4027, 4028, 4032
\l__stex_symdecl_set_textsymdecl_-
    macro:nnn ..... 3947, 3965
\l__stex_symdecl_sym_from_str_-
    i:nnnn ..... 4037, 4068
\l__stex_symdecl_sym_i_finish:nnnnnnN
    ..... 4043, 4050

```

```

\__stex_symdecl_sym_i_gobble:nnnnnn  \stexstylemodule ..... 94, 124
..... 4045, 4048 \stexstylenotation ..... 94
\__stex_vars_add: ..... 4759, 4787 \stexstyleparagraph ..... 94
\l__stex_vars_args_tl ... 4847, 4849 \stexstyleproof ..... 94
\l__stex_vars_bind_bool ..... 4742, 4745, 4747, 4831 \stexstylerealization ..... 94
\l__stex_vars_check_var:nnnnnnnnN \stexstylerealize ..... 94
..... 4897, 4901 \stexstylerenamedecl ..... 94
\l__stex_vars_cs ..... 4845, 4849 \stexstylerequiremodule ..... 94
\l__stex_vars_get_var:n ..... 4895, 4913, 4921 \stexstylespfsketch ..... 94
\l__stex_vars_html: ..... 4761, 4814 \stexstylesubproof ..... 94
\l__stex_vars_macro: ..... 4760, 4801 \stexstylesymdecl ..... 94
\l__stex_vars_set_vars:nnnnnnN ... 4882, 4903, 4906 \stexstylesymdef ..... 94
\sTeX/ComputerScience/Software .... 10 \stexstyletextsymdecl ..... 94
stex_annotation_env (env.) ..... 131 \stexstyleusemodule ..... 94
\stexcommentfont ..... 7205, 7275, 7342, 7364, 7427, 7448 \stexstylevardef ..... 94
\stexdoctitle ..... 1241, 1300, 1312, 2516, 7995, 8135, 8318 \stexstylevarannotation ..... 94
\STEXexport ..... 44, 76, 77, 111, 2565 \stexstylevarseq ..... 94
\stexhtmlfalse ..... 324 \stopsolutions ..... 101, 8438
\stexhtmltrue ..... 321 str commands:
\stexInternalAssocArgMarkerI ..... 117
\stexInternalAssocArgMarkerII ..... 117
\stexInternalNotation .. 4219, 4542, 4562
\stexInternalSetSrefSymURL ..... 1498, 1799, 1802, 1805
\stexInternalSrefRestoreTarget ..... 1496, 1522, 1722
\stexInternalSymbolAfterInvocationTL ..... 117, 121
\stexInternalTermMathArgiii ..... 117
\stexInternalTermMathAssocArgiiii ..... 117
\stexInternalTermMathOMAiii ..... 117
\stexInternalTermMathOMBiii ..... 117
\stexInternalTermMathOMSOrOMViii .. 117
\stexInvisible . 71, 4316, 5180, 7255, 7310
\stexRestoreNotsEnd ..... 2417, 2441, 2448
\stexstyle ..... 123
\stexstyleassertion ..... 94
\stexstyleassign ..... 94
\stexstyleassignMorphism ..... 94
\stexstylecopymod ..... 94
\stexstylecopymodule ..... 94
\stexstyledefinition ..... 94
\stexstyleexample ..... 94
\stexstyleextstructure ..... 94
\stexstyleimportmodule ..... 94, 123
\stexstyleinterpretmod ..... 94
\stexstyleinterpretmodule ..... 94
\stexstylemathstructure ..... 94
\stexstyleMMTinclude ..... 94

```

6110, 6159, 6166, 6596, 6809, 6810,
 6815, 6832, 6852, 6920, 7015, 7068,
 7315, 7324, 7393, 7407, 7413, 7638,
 7764, 7823, 7986, 8300, 8390, 8430,
 8447, 8487, 8496, 8536, 8545, 8585
`\str_if_empty:nTF` 471, 498,
 682, 733, 828, 1506, 1860, 2495, 3396
`\str_if_eq:NNTF` 171, 782
`\str_if_eq:nnTF`
 145, 161, 162, 163, 185,
 301, 534, 539, 569, 585, 594, 606,
 612, 696, 734, 735, 751, 1630, 1714,
 1756, 1761, 1763, 1817, 2282, 2364,
 2603, 2636, 2741, 3012, 3017, 3026,
 3029, 3205, 3231, 3415, 3431, 3655,
 4237, 4249, 4261, 4902, 4905, 5096,
 7032, 7827, 7830, 7895, 8054, 8617
`\str_if_eq_p:nn` 4039, 4040, 4079, 4080
`\str_if_exist:NTF` 232, 1843
`\str_if_in:NnTF` 4315, 5998, 6964
`\str_item:Nn` 696, 751, 3879
`\str_lowercase:n` 8617
`\str_map_break:` 3825
`\str_map_break:n` 3826, 3830, 3834,
 3838, 3842, 3846, 3850, 3854, 3858
`\str_map_inline:Nn` 3823
`\str_new:N`
 134, 1561, 1562, 2032, 2287, 3713
`\str_put_right:Nn` 7128
`\str_range:Nnn` 2026
`\str_range:nnn` 535, 540, 574
`\str_replace_all:Nnn` 695
`\str_set:Nn` 47,
 55, 136, 184, 227, 689, 690, 693,
 709, 826, 1083, 1166, 1178, 1179,
 1508, 1510, 1512, 1576, 1577, 1605,
 1616, 1624, 1631, 1695, 1701, 1706,
 1712, 1721, 1862, 1869, 1874, 1985,
 1995, 2024, 2062, 2321, 2337, 2360,
 2430, 2659, 2813, 2818, 2821, 2832,
 2855, 2856, 3039, 3040, 3041, 3074,
 3079, 3084, 3089, 3095, 3103, 3105,
 3111, 3115, 3116, 3117, 3126, 3130,
 3131, 3132, 3139, 3146, 3149, 3152,
 3158, 3165, 3168, 3171, 3189, 3216,
 3261, 3459, 3549, 3555, 3559, 3689,
 3719, 3741, 3828, 3832, 3836, 3840,
 3844, 3848, 3852, 3856, 3860, 3909,
 3911, 3913, 3915, 4011, 4012, 4052,
 4053, 4084, 4085, 4136, 4168, 4197,
 4202, 4321, 4361, 4372, 4498, 4752,
 4754, 4886, 4930, 4932, 4945, 4947,
 4950, 5205, 5206, 5619, 5663, 5704,
 5996, 5999, 6078, 6101, 6111, 6120,
 6214, 6278, 6282, 6284, 6592, 6597,
 6821, 6835, 6841, 6962, 6965, 7012,
 7065, 7317, 7326, 7415, 7637, 7639,
 8388, 8389, 8392, 8445, 8446, 8449,
 8494, 8495, 8498, 8543, 8544, 8547
`\str_set_eq:NN` 1067,
 1567, 1568, 2532, 2803, 2833, 3938,
 3939, 4138, 4167, 4458, 4468, 4487,
 4501, 4502, 4776, 4971, 5960, 5973,
 5986, 6103, 6168, 6811, 6820, 6870,
 6922, 7301, 7382, 7600, 7604, 7628,
 7630, 7643, 7657, 7659, 7660, 8322
`\str_uppercase:n` 7895
`\l_tmpa_str` 158, 159, 160,
 161, 162, 163, 164, 168, 184, 188,
 189, 190, 192, 1900, 1901, 1902,
 1907, 1915, 2818, 2821, 2826, 2833
`\subparagraph` 22, 72
`subproof (env.)` 7301
`\subproof` 7259, 7360
`\subproofautorefname` 7301
`\subsection` 21, 22, 72
`\subsubsection` 22, 72
`\svar` 85, 3767, 4927
`\symbol` 79
`\symdecl` 18,
 19, 26, 27, 34, 35, 42, 77–79, 89,
 94, 114, 119, 125, 132, 133, 2981,
 3713, 7501, 7503, 7532, 7540, 7543
`\symdef` 27, 28, 30, 35, 36, 78,
 85, 114, 119, 125, 2983, 4478, 7485,
 7488, 7494, 7496, 7498, 7500, 7510,
 7511, 7512, 7513, 7519, 7527, 7538,
 7547, 7548, 7551, 7554, 7556, 7558
`\Symname` 79, 90, 5902
`\symname` 14, 15, 19, 57, 79, 80, 90, 92, 5902
`\symref` 14, 15, 19, 43, 78, 79, 83, 90, 92, 5902
`\symrefemph` 92, 93, 6014
`\symuse` 43, 79, 5151, 5230, 5707, 6388,
 6395, 6406, 6488, 6553, 6623, 6624
`sys commands:`
`\sys_get_shell:nnN` 44
`\sys_if_platform_windows:TF`
 50, 687, 746, 973, 1042

T
`\target` 124, 127–129
`\test` 106
`test` 100
`\testemptypage` 65, 105
`\testemptypage` 8698
`testheading (env.)` 106
`\testheading` 8826
`\testnewpage` 65, 105

| | |
|--|---|
| \testnewpage | 8702 |
| \testsmallspace | 65, 65, 65, 105, 105, 105 |
| \testspace | 65, 105 |
| \testspace | 8701 |
| \TeX | 18, 19 |
| \tex | 18, 19 |
| TeX and L ^A T _E X 2 _E commands: | |
| \@ | 2081, 2084, 2088 |
| \@addtoreset | 8282 |
| \@arabic | 8146 |
| \@author | 8125 |
| \@auxout | 1792, 7680, 8364 |
| \@bonuspointsfalse | 8857 |
| \@bonuspointstrue | 8862 |
| \@currentHref | 7317, 7326, 7415 |
| \@currentcounter | 1525 |
| \@currentlabel | 1526, 7316, 7317, 7325, 7326, 7414, 7415, 7985 |
| \@currentlabelname | 1528, 1529 |
| \@currenvir | 7869, 7870 |
| \@dbllarg | 8118, 8127 |
| \@gobble | 360 |
| \@ifnextchar | 359 |
| \@ifstar | 8097 |
| \@mainmatterfalse | 1455, 1465 |
| \@mainmattertrue | 1476, 1487 |
| \@notprerr | 139, 1421 |
| \@onlypreamble | 139, 1421 |
| \@rustexfalse | 297 |
| \@title | 1270, 1271, 8134 |
| \activate@excursion | 8158, 8163 |
| \bb@loaded | 8250, 8721 |
| \beamer@shortauthor | 8121, 8123, 8138 |
| \beamer@shorttitle | 8130, 8132, 8141 |
| \c@chapter | 7850 |
| \c@framenumber | 8146 |
| \c@part | 7862 |
| \compemph@uri | 93, 121, 4358, 6006, 6014 |
| \correction@table | 8805 |
| \defemph@uri | 93, 6014 |
| \define@key | 2034, 2048, 2061 |
| \Gin@eheight | 8898, 8901, 8906, 8911 |
| \Gin@ewidth | 8088, 8089, 8897, 8907, 8911 |
| \Gin@exclamation | 8897, 8898, 8906 |
| \Gin@mrepos | 2035, 2038, 2042, 2063, 2067, 2072 |
| \hwexam@bonuspts | 8859 |
| \hwexam@checktime | 8853 |
| \hwexam@duration | 8829, 8832, 8838, 8843 |
| \hwexam@kw@due | 8780 |
| \hwexam@kw@forgrading | 8821 |
| \hwexam@kw@given | 8777 |
| \hwexam@kw@grade | 8822 |
| \hwexam@kw@probs | 8822 |
| \hwexam@kw@pts | 8823 |
| \hwexam@kw@reached | 8824 |
| \hwexam@kw@sum | 8822 |
| \hwexam@kw@testemptypage | 8699 |
| \hwexam@min | 8830, 8837, 8842, 8853 |
| \hwexam@minutes@kw | 8830 |
| \hwexam@reqpts | 8839, 8844, 8856, 8860 |
| \hwexam@tools | 8840, 8845 |
| \hwexam@totalmin | 8852, 8853 |
| \hwexam@totalpts | 8851, 8860 |
| \if@bonuspoints | 8855 |
| \if@rustex | 305 |
| \itemize@inner | 7935, 7941, 7950 |
| \itemize@label | 7939, 7942, 7945 |
| \itemize@level | 7933, 7938, 7941, 7950 |
| \itemize@outer | 7934, 7938 |
| \lst@mhrepos | 2049, 2052, 2056 |
| \ltx@ifpackageloaded | 140, 144, 2033, 2047, 2060, 7195, 8249, 8639, 8720 |
| \m@switch | 5873 |
| \mdf@patchamsthdm | 7959 |
| \metakeys@show@keys | 7936 |
| \notesslides@slidelabel | 7962, 7977 |
| \ns@author | 8118, 8119 |
| \ns@title | 8127, 8128 |
| \pgf@temp | 2078, 2079, 2080 |
| \pgfkeys@spdef | 2078 |
| \pgfutil@empty | 2080 |
| \pgfutil@InputIfFileExists | 2087 |
| \prematrestop@endsfragment | 7868, 7871, 7877 |
| \problem@kw@* | 8246 |
| \problem@kw@correct | 8647 |
| \problem@kw@minutes | 8345, 8598 |
| \problem@kw@points | 8342, 8593 |
| \problem@kw@solution | 8430 |
| \problem@kw@wrong | 8649 |
| \problem@restore | 8365, 8369, 8799 |
| \stex@backend | 130, 295 |
| \symrefemph@uri | 92, 93, 5918, 5928, 5939, 6014 |
| \varemph@uri | 93, 121, 5959, 5972, 5985, 6014 |
| \texname | 19 |
| \text | 15 |
| \textbf | 14, 6056, 7755, 8356, 8770, 8775 |
| \textcolor | 8685 |
| \textsymdecl | 18, 19, 78, 119, 2982, 3895 |
| \textwarning | 98 |
| \textwidth | 8086, 8108, 8818 |
| \the | 250, 251, 253, 255, 257, 2081, 2082, 2083 |
| \theassignment | 8763, 8770 |
| \thechapter | 1331, 1372, 1400, 1435, 7781, 7786, 7790, 7794, 7798, 7802 |

```

\theframenumber ..... 7985
\theparagraph ..... 7798, 7802
\thepart .... 1327, 1369, 1396, 1429, 7776
\theplainsproblem ..... 8286, 8287
\thesection ....
    .. 7786, 7790, 7794, 7798, 7802, 8287
\thesproblem . 8283, 8287, 8356, 8365, 8763
\thesubparagraph ..... 7802
\thesubsection ... 7790, 7794, 7798, 7802
\thesubsubsection .... 7794, 7798, 7802
\this .....
    .. 52–54, 86, 87, 5202,
        6075, 6504, 6581, 6582, 6586, 6611
\thisarchive ..... 3287, 3328
\thisargs ..... 3729, 4485
\thiscopyname .....
    .. 3479, 3499, 3523, 3593, 3613, 3634
\thisdeclname .... 3726, 3954, 4168, 4482
\thisdecluri .....
    .. 3725, 3953, 4169, 4172, 4481, 4489
\thisdefiniens ..... 3728, 4484
\thisfor ..... 6871
\thismodulename .....
    .. 123, 2533, 3064, 3289, 3330, 8323
\thismoduleuri .....
    .. 123, 2532, 3063, 3288, 3329, 3480,
        3500, 3524, 3594, 3614, 3635, 8322
\thisname ..... 6870
\thisnotation 4170, 4469, 4488, 4777, 4972
\thisnotationvariant .....
    .. 4167, 4468, 4487, 4776, 4971
\ThisStyle ..... 5873
\thisstyle .....
    .. 123, 458, 461,
        462, 463, 509, 512, 513, 514, 515,
        523, 526, 527, 3065, 3290, 3331,
        3730, 3955, 4467, 4486, 4775, 4970
\thistitle ..... 93, 123, 2514,
    2515, 2516, 6869, 8324, 8357, 8358
\thistype ..... 3727, 4483
\thisvarname .....
    .. 4466, 4470, 4774, 4778, 4969, 4973
\throwaway ..... 125
\tikzinput ... 108, 2072, 8890, 8895, 8921
tikzinput commands:
    \c_tikzinput_image_bool 34, 8879, 8886
\tiny ..... 7963, 7977
\ttitle ..... 106
\ttitle ..... 8127
tl commands:
    \tl_clear:N ..... 399,
        458, 509, 770, 1343, 1720, 2171,
        2290, 2496, 2760, 3065, 3290, 3331,
        3543, 3696, 3697, 3698, 3730, 3875,
        3898, 3899, 3935, 3955, 4189, 4198,
        4294, 4437, 4467, 4486, 4525, 4720,
        4775, 4935, 4958, 4970, 5200, 5210,
        5307, 5320, 5518, 5905, 5906, 5907,
        6004, 6095, 6576, 6604, 6645, 6785,
        6786, 6787, 7088, 7089, 7091, 7092,
        7093, 7094, 7371, 7372, 8173, 8624,
        8626, 8627, 8738, 8739, 8740, 8810,
        8811, 8812, 8837, 8838, 8839, 8840
\tl_const:Nn ..... 5845, 5846
\tl_count:N ..... 5462, 5468
\tl_count:n ..... 5593, 5604
\tl_gclear:N ..... 2316
\tl_gput_left:Nn ..... 371, 373, 374
\tl_gput_right:Nn ..... 251, 2117,
    2121, 2140, 2293, 2557, 3298, 8159
\tl_gset:Nn ..... 253,
    366, 367, 1022, 1242, 1249, 2141,
    2145, 4205, 5163, 5167, 8802, 8803
\tl_gset_eq:NN ..... 45, 2792
\tl_head:N .. 885, 939, 3994, 6526, 6532
\tl_head:n ..... 553, 594, 597, 4671, 5594, 5605
\tl_if_empty:NTF .. 523, 786, 1024,
    1260, 1270, 1313, 1747, 2379, 2515,
    2525, 3348, 3570, 3757, 3788, 3791,
    3794, 3923, 4098, 4111, 4183, 4192,
    4200, 4370, 4524, 4837, 4840, 4843,
    4851, 4873, 4956, 5030, 5033, 5036,
    5044, 5274, 5286, 5377, 5397, 5482,
    5621, 5734, 5747, 5789, 5818, 6077,
    7201, 7221, 7224, 7612, 8089, 8180,
    8312, 8317, 8357, 8647, 8649, 8651,
    8758, 8771, 8776, 8779, 8829, 8856
\tl_if_empty:nTF ... 368, 552, 568,
    584, 593, 750, 759, 881, 882, 944,
    951, 1098, 1566, 1724, 1746, 1904,
    1926, 1957, 2015, 2106, 2213, 2579,
    2594, 2646, 2706, 2731, 2801, 2816,
    2915, 2974, 3075, 3109, 3124, 3138,
    3157, 3188, 3215, 3404, 3992, 4105,
    4118, 4572, 4622, 4929, 5425, 5446,
    5522, 5545, 5948, 6149, 6273, 6322,
    6325, 6370, 6380, 6469, 6471, 6565,
    6574, 6669, 6679, 6743, 6888, 7010,
    7063, 7079, 7575, 8120, 8129, 8688
\tl_if_empty_p:N ..... 776
\tl_if_eq:NNTF 1140, 6167, 6526, 6532
\tl_if_eq:NnTF ..... 8206
\tl_if_eq:nnTF .....
    .. 2448, 4696, 5090, 5605, 5650, 5688
\tl_if_exist:NTF ..... 209,
    250, 265, 304, 462, 514, 526, 1286,
    1295, 1528, 1688, 2549, 6234, 7807
\tl_if_in:NnTF ..... 2228, 2246, 2250
\tl_item:nn ..... 5596

```

```

\tl_map_inline:Nn ..... 2181
\tl_map_inline:nn ..... 216, 220
\tl_new:N ..... 964, 1034, 1240, 1563, 2113, 2114, 2137, 2148, 2466, 4741, 5156, 5160
\tl_put_left:Nn ..... 4611, 4612, 5351, 8292, 8751
\tl_put_right:Nn ..... 1216, 2367, 2742, 2746, 3877, 3878, 4300, 4306, 4316, 4532, 4537, 4540, 4643, 4651, 4656, 4664, 4727, 5188, 5212, 5222, 5332, 6575, 6671, 6718, 6729, 6744, 8814, 8815, 8816
\tl_range:nnn ..... 570, 586
\tl_set:Nn ..... 210, 247, 350, 453, 472, 474, 493, 494, 499, 500, 502, 503, 749, 783, 863, 957, 1021, 1278, 1319, 1453, 1463, 1578, 1744, 2035, 2038, 2533, 2642, 2644, 2647, 2862, 3043, 3044, 3045, 3046, 3047, 3287, 3328, 3479, 3499, 3523, 3550, 3556, 3560, 3593, 3613, 3634, 3725, 3798, 3953, 4014, 4015, 4016, 4017, 4018, 4055, 4056, 4057, 4058, 4059, 4087, 4088, 4089, 4090, 4091, 4169, 4184, 4193, 4218, 4235, 4238, 4247, 4250, 4256, 4263, 4282, 4481, 4529, 4542, 4560, 4581, 4596, 4597, 4609, 4648, 4661, 4680, 4693, 4719, 4802, 4847, 4888, 4889, 4890, 4891, 4892, 4997, 5040, 5124, 5157, 5162, 5164, 5171, 5201, 5207, 5208, 5209, 5321, 5342, 5462, 5468, 5481, 5487, 5490, 5493, 5536, 5622, 5654, 5656, 5706, 5849, 5850, 5894, 5895, 6134, 6146, 6215, 6288, 6300, 6336, 6337, 6341, 6344, 6351, 6387, 6426, 6500, 6506, 6516, 6579, 6581, 6606, 6607, 6613, 6622, 6670, 6672, 6714, 6719, 6731, 6734, 6738, 6741, 6746, 6749, 6753, 6756, 7194, 7200, 7776, 7777, 7781, 7782, 7786, 7787, 7790, 7791, 7794, 7795, 7798, 7799, 7802, 7803, 7818, 8267, 8268, 8355, 8379, 8638, 8645, 8655, 8790, 8796, 8797, 8853, 8859
\tl_set_eq:NN ..... 349, 370, 372, 377, 1569, 2514, 3063, 3064, 3288, 3289, 3329, 3330, 3480, 3500, 3524, 3594, 3614, 3635, 3726, 3727, 3728, 3729, 3954, 4172, 4240, 4252, 4276, 4424, 4432, 4466, 4482, 4483, 4484, 4485, 4528, 4765, 4774, 4865, 4870, 4874, 4954, 4969, 5401, 5541, 5582, 5661, 6432, 6504, 6586, 6611, 6869, 7090, 8323, 8324, 8851, 8852
\tl_set_rescan:Nnn ..... 138, 189
\tl_tail:n ..... 553, 554, 599, 2213, 5567, 6396
\tl_to_str:n .. 65, 68, 96, 99, 109, 121, 217, 221, 222, 246, 349, 571, 587, 683, 705, 712, 811, 829, 831, 834, 853, 859, 910, 980, 1072, 1176, 1248, 1263, 1499, 1786, 1811, 1812, 1840, 1841, 1847, 1896, 1944, 2104, 2125, 2141, 2144, 2145, 2182, 2185, 2227, 2229, 2245, 2247, 2251, 2269, 2278, 2422, 2424, 2425, 2428, 2429, 2459, 2460, 2518, 2586, 2587, 2588, 2610, 2843, 2997, 3072, 3347, 3356, 3544, 4279, 4580, 4594, 4673, 4718, 5507, 6144, 6177, 6227, 6294, 6314, 6318, 6477, 6605, 6660, 6690, 6704, 6728, 6766, 6802, 7677, 8250, 8721
\tl_trim_spaces:N ..... 48
\l_tmpa_tl ..... 44, 45, 47, 4310, 4720, 4727, 4734, 5151, 5449, 5455, 5460, 5462, 5466, 5468, 5471, 5480, 5482, 5487, 5490, 5493, 5708, 6384, 8655, 8667, 8668, 8810, 8814, 8822
\l_tmpb_tl ..... 5449, 5455, 5457, 5471, 5481, 5485, 8645, 8658, 8660, 8811, 8815, 8823
tmpc commands:
\l_tmpc_t1 ..... 8812, 8816, 8824
\to ..... 7520, 7523, 7533, 7534
\today ..... 8148
\TODO ..... 5009, 5010, 6371, 8204
todo internal commands:
\l__todo_mmt_module_str 7600, 7605, 7618, 7621, 7624, 7628, 7643, 7659
\l__todo_newlabel:n ..... 7676, 7683
\l__todo_old_metagroup_cd 7644, 7656
\l__todo_old_newlabel: ... 7677, 7682
\l__todo_stex_module_str ..... 7604, 7605, 7630, 7657
token commands:
\c_math_subscript_token ..... 44, 77, 4406, 4407, 4410, 4411, 4415, 6516, 7525, 7536, 7538
\topsep ..... 7210
\type ..... 106

```

U

```

\undefined ..... 124, 40
\univ ..... 58
\unless ..... 7869
\uppercase ..... 5942
\uri ..... 128, 129

```

| | |
|----------------------------|--|
| use commands: | |
| \use:N | 248, 381, 459, 463, 465, 510, 515, 517, 524, 527, 529, 1314, 1316, 1723, 1746, 1844, 1849, 2182, 2185, 2660, 3925, 5066, 5276, 5302, 6434, 7808, 7814 |
| \use:n | 1832, 2025, 2033, 2047, 2060, 3795, 4844, 5037, 5232, 5322, 5354, 6486, 6593 |
| \use:nn | 54, 197, 200, 206, 546, 1316, 2457, 2481, 2812, 2857, 3013, 3018, 3752, 3919, 4173, 4324, 4363, 4471, 4490, 4779, 4974, 5121, 5143, 5148, 5364, 5567, 5614, 5699, 5874, 6115, 6147, 6552, 6619, 6661, 6823, 7646, 8184, 8381, 8792 |
| \use_i:nn | 6593, 8815 |
| \use_ii:nn | 1836, 2934, 2949, 6633 |
| \use_none:nnnnnn | 3032 |
| \usebox | 8040 |
| \usemodule | 10, 13, 17, 18, 21, 32, 44, 83, 84, 86, 202, 422, 429, 3051 |
| \usepackage | 7, 21, 2025, 7883 |
| \useSGvar | 99, 8197 |
| \usestructure | 86, 6762 |
| \usetHEME | 7883 |
| \usetikzlibrary | 70, 2103, 8889 |
| V | |
| \varbind | 42, 46, 85, 90, 6916, 6939, 7050, 7058 |
| W | |
| \vardef | 30, 42, 46, 85, 114, 115, 4741, 7081, 7409 |
| \varempth | 93, 6014 |
| \Varname | 5902 |
| \varname | 5902 |
| \varannotation | 85, 4455 |
| \varref | 5902 |
| \varseq | 38, 85, 4943 |
| \vbox | 125, 7196, 7957, 8046, 8061, 8411, 8468, 8517, 8566, 8640 |
| vbox commands: | |
| \vbox_set:Nn | 2150 |
| \vdash | 7548 |
| \fill | 7812, 8699 |
| \text{M} | 48 |
| \text{Ms} | 50 |
| \text{n} | 38 |
| \text{op} | 40, 42 |
| \text{rule} | 7196, 8108, 8640 |
| \text{skip} | 7196, 8107, 8109, 8640 |
| \text{space} | 8701 |
| X | |
| \xspace | 1286, 1295, 3969, 3970, 3976, 3977 |
| Y | |
| \yield | 7264, 7455, 7458 |