

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

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

2023-09-17

Contents

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

*Version 3.3.0 (last revised 2023-09-17)

Definientia	47
Using Symbols Without Semantic Macros and Exporting Code in Modules	48
3.2.2 Assertions	49
3.2.3 Proofs	52
3.3 Mathematical Structures	52
3.3.1 Declaring and Using Structures	52
Instantiating Structures	54
3.3.2 Extending Structures and Axioms	55
Conservative Extensions	56
3.3.3 Nesting Structures and \this	57
3.4 Complex Inheritance and Theory Morphisms	59
3.4.1 Glueing Structures Together	61
3.4.2 Realizations	63
4 Extensions for Education	65
4.1 Slides and Course Notes	65
4.2 Problems and Exercises	65
4.2.1 Simple Problems	66
4.2.2 Multiple Choice Blocks	67
4.2.3 Filling-In Concrete Solutions	68
4.2.4 Including Problems	70
4.2.5 Testing and Spacing	70
4.3 Homework Assignments and Exams	70
4.3.1 Introduction	70
4.3.2 Package Options	70
4.3.3 Assignments	71
4.3.4 Including Assignments	71
4.3.5 Typesetting Exams	71
II User Manual	73
5 Basics	74
5.1 Package and Class Options	74
5.2 Math Archives and the MathHub Directory	75
5.2.1 The Structure of Math Archives	76
5.2.2 MANIFEST.MF-Files	76
5.3 The lib-Directory	77
5.4 Basic Macros	78
6 Document Features	79
6.1 Document Fragments	79
6.2 Using and Referencing Document Fragments	80
6.3 Cross-Document References	80

7 Modules and Symbols	83
7.1 Modules	83
7.1.1 Signature Modules, Languages, and Multilinguality	84
7.2 Symbol Declarations	84
7.2.1 Returns	85
7.3 Referencing Symbols	85
7.4 Notations and Semantic Macros	87
7.4.1 Precedences and Bracketing	88
7.4.2 Notations for Argument Sequences	89
7.4.3 Semantic Macros	89
7.5 Simple Inheritance	90
7.6 Variables and Sequences	92
7.7 Structures	93
7.7.1 Semantic Macros for Structures	93
8 Statements	96
8.1 More on Definitions	97
8.2 More on Assertions	98
9 Customizing Typesetting	99
9.1 Highlighting Symbol References	99
9.2 Styling Environments and Macros	100
9.3 Custom CSS for Environments	101
10 Additional Packages	102
10.1 NotesSlides Manual	102
10.1.1 Introduction	102
10.1.2 Package Options	102
10.1.3 Notes and Slides	103
10.1.4 Customizing Header and Footer Lines	104
10.1.5 Frame Images	104
10.1.6 Ending Documents Prematurely	105
10.1.7 Global Document Variables	105
10.1.8 Excursions	106
10.2 Problem Manual	107
10.2.1 Introduction	107
10.2.2 Problems and Solutions	107
10.2.3 Markup for Added-Value Services	108
Multiple Choice Blocks	109
Filling-In Concrete Solutions	110
10.2.4 Including Problems	111
10.2.5 Testing and Spacing	112
10.3 HWExam Manual	112
10.3.1 Introduction	112
10.3.2 Package Options	112
10.3.3 Assignments	113
10.3.4 Including Assignments	113
10.3.5 Typesetting Exams	113
10.4 Tikzinput Manual	114

III Documentation	116
11 S^TE_X Developer Manual	117
11.1 Documents	118
11.2 Modules	118
11.3 Symbols	120
11.4 Notations	122
11.5 Structural Features	125
11.6 Imports and Morphisms	127
11.7 Expressions and Semantic Macros	128
11.8 Optional (Key-Value) Argument Handling	129
11.9 Stylistable Commands and Environments	130
11.10 Math Archives	131
11.11 SMS-Mode	132
11.11.1 Second Pass	132
11.11.2 First Pass	133
11.12 Strings, File Paths, URIs	133
11.12.1 File Paths	134
File Path Constants and Variables	135
11.12.2 URIs	135
URI Constants and Variables	136
11.13 Language Handling	136
11.14 Inserting Annotations	137
11.14.1 Backend macros	137
11.15 Persisting Content from Math Archives in sms-Files	138
11.16 Utility Methods	138
11.16.1 Group-like Behaviours	139
12 Additional Packages	141
12.1 NotesSlides Documentation	141
12.2 Problem Documentation	141
12.3 HWExam Documentation	141
12.4 Tikzinput Documentation	141
IV Implementation	142
13 The S^TE_X Implementation	143
13.1 Setting up	143
13.2 Utilities	144
13.2.1 Calling kpsewhich and Environment Variables	144
13.2.2 Logging	145
13.2.3 Languages	146
13.2.4 Group-like Behaviours	148
13.2.5 HTML Annotations	149
13.2.6 Auxiliary Methods	151
13.2.7 Persistence	157
13.2.8 Files, Paths and URIs	158
13.2.9 File Hooks	166
13.3 Math Archives	167

13.4	Documents	171
13.4.1	Title	171
13.4.2	Sectioning	172
13.4.3	References	176
13.4.4	Inputs	184
13.5	SMS Mode	189
13.6	Modules	193
13.6.1	The smodule-environment	193
13.6.2	Structural Features	204
13.7	Inheritance	209
13.7.1	\importmodule/\usemodule	209
13.7.2	Theory Morphisms	215
13.8	Symbols	221
13.8.1	Declarations	221
13.8.2	Notations	230
a/B-mode argument handling		241
13.8.3	Variables	242
13.8.4	Sequences	247
13.8.5	Expressions	251
Invoking Semantic Macros		252
Argument Handling and Annotating		258
Term HTML Annotations		262
Automated Bracketing		265
Symname and Variants		266
Highlighting		268
13.9	Mathematical Structures	270
13.10	Statements	283
13.11	Proofs	289
13.12	Metatheory	297
13.13	MMT Interfaces	299
14	Additional Packages	303
14.1	Implementation: The notesslides Package	303
14.1.1	Class and Package Options	303
14.1.2	Notes and Slides	307
14.1.3	Environment and Macro Patches	310
14.1.4	Styling Across Notes/Slides	312
14.1.5	Beamer Compatibility	312
14.1.6	TODO Excursions	313
14.2	Implementation: The problem Package	314
14.2.1	Package Options	314
14.2.2	Problems and Solutions	315
14.3	Implementation: The hwexam Package	330
14.3.1	Package Options	330
14.3.2	Assignments	331
14.3.3	Leftovers	334
14.4	Tikzinput Implementation	334



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

Setting up [sTeX](#) with the dedicated [IDE](#) is easy:

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

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

4. Download the [MMT](#) `.jar`-file at the link provided in the setup and select it. The [IDE](#) should then be able to determine your [MMT](#) version.

And that's it. Click on *Finish* and your setup is finished. The extension will start and download **RuSTEX** and some fundamental **math archives** for you automatically (an internet connection is required when finishing the setup).

Part I

Tutorial

The dynamic [HTML](#) version of this part can be found at

[https://stexmmt.mathhub.info/:
sTeX/fullhtml?archive=sTeX/Documentation&filepath=tutorial.en.xhtml](https://stexmmt.mathhub.info/sTeX/fullhtml?archive=sTeX/Documentation&filepath=tutorial.en.xhtml)

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

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

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

Chapter 2

The Basics

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

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

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

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

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

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

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

A plain [LaTeX](#)-version of the above document fragment, without any [sTeX](#) markup, could look like this:

Example 1

Input:

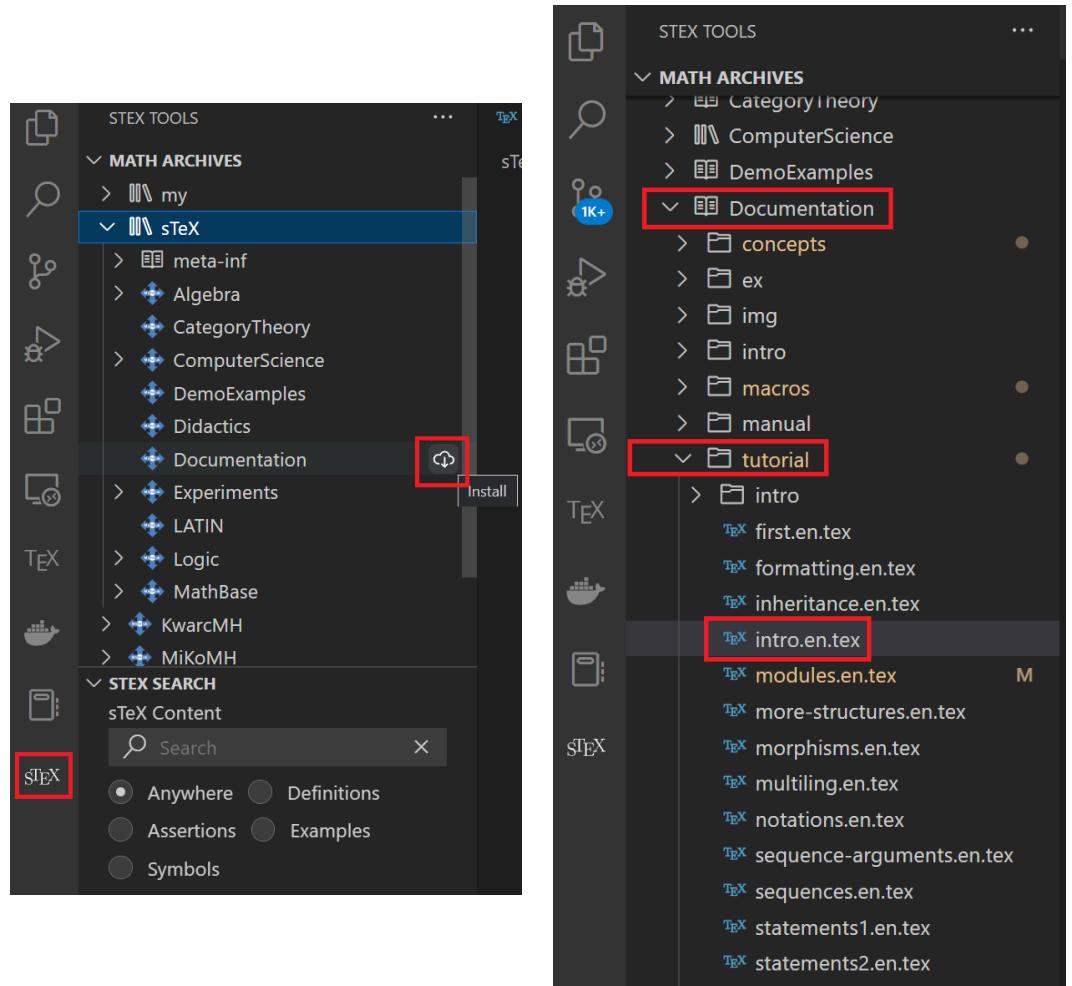


Figure 2: Installing Math Archives

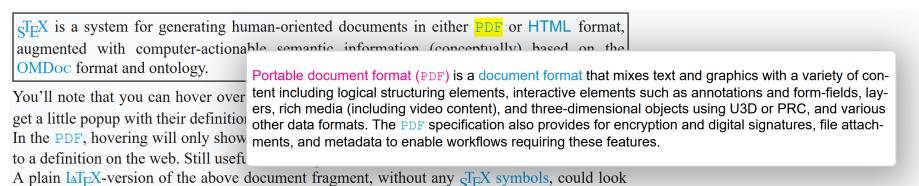


Figure 3: Definition on Hover

```

File [sTeX/Documentation]tutorial/intro/intro1plain.en.tex

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

```

Output:

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

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

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

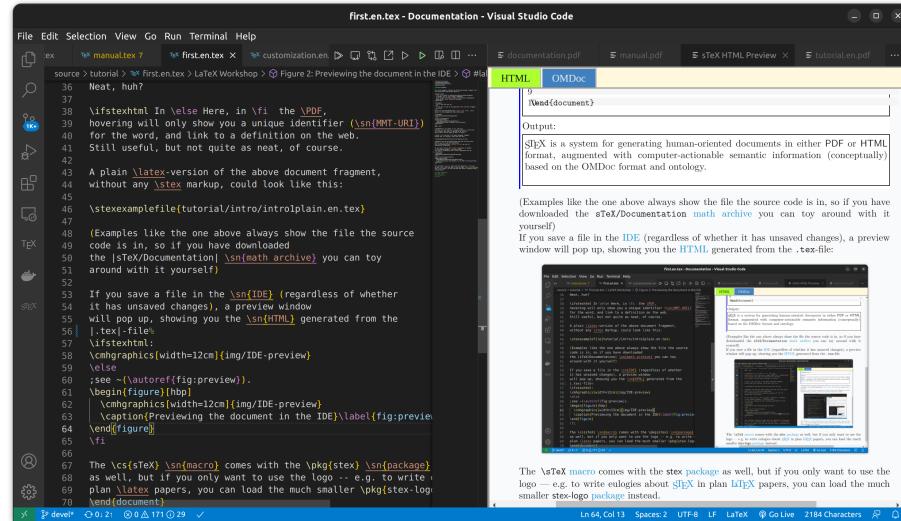


Figure 4: Previewing the document in the IDE

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

2.1 Text symbols

The most central concept behind **s_TeX** is that of a *symbol*:

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

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

2.1.1 Using Modules & Search in the IDE

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

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

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

A **text symbol** is a **symbol** `foo` with an associated **semantic macro** `\foo`. The **macro** `\foo` is allowed in text or math mode and produces a predefined piece of text output annotated with `foo`.

The variant `\fooname` produces the same output without annotation.

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

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

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

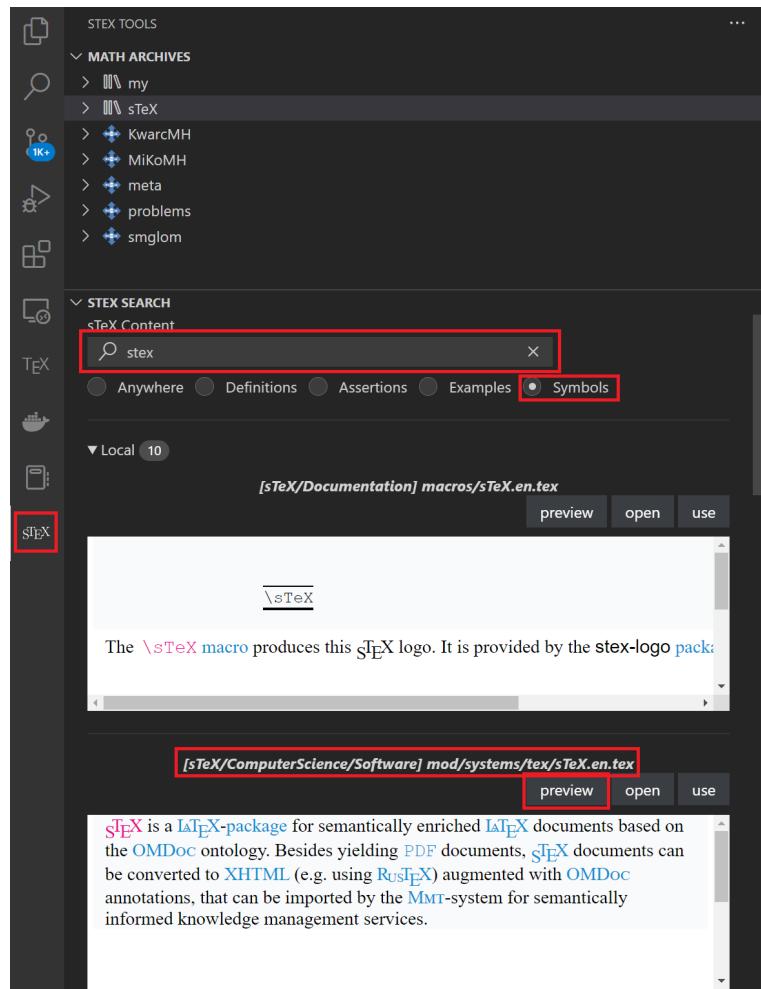


Figure 5: Search in the STeX IDE



Figure 6: OMDoc Preview

Example 2

Input:

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

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



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

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

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

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

2.6 Building and Exporting `HTML`

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



Figure 13: Missing Fragments in the `HTML` Preview

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



Figure 14: The Build PDF/XHTML/OMDoc Button

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

This will do the following:

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

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



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

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

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

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



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

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

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

Chapter 3

Mathematical Concepts

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

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

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

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

3.1 Simple Symbol Declarations

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

3.1.1 Semantic Macros and Notations

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

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

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

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

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

STEX

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:

STEX

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

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

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

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

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

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



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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Exercise

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

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

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

3.1.2 Types and Variables

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

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

\eqab  

invalid unit:  

http://mathhub.info/sTeX/MathTutorial/mod/Equality1/Equality?en?term 1?definition: Judgment |-- (implicit bind  

[a:/I/1, b:(/I/2 a)] (apply (apply equal a) b)) ::  

/omitted_type (Invalid Unit)  

View Problem (Alt+F8) No quick fixes available
```

Figure 17: Type Checking Warning

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

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

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

Figure 18: Type Checking Proof Tree

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

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



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



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

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

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



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

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



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

Types are particularly useful for **variables**:



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

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

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

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

Example 5

Input:

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

Output:

```
The natural numbers N are the numbers 0,1,2,....
Addition a+b refers to the process of adding two natural numbers.
```

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

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

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

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

Let's change that by adding two `variable` definitions¹:

Example 6

Input:

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

Output:

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

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

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

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

Example 7

Input:

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

Output:

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

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

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

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

Figure 19: Syntax Preview

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

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

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



So far (and when using the `use` button in the [IDE](#)), we have been using the `\usemodule` macro to import content. `\usemodule` is allowed anywhere and imports the referenced `module` content local to the current [TeX](#) group.

Now that we use imported `symbols` in `types` (and since we are *in* a `module`), we

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

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



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

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

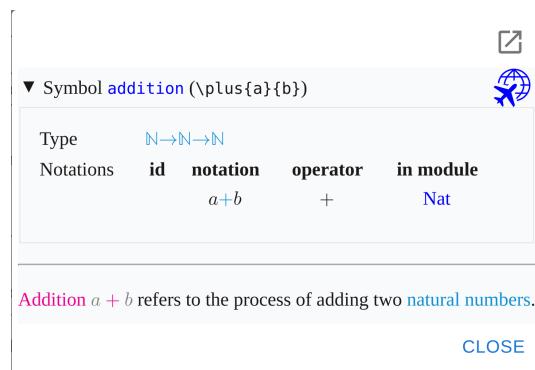


Figure 20: On-Click Popup in the `HTML`

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

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

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

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

Here’s what just happened:

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

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

▼ Document Elements

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

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

Figure 21: Inferred Type

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

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

3.1.3 Flexary Macros and Argument Modes

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

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

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

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

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

Example 8

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

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

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

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

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

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

Now we get new errors, that are easy to explain: Our **notation**

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

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

Now we can finally write `$\plus{a,b,c,d,e}$` and get $a + b + c + d + e$ – hooray!

...expect that our squiggly yellow **Invalid Unit** warnings are back. That's because the **type** of **addition** still corresponds to a binary operation, rather than a unary function on sequences.

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

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

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

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

Exercise

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

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

3.1.4 Precedences

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

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

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

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

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

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

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

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

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

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

3.1.5 Implicit Arguments

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

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

We will see the advantage of this approach over time. For now, consider that given objects a and b , the expression " $a = b$ " is either true or false³, and "*equal*" takes two argu-

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

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

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

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

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

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

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

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

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

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

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

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

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

Module Equality

Includes Function

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

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

Notations id notation operator in module

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

Document Elements

Includes `Nat`

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

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

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

Figure 22: Implicit Arguments

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

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

3.1.6 Finishing Equality

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

```

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

```

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

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

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

Using that, we can now write the following:

```

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

```

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

3.1.7 Variable Sequences

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

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

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

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

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

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

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

The screenshot shows a software interface for defining mathematical structures. At the top, there's a section titled "Module Equality". Below it, under "Includes Function", is a symbol "equal" (\eq{a_1, ..., a_n}) with a small globe icon next to it. Under "Document Elements", there's a section for "Nat, Conjunction". It shows a conjunction of equalities: $\{n: \mathbb{N}, m: \mathbb{N}, p: \mathbb{N}\} \frac{n=m \wedge m=p}{\mathbb{N}}$. Below this, in a box, is the "Inferred Type": $\{n: \mathbb{N}, m: \mathbb{N}, p: \mathbb{N}\} \text{Prop}$.

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 $\mathit{\fun{vop}}\!\!\{ \vA, \vA \}\vA$ is called
18 \definename{associative}, if
19 \definiens{$\forall a,b,c:A.\, \mathit{\eq{(\mathit{vop}{(\mathit{vA},\mathit{vb})},\mathit{vc})} = \mathit{vop}{(\mathit{vA},\mathit{vop}{(\mathit{vb},\mathit{vc})})}}
20 }$ \comp{for all} $\mathit{arg}[1]\{\mathit{inset}{\mathit{va},\mathit{vb},\mathit{vc}}\vA\$}.
21 }$ \comp{for all} $\mathit{arg}[1]\{\mathit{inset}{\mathit{va},\mathit{vb},\mathit{vc}}\vA\$}.
22 }$ \comp{for all} $\mathit{arg}[1]\{\mathit{inset}{\mathit{va},\mathit{vb},\mathit{vc}}\vA\$}.
23 \end{sdefinition}
24 %

```

Output:

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

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

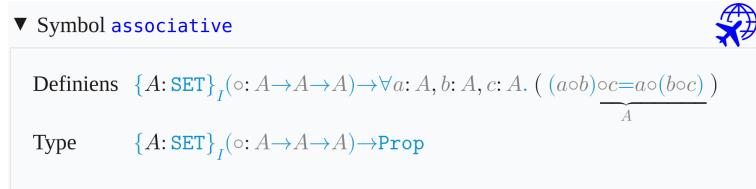


Figure 24: Type Inferred from Definiens

Using Symbols Without Semantic Macros and Exporting Code in Modules

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

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

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

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

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

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

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

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

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

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

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



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

Exercise

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

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

TODO⁵

3.2.2 Assertions

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

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

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

⁵TODO: intent?

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

which yields

Addition is associative

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

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

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

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

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

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

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

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

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

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

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

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

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

Figure 25: Assertion Statement in **OMDoc**

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

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

The **judgments as types** paradigm represents the validity of **proposition** via a designated *type of proofs*: For any **proposition** P , we introduce a collection $\vdash P$ of *proofs* of P .
sTeX
To say that the **proposition** *holds* is then equivalent to positing that *some* element $p : \vdash P$ exists – in which case *proofs* become typed objects in their own right.

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

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

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

1. $\varphi(0)$ and
 2. if $\varphi(m)$ holds for some m , then $\varphi(m + 1)$ also holds,
- then $\varphi(n)$ holds for all $n \in \mathbb{N}$.

Exercise

Annotate the above by:

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

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

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

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

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

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

```
> Assertion induction axiom ⊢ ∀φ:N→Prop.φ(0)⇒( ∀m:N.φ(m)⇒φ(m+1) )⇒( ∀n:N.φ(n) )
```

Figure 26: The Induction Axiom in OMDoc

3.2.3 Proofs



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

3.3 Mathematical Structures

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

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

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

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

3.3.1 Declaring and Using Structures

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

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

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

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

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

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

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

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

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

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

Example 11

Input:

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

Output:

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

Instantiating Structures

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

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

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

Syntax	Result
<code>\$\vM\$</code>	M
<code>\$\vM{}\$</code>	$\langle U_M, o_M \rangle$
<code>\$\vM{univ}\$</code>	U_M
<code>\$\vM{op}!\$</code>	o_M
<code>\$\vM{op}{a,b,c}\$</code>	$a o_M b o_M c$

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

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

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

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

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

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

This allows us to do things like:

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

yielding

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

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

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



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

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

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

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

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

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

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



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

3.3.2 Extending Structures and Axioms

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

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

Example 12

Input:

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

Output:

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

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

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



Figure 27: Axioms in OMDoc

Conservative Extensions

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

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

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

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

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

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

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

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

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

Consider the following conservative extension:

Example 13

Input:

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

Output:

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

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

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

...producing a^2 .

3.3.3 Nesting Structures and `\this`

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

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

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

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

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

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

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

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

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

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

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

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

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

Example 14

Input:

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

Output:

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

Now if we instantiate our `magma homomorphism`:

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

Here is a list of what we can do now:

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

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

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

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

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

3.4 Complex Inheritance and Theory Morphisms

We are starting to approach seriously experimental territory.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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

Exercise

Try to implement all of the above yourself!

3.4.1 Glueing Structures Together

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

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

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

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

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

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

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

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

```
\copymod{meetsl}{semilattice}{
    op @ meet
}
\copymod{joinsl}{semilattice}{
    univ = \univ,
    op @ join
}
```

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

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

```
\copymod{meetsl}{semilattice}{
    univ @ univ,
    op @ [meet]meet
}
\copymod{joinsl}{semilattice}{
    univ = \univ,
    op @ [join]join
}
```

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

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

3.4.2 Realizations

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

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

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

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

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

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

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

Proof: We need to prove the following

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

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

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

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

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

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

Example 15

Input:

```

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

8 \begin{extstructure*}{semilattice}
9   \realize{meets1}[
10     univ = \univ,
11     meet = \op!,
12     rel @ [order]order = \map{a,b}{\eq{\op{a,b},a}},
13     reflexive axiom = trivial,
14     transitive axiom = trivial,
15     antisymmetric axiom = trivial,
16     infimum axiom = trivial
17   }
18 \end{extstructure*}
19
20 \vardef{mysl}[return=\semilattice]{S}
21 $mysl{order}{a,b} \qquad \mysl{}[\univ,op,order]$
```

Output:

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

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

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

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

```

\begin{realization}{meets1}
\assign{univ}{\univ}
\assign{meet}{\op!}
\renamedecl{rel}[order]{order}
...
```

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

```

\usestructure{meets1}
\begin{sdefinition}[for=order]
\varbind{va,vb}
Let $ \semilattice! [\univ,op] $ a \sn{semilattice}.
We let $ \rel{\va,\vb} $.
iff $ \definiens{\eq{\op{\va,\vb},\va}} $.
\end{sdefinition}
```

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

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

Chapter 4

Extensions for Education

The last two chapters have shown generic markup and semantization facilities in **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 4.3](#)).

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

Output:

Exercise (Fitting Elefants)

How many Elefants can you fit into a Volkswagen beetle?

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

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

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

^aKorrekt

^bFalsch

that is for C and C++

^cFalsch

that is for Standard ML

^dNoooooooooo

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=Noooooooooo,feedback=that is for Java]{public static void}
9 \end{mcb}
10 \end{sproblem}
```

Output:

Exercise (Functions)

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

- def
- function
- fun
- public static void

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

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.

4.2.4 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.

4.2.5 Testing and Spacing

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

<code>\testspace</code>	<code>\testspace</code> takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: <code>\testsmallspace</code> , <code>\testmediumspace</code> , <code>\testbigspace</code> give small (1cm), medium (2cm), and big (3cm) vertical space.
<code>\testsmallspace</code>	<code>\testnewpage</code> makes a new page in <code>test</code> mode, and <code>\testemptypage</code> generates an empty page with the cautionary message that this page was intentionally left empty.
<code>\testemptypage</code>	<code>TODO</code> ⁸

4.3 Homework Assignments and Exams

4.3.1 Introduction

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

4.3.2 Package Options

solutions

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

notes**hints****gnotes****pts****min**

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

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

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

4.3.3 Assignments

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

4.3.4 Including Assignments

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

4.3.5 Typesetting Exams

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

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

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2023-09-17

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	Sum
total	0	0	0	10	0	0	0		0	20
reached										

good luck

9

⁹MK: The first three “problems” come from the stex examples above, how do we get rid of this?

Part II

User Manual

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

Chapter 5

Basics

5.1 Package and Class Options

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

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

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

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

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

This offers two advantages:

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

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

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

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

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

5.2 Math Archives and the MathHub Directory

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

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

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

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

- `sTeX/*` – a group of semi-experimental documents showcasing `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` `TODO11`.

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

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

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

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

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

7.2.1 Returns

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

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

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

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

7.3 Referencing Symbols

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

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

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 `\dobraackets{<code>}` wraps parentheses around `{<code>}`.

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

7.4.2 Notations for Argument Sequences

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

\argssep `\argssep{<parameter token>}{<separator>}`

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

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

\argmap `\argmap{<parameter token>}{<code>}{<separator>}`

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

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

\argarraymap `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`.

M → `Variables` induces the same `MMT/OMDoc` terms as `symbols` do, except for the head of the term being `<OMV name="..."/>` instead of `<OMS/>`.

`\varnotation` `\varnotation{\<variable>}[\<options>]{\<notation>}`

Takes the exact same arguments as `\notation`, but attaches an additional `notation` to the `variable` `\<variable>` rather than a `symbol`.

`\svar` `\svar[\<name>]{\<text>}`

Semantically marks up `\<text>` as representing a `variable` `\<name>`. The `variable` does not need to have been defined prior. If no `\<name>` is given, `\<text>` will be used as the name.

This is useful in situations like “throwaway expressions” or remarks; e.g.

`$\plus{\svar{n}, \svar{m}}$` means...

`\varseq` `\varseq{\<name>}[\<options>]{\<range>}{\<notation>}`

Declares a new `variable` sequence. The `\<options>` are the same as for `\vardef`. If not provided, `args=1` by default (a 0-ary sequence would just be a normal `variable`).

A `type` (given as `type=`) is interpreted to be the `type` of every element a_i of the sequence a_1, \dots, a_n (not of the sequence itself). If the `type` is itself a sequence A_1, \dots, A_n , the assumption is that its range is the same as the one of the new sequence, and the type of every a_i in the sequence is A_i .

`\<range>` needs to be a comma-separated sequence of either `args` many arguments, or `\ellipses`.

The resulting semantic macro is allowed anywhere `STEX` expects an argument mode `a` or `B` argument.

\ellipses Represents ellipses in a range; produces `\ellipses` in math mode.

\seqmap `\seqmap{\langle code \rangle}{\langle sequence \rangle}`

Maps the function $\langle code \rangle$ (containing #1) over every element of the $\langle sequence \rangle$.
Is allowed anywhere STeX expects an argument mode a or B argument.

7.7 Structures

Mathematical structure bundle interdependent symbols together.

`mathstructure (env.)`

`\begin{mathstructure}{\langle mname \rangle}[\langle name \rangle, this=\langle code \rangle]` opens a new mathematical structure with name $\langle mname \rangle$ (if provided) or $\langle name \rangle$ (otherwise), and semantic macro `\mname`. It subsequently behaves like the `smodule` environment.

\this

The optional `this=\langle code \rangle` option allows for setting the typesetting of the `\this` macro within a `mathstructure`. In particular, `\this` can be used in notations for symbols declared in the structure. `\this` can be thought of as representing “the” (current) instance of this structure.

`extstructure (env.)`

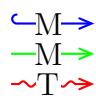
`\begin{extstructure}{\langle mname \rangle}[\langle name \rangle, this=\langle code \rangle]{\langle structs \rangle}` opens a new mathematical structure extending the structures given in $\langle structs \rangle$ (a comma-separated list of names).

`extstructure* (env.)`

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

\usestructure

The `\usestructure` macro behaves like `\usemodule` for mathematical structures, making the symbols available to use directly.

`mathstructure` make use of the *Theories-as-Types* paradigm (see [MueRabKoh:tat18]):
 `\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
\stexstylepsfesketch
\stexstyleMMTinclude
```

9.3 Custom CSS for Environments

Chapter 10

Additional Packages

10.1 NotesSlides Manual

10.1.1 Introduction

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

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

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

10.1.2 Package Options

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

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

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

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

10.1.3 Notes and Slides

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

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



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

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

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

10.2.3 Markup for Added-Value Services

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

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

Multiple Choice Blocks

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

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

- `T` for true answers, `F` for false ones,
- `Ttext` the verdict for true answers, `Ftext` for false ones, and
- `feedback` for a short feedback text given to the student.

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

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

^aKorrekt

^bFalsch

that is for C and C++

^cFalsch

that is for Standard ML

^dNoooooooooo

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=Noooooooooo,feedback=that is for Java]{public static void}
9   \end{mcb}
10 \end{sproblem}
```

Output:

Exercise (Functions)

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

- def
- function
- fun
- public static void

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

Filling-In Concrete Solutions

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

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

Example 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}
```

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

Example 27 How many Elefants can you fit into a Volkswagen beetle?

Input:

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

Output:

Exercise (Fitting Elefants)

How many Elefants can you fit into a Volkswagen beetle?

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

Obviously, the required argument of `\fillinsol` can be used for auto-grading. For concrete data like numbers, this is immediate, for more complex data like strings “soft comparisons” might be in order. ¹⁶

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

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 im MCQ. This still needs a bit of design. Also we want to make the formatting of the answer in solutions/test mode configurable.

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

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

10.2.5 Testing and Spacing

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

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

10.3 HWExam Manual

10.3.1 Introduction

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

10.3.2 Package Options

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

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

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the L^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 **type** referenced in the title of the assignment sheet), **type** (for the assignment type; e.g. “quiz”, **given** or “homework”), **given** (for the date the assignment was given), and **due** (for the date **due** the assignment is due).

10.3.4 Including Assignments

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

10.3.5 Typesetting Exams

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

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

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2023-09-17

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 $\backslash notation$ -command should contain the variant identifier and the precedences, as these are intrinsic parts of the notation.
- It should *not* contain the symbol itself however, so that notations can be copied between symbols.
- Notations as provided by users will largely adhere to the standard L^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

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

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

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

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

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

```
\_stex_invoke_variable:nnnnnnN
```

analogous to `_stex_invoke_symbol:nnnnnnN`, but for variables; sets the `\comp`-controlled highlighting to `\varemph@uri`.

```
\stex_invoke_symbol:
```

branches based on the mode and following characters:

- If math, check next character:
 - ! operator; defer to operator notation
 - else defer to notation and check the value of `\l_stex_current_return_tl ={\{return\}}`.
 - * If $\langle return \rangle$ is empty, call the notation macro,
 - * otherwise, call $\langle return \rangle \{\stex_invoke_symbol!\}$.
- If text:

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

TODO

11.8 Optional (Key-Value) Argument Handling

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

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

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

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

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

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

Example 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-⟨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_annotate:nnn
```

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

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

\stex_annotation_invisible:n
\stex_annotation_invisible:n

\stex_annotation_invisible:n {⟨code⟩}

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

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

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

\stex_mathml_intent:nn MathML Intent (TODO)
\stex_mathml_arg:nn

11.15 Persisting Content from Math Archives in sms-Files

\stex_persist:n
\stex_persist:e

\stex_persist:n {⟨code⟩}

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

TeXhackers note: ⟨code⟩ is being read with `expl3` category codes (except for spaces having catcode 10), but not pretokenized; i.e. ⟨code⟩ can safely change the current catcode scheme.

\c_stex_persist_mode_bool
\c_stex_persist_write_mode_bool

whether `usesms` or `writesms` are active.

11.16 Utility Methods

\stex_macro_body:N *

expands to the *expansion* of the provided macro, including parameter tokens, with the original category codes intact; e.g. if `\def\foo#1{First #1}`, then \stex_macro_body:N `\foo` expands to `First #1`.

\stex_macro_definition:N *

expands to the token list *defining* the provided macro, including parameters, command attributes (i.e. `\long`, `\protected`), with the original category codes intact; e.g. if `\protected\def\foo#1{First #1}`, then \stex_macro_definition:N `\foo` expands to `\protected\def\foo#1{First #1}`.

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

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

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

Makes `\macro` throw an error message `error/deactivated-macro{(msg)}`, notifying an author that the macro is only allowed in certain environments.

`\stex_reactivate_macro:N` restores the functionality of the macro.

`\stex_kpsewhich:Nn`

`\stex_kpsewhich:Nn {(\macro)} {(args)}`

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

TeXhackers note: Does not require `shell-escape`

`\stex_get_env:Nn`

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

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

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

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

TeXhackers note: Calls `\input{non-existent file}`.

`\stex_ignore_spaces_and_pars:`

As the name suggests, ignores all subsequent spaces and `\pars` until the first non-expandable macro is encountered.

Useful for e.g. ending `\symdecl` and related macros with, so that formatting sources with empty lines does not cause paragraph breaks.

11.16.1 Group-like Behaviours

`\stex_pseudogroup_with:nn`

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

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

TeXhackers note: Does *not* work recursively!

`\stex_pseudogroup:nn`

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

Expands `(code2)`, and inserts the result after `(code1)`. Works recursively and allows for restoring the values of macros in combination with `\stex_pseudogroup_restore:N`, but *only for macros that take no arguments*:

`\stex_pseudogroup_restore:N *` `\stex_pseudogroup_restore:N {(\macro)}`

Example 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   nofrontmatter .bool_set:N = \c_stex_no_frontmatter_bool,
36   unknown     .code:n     = {}
37 }
38 \exp_args:NNo \clist_set:Nn \c_stex_debug_clist \c_stex_debug_clist
39 \ProcessKeysOptions { stex / package }

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

```

13.2 Utilities

41 \cs_set_eq:NN \stex_undefined: \undefined

13.2.1 Calling kpsewhich and Environment Variables

42 <@@=stex_envs>

\stex_kpsewhich:Nn

```

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

```

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

\stex_get_env:Nn

```

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

```

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

13.2.2 Logging

```

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

```

(End of definition for `\stex_debug:nn`. This function is documented on page 117.)

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

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

```

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

```

(End of definition for `\stex_fatal_error:n` and `\stex_fatal_error:nnn`. These functions are documented on page 139.)

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

```

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

```

```

107     }
108 }
```

13.2.3 Languages

109 `<@=stex_lang>`

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

```

110 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
111   en = english ,
112   de = ngerman ,
113   ar = arabic ,
114   bg = bulgarian ,
115   ru = russian ,
116   fi = finnish ,
117   ro = romanian ,
118   tr = turkish ,
119   fr = french
120 }
121 \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 }
133 % todo: chinese simplified (zhs)
134 %         chinese traditional (zht)
```

(End of definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 136.)

`\l_stex_current_language_str`

135 `\str_new:N \l_stex_current_language_str`

(End of definition for `\l_stex_current_language_str`. This variable is documented on page 136.)

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

`\stex_set_language:n`

`\stex_set_language:x`

`\stex_set_language:o`

```

136 \cs_new_protected:Nn \stex_set_language:n {
137   \str_set:Nn \l_stex_current_language_str { #1 }
138   \prop_if_in:NnTF \c_stex_languages_prop {#1} {
139     \tl_set_rescan:Nnx \l__stex_lang_lang_str {}{\prop_item:Nn \c_stex_languages_prop {#1}}
140     \cs_if_eq:NNTF @onlypreamble @notprerr {
141       \ltx@ifpackageloaded{babel} {
142         \exp_args:N\selectlanguage\l__stex_lang_lang_str
143       } {}
144     } {
145       \ltx@ifpackageloaded{babel} {} {
146         \str_if_eq:nnTF {#1} {tr} {
```

```

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

```

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

`\stex_language_from_file:`

```

158 \cs_new_protected:Nn \stex_language_from_file: {
159     \seq_get_right:NN \g_stex_current_file \l_tmpa_str
160     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
161     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % = ".tex/.dtx/.sty"
162     \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
163         \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
164             \exp_args:No \str_if_eq:nnF \l_tmpa_str {ltx} {
165                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
166             }
167         }
168     }
169     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
170     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
171         \seq_pop_right:NN \l_tmpa_seq \l__stex_lang_str
172         \str_if_eq:NNF \l__stex_lang_str \l_stex_current_language_str {
173             \exp_args:NNo \prop_if_in:NnT \c_stex_languages_prop \l__stex_lang_str {
174                 \stex_set_language:o \l__stex_lang_str
175             }
176         }
177         \stex_debug:nn{lang} {Language~\l_stex_current_language_str~
178             inferred~from~file~name}
179     }
180 }

```

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

Loading babel:

```

181 \clist_if_empty:NF \c_stex_languages_clist {
182     \bool_set_false:N \l__stex_lang_turkish_bool
183     \seq_clear:N \l_tmpa_seq
184     \clist_map_inline:Nn \c_stex_languages_clist {
185         \str_set:Nx \l_tmpa_str {#1}
186         \str_if_eq:nnT {#1}{tr} {
187             \bool_set_true:N \l__stex_lang_turkish_bool
188         }
189         \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
190             \tl_set_rescan:Nno \l_tmpa_str {} \l_tmpa_str
191             \seq_put_right:No \l_tmpa_seq \l_tmpa_str
192         } {
193             \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}

```

```

194     }
195   }
196 \stex_debug:nn{lang} {Languages:~\seq_use:Nn \l_tmpa_seq {,~} }
197 \bool_if:NTF \l_stex_lang_turkish_bool {
198   \exp_args:NNe \use:nn \RequirePackage
199   {[main=\seq_use:Nn \l_tmpa_seq, ,shorthands=:!]}{babel}
200 }{
201   \exp_args:NNe \use:nn \RequirePackage
202   {[main=\seq_use:Nn \l_tmpa_seq, ]}{babel}
203 }
204 }
```

13.2.4 Group-like Behaviours

205 ⟨@=stex_groups⟩

\stex_pseudogroup:nn

\stex_pseudogroup_restore:N

```

206 \cs_new_protected:Npn \stex_pseudogroup:nn {
207   \exp_args:Nne \use:nn
208 }
209 \cs_new:Nn \stex_pseudogroup_restore:N {
210   \tl_if_exist:NTF #1 {
211     \tl_set:Nn \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
212   }{
213     \cs_undefine:N \exp_not:N #1
214   }
215 }
```

(End of definition for \stex_pseudogroup:nn and \stex_pseudogroup_restore:N. These functions are documented on page 139.)

\stex_pseudogroup_with:nn

```

216 \cs_new_protected:Nn \stex_pseudogroup_with:nn {
217   \tl_map_inline:nn{#1} {
218     \cs_set_eq:cN{\__stex_groups_\tl_to_str:n{##1}}{#1}
219   }
220   #2
221   \tl_map_inline:nn{#1} {
222     \cs_set_eq:Nc{#1}{\__stex_groups_\tl_to_str:n{##1}}
223     \cs_undefine:c{\__stex_groups_\tl_to_str:n{##1}}
224   }
225 }
```

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

\stex_metagroup_new:n List of all currently existing metagroup identifiers

\stex_metagroup_new:o

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

```

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

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

```

\stex_metagroup_do_in:nn
\stex_metagroup_do_in:nx
232 \prg_new_conditional:Nnn \__stex_groups_exists:n {TF} {
233     \str_if_exist:cTF{l__stex_groups_#1_int}
234         \prg_return_true: \prg_return_false:
235 }
236
237 \cs_new_protected:Nn \stex_metagroup_do_in:nn {
238     \__stex_groups_exists:nTF{#1}{
239         \__stex_groups_do_in:nn{#1}{#2}
240     }{
241         \msg_error:nnn{stex}{error/metagroup/missing}{#1}
242     }
243 }
244 \cs_generate_variant:Nn \stex_metagroup_do_in:nn {nx}
245
246 \cs_new_protected:Nn \__stex_groups_do_in:nn {
247     \exp_args:Nnx\stex_debug:nn{metagroup}{adding~to~\detokenize{#1}:^~J\tl_to_str:n{#2}}
248     \tl_set:Nn\__stex_groups_tmp{#2}
249     \exp_args:Nx \int_compare:nNnF {\use:c{l__stex_groups_#1_int}}
250         = \currentgrouplevel {
251             \tl_if_exist:cTF{g__stex_groups_#1_\the\currentgrouplevel _content} {
252                 \exp_args:Nno \tl_gput_right:cng{g__stex_groups_#1_\the\currentgrouplevel _content}
253             }{
254                 \exp_args:Nno \tl_gset:cn{g__stex_groups_#1_\the\currentgrouplevel _content}
255             }\__stex_groups_tmp
256             \bool_if_exist:cF {l__stex_groups_\the\currentgrouplevel _bool} {
257                 \group_insert_after:N \__stex_groups_do:
258                 \bool_set_true:c {l__stex_groups_\the\currentgrouplevel _bool}
259             }
260         }
261         \__stex_groups_tmp
262     }
263
264 \cs_new_protected:Nn \__stex_groups_do: {
265     \seq_map_inline:Nn \l__stex_groups_ids_seq {
266         \tl_if_exist:cT{g__stex_groups_#1_\int_eval:n{\currentgrouplevel+1}_content} {
267             \exp_args:NNno \exp_args:Nno \__stex_groups_do_in:nn{##1} {
268                 \csname g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content\endcsname
269             }
270             \cs_undefine:c{g__stex_groups_##1_\int_eval:n{\currentgrouplevel+1}_content}
271         }
272         \bool_if_exist:cF {l__stex_groups_\int_eval:n\currentgrouplevel _bool} {
273             \group_insert_after:N \__stex_groups_do:
274             \bool_set_true:c {l__stex_groups_\int_eval:n\currentgrouplevel _bool}
275         }
276     }
277 }

```

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

13.2.5 HTML Annotations

278 `<@=stex_annotation>`

```

\stex_if_do_html:TF Whether to (locally) produce HTML output
279 \bool_new:N \_stex_html_do_output_bool
280 \bool_set_true:N \_stex_html_do_output_bool
281
282 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
283     \bool_if:nTF \_stex_html_do_output_bool
284     \prg_return_true: \prg_return_false:
285 }

```

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

`\stex_suppress_html:n` Temporarily disable HTML output

```

286 \cs_new_protected:Nn \stex_suppress_html:n {
287     \stex_pseudogroup:nn{
288         \bool_set_false:N \_stex_html_do_output_bool
289         #1
290     }{
291         \stex_if_do_html:T {
292             \bool_set_true:N \_stex_html_do_output_bool
293         }
294     }
295 }

```

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

We determine the backend:

```

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

```

```

323   \bool_set_true:N \_stex_html_do_output_bool
324 }{
325   \stexhtmlfalse
326   \bool_set_false:N \_stex_html_do_output_bool
327 }

```

(End of definition for `\stex_if_html_backend:TF`, `\ifstexhtml`, and `\stex@backend`. These functions are documented on page 137.)

`_stex_annotation_force_break:n`

```

328 \stex_if_html_backend:TF {
329   \cs_new_protected:Nn \_stex_annotation_force_break:n {
330     \stex_annotation_invisible:n{~}
331     #1
332     \stex_annotation_invisible:n{~}
333   }
334 }{
335   \cs_new_protected:Nn \_stex_annotation_force_break:n { #1 }
336 }

```

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

`\mmlintent`

```

\mmlarg
337 \stex_if_html_backend:TF {
338   \cs_new_protected:Npn \mmlintent #1 #2 {
339     \stex_annotation_nn{mml:intent={#1}}{#2}
340   }
341   \cs_new_protected:Npn \mmlarg #1 #2 {
342     \stex_annotation_nn{mml:arg={#1}}{#2}
343   }
344 }{
345   \cs_new_protected:Npn \mmlintent #1 #2 { #2 }
346   \cs_new_protected:Npn \mmlarg #1 #2 { #2 }
347 }

```

(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 <@@=stex_aux>
349 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
350   \tl_set_eq:cN{\tl_to_str:n{#1}~~~orig}{#1}
351   \tl_set:Nn#1{
352     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
353   }
354 }
355 \cs_new_protected:Nn \stex_reactivate_macro:N {
356   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1}~~~orig\endcsname
357 }

```

(End of definition for `\stex_deactivate_macro:Nn` and `\stex_reactivate_macro:N`. These functions are documented on page 139.)

```

\stex_ignore_spaces_and_pars:
358 \protected\def\stex_ignore_spaces_and_pars:{%
359   \begingroup\catcode13=10\relax
360   \@ifnextchar\par{%
361     \endgroup\expandafter\stex_ignore_spaces_and_pars:\@gobble
362   }{%
363     \endgroup
364   }
365 }

```

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

`\stex_keys_define:nnnn`

```

366 \cs_new_nopar:Nn \stex_keys_define:nnnn {
367   \tl_gset:cn {__stex_aux_keys_#1_pre_tl}{#2}
368   \tl_gset:cn {__stex_aux_keys_#1_def_tl}{#3}
369   \tl_if_empty:nF{#4}{%
370     \clist_map_inline:nn{#4}{%
371       \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_pre_tl}
372       \tl_gput_left:co{__stex_aux_keys_#1_pre_tl} \l__stex_aux_tl
373       \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_##1_def_tl}
374       \tl_gput_left:cn{__stex_aux_keys_#1_def_tl} ,
375       \tl_gput_left:co{__stex_aux_keys_#1_def_tl} \l__stex_aux_tl
376     }
377   }
378   \tl_set_eq:Nc \l__stex_aux_tl {__stex_aux_keys_#1_def_tl}
379   \exp_args:Nno \keys_define:nn {stex / #1} {\l__stex_aux_tl}
380 }

```

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

`\stex_keys_set:nn`

```

381 \cs_new_nopar:Nn \stex_keys_set:nn {
382   \use:c{\__stex_aux_keys_#1_pre_tl}
383   \keys_set:nn {stex / #1} { #2 }
384 }

```

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

Some ubiquitous key sets:

```

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

```

```

400     \tl_clear:N \l_stex_key_title_tl
401 }{
402     title .tl_set:N = \l_stex_key_title_tl
403 }{}
404
405 \stex_keys_define:nnnn{style}{%
406     \clist_clear:N \l_stex_key_style_clist
407 }{
408     style .clist_set:N = \l_stex_key_style_clist
409 }{}
410
411 \stex_keys_define:nnnn{deprecate}{%
412     \str_clear:N \l_stex_key_deprecate_str
413 }{
414     deprecate .str_set_x:N = \l_stex_key_deprecate_str
415 }{}
416
417 \stex_keys_define:nnnn{uses}{%
418     uses .code:n = {
419         \clist_map_inline:nn{#1}{%
420             \stex_if_starts_with:nTF{##1}[%
421                 \_stex_aux_split_at_bracket:w ##1 \_stex_end:
422             }{
423                 \usemodule{##1}
424             }
425         }
426     }%
427 }{}%
428
429 \cs_new_protected:Npn \_stex_aux_split_at_bracket:w [ #1 ] #2 \_stex_end: {%
430     \usemodule[#1]{#2}
431 }

```

_stex_do_deprecation:n

```

432 \cs_new:Nn \_stex_do_deprecation:n {
433     \str_if_empty:NF \l_stex_key_deprecate_str {
434         \msg_warning:nnxx{stex}{warning/deprecated}{#1}{\l_stex_key_deprecate_str}
435     }
436 }

```

(End of definition for `_stex_do_deprecation:n`. This function is documented on page 117.)

_stex_do_id:

```

437 \cs_new_protected:Nn \_stex_do_id: {
438     \stex_if_smsmode:F {
439         \str_if_empty:NF \l_stex_key_id_str {
440             \exp_args:No \stex_ref_new_doc_target:n \l_stex_key_id_str
441         }
442     }
443 }

```

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

```

\stex_new_stytable_env:nmmmmnnn
\stex_new_stytable_cmd:nnnn
\stex_style_apply:
444 \cs_new_protected:Nn \stex_new_stytable_cmd:nnnn {
445   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[2] [] {
446     \__stex_aux_patch:nnn{#1}{##1}{##2}
447   }
448   \exp_after:wN \NewDocumentCommand\cs:w #1\cs_end:{#2} {
449     \cs_set:Npn \stex_style_apply: {
450       \__stex_aux_apply_patch:n{#1}
451     }
452     #3
453   }
454   \tl_set:cn {__stex_aux_style_#1:} { #4 }
455 }
456
457 \cs_new_protected:Nn \__stex_aux_apply_patch:n {
458   \clist_if_empty:NTF \l_stex_key_style_clist {
459     \tl_clear:N \thisstyle
460     \use:c{__stex_aux_style_#1:}
461   }{
462     \clist_get:NN \l_stex_key_style_clist \thisstyle
463     \tl_if_exist:cTF{__stex_aux_style_#1_\thisstyle :} {
464       \use:c{__stex_aux_style_#1_\thisstyle :}
465     }{
466       \use:c{__stex_aux_style_#1:}
467     }
468   }
469 }
470
471 \cs_new_protected:Nn \__stex_aux_patch:nnn {
472   \str_if_empty:nTF {#2} {
473     \tl_set:cn{__stex_aux_style_#1:}{#3}
474   }{
475     \tl_set:cn{__stex_aux_style_#1_#2:}{#3}
476   }
477 }
478
479 \cs_new_protected:Nn \stex_new_stytable_env:nnnnnnnn {
480   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[3] [] {
481     \__stex_aux_patch:nnnn{#1}{##1}{##2}{##3}
482   }
483   \NewDocumentEnvironment{#7#1}{#2} {
484     \cs_set:Npn \stex_style_apply: {
485       \__stex_aux_apply_patch_begin:n{#1}
486     }
487     #3
488   }{
489     \cs_set:Npn \stex_style_apply: {
490       \__stex_aux_apply_patch_end:n{#1}
491     }
492     #4
493   }
494   \tl_set:cn {__stex_aux_style_#1_start:} { #5 }
495   \tl_set:cn {__stex_aux_style_#1_end:} { #6 }
496 }

```

```

497 \cs_new_protected:Nn \__stex_aux_patch:n {
498   \str_if_empty:nTF {#2} {
499     \tl_set:cn{\__stex_aux_style_#1_start:}{#3}
500     \tl_set:cn{\__stex_aux_style_#1_end:}{#4}
501   }{
502     \tl_set:cn{\__stex_aux_style_#1_#2_start:}{#3}
503     \tl_set:cn{\__stex_aux_style_#1_#2_end:}{#4}
504   }
505 }
506 }
507
508 \cs_new_protected:Nn \__stex_aux_apply_patch_begin:n {
509   \clist_if_empty:NTF \l_stex_key_style_clist {
510     \tl_clear:N \thisstyle
511     \use:c{\__stex_aux_style_#1_start:}
512   }{
513     \clist_get:NN \l_stex_key_style_clist \thisstyle
514     \stex_debug:nn{styling}{dominant-style:~\thisstyle}
515     \tl_if_exist:cTF{\__stex_aux_style_#1_\thisstyle_start:}{
516       \use:c{\__stex_aux_style_#1_\thisstyle_start:}
517     }{
518       \use:c{\__stex_aux_style_#1_start:}
519     }
520   }
521 }
522
523 \cs_new_protected:Nn \__stex_aux_apply_patch_end:n {
524   \tl_if_empty:NTF \thisstyle {
525     \use:c{\__stex_aux_style_#1_end:}
526   }{
527     \tl_if_exist:cTF{\__stex_aux_style_#1_\thisstyle_end:}{
528       \use:c{\__stex_aux_style_#1_\thisstyle_end:}
529     }{
530       \use:c{\__stex_aux_style_#1_end:}
531     }
532   }
533 }

```

(End of definition for `\stex_new_stytable_env:nnnnnnn`, `\stex_new_stytable_cmd:nnnn`, and `\stex_style_apply:..`. These functions are documented on page 130.)

`\stex_str_if_ends_with_p:nn`

`\stex_str_if_ends_with:nnTF`

```

534 \prg_new_conditional:Nnn \stex_str_if_ends_with:nn {p,T,F,TF} {
535   \exp_args:Ne \str_if_eq:nnTF {
536     \str_range:nnn{#1}{-}\str_count:n{#2}{-1}
537   }{#2}\prg_return_true: \prg_return_false:
538 }

```

(End of definition for `\stex_str_if_ends_with:nnTF`. This function is documented on page 133.)

`\stex_str_if_starts_with_p:nn`

`\stex_str_if_starts_with:nnTF`

```

539 \prg_new_conditional:Nnn \stex_str_if_starts_with:nn {p,T,F,TF} {
540   \exp_args:Ne \str_if_eq:nnTF {
541     \str_range:nnn{#1}{1}{\str_count:n{#2}}
542   }{#2}\prg_return_true: \prg_return_false:

```

543 }

(End of definition for `\stex_str_if_starts_with:nTF`. This function is documented on page 133.)

`\stex_macro_body:N`

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

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

`\stex_macro_definition:N`

```
559 \cs_new:Nn \stex_macro_definition:N {
560     \__stex_aux_prefix:e {\cs_prefix_spec:N #1}
561     \def\exp_not:N #1
562     \__stex_aux_params:e {\cs_parameter_spec:N #1}
563     {
564         \stex_macro_body:N #1
565     }
566 }
567
568 \cs_new:Nn \__stex_aux_prefix:n {
569     \tl_if_empty:nF{#1} {
570         \str_if_eq:eeTF {
571             \tl_range:nnn{#1}{1}{10}~
572         }{\tl_to_str:n{\protected}}{
573             \protected
574             \__stex_aux_prefix_long:e {
575                 \str_range:nnn{#1}{11}{-1}
576             }
577         }{
578             \__stex_aux_prefix_long:n {#1}
579         }
580     }
581 }
582 \cs_generate_variant:Nn \__stex_aux_prefix:n {e}
583
584 \cs_new:Nn \__stex_aux_prefix_long:n {
585     \tl_if_empty:nF{#1} {
586         \str_if_eq:eeT {
587             \tl_range:nnn{#1}{1}{10}~
588         }{\tl_to_str:n{\long}}{\long}
589     }
590 }
```

```

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

```

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

13.2.7 Persistence

```
604 <@=stex_persist>
```

We check the environment variables:

```

605 \stex_get_env:Nn\__stex_persist_env_str{STEX_USESMS}
606 \str_if_empty:NF\__stex_persist_env_str{
607     \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false} {
608         \bool_set_true:N \c_stex_persist_mode_bool
609     }
610 }
611 \stex_get_env:Nn\__stex_persist_env_str{STEX_WRITESMS}
612 \str_if_empty:NF\__stex_persist_env_str{
613     \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false} {
614         \bool_set_true:N \c_stex_persist_write_mode_bool
615     }
616 }

```

```

\stex_persist:n
\stex_persist:e
617 \iow_new:N \c__stex_persist_sms_iow
618
619 \bool_if:NTF \c_stex_persist_write_mode_bool {
620     \stex_if_html_backend:TF{
621         \cs_new:Npn \stex_persist:n #1 {}
622         \cs_new:Npn \stex_persist:e #1 {}
623     }{
624         \cs_new_protected:Nn \stex_persist:n {
625             \iow_now:Nn \c__stex_persist_sms_iow {#1}
626         }
627         \cs_generate_variant:Nn \stex_persist:n {e}
628     }
629 }{
630     \cs_new:Npn \stex_persist:n #1 {}
631     \cs_new:Npn \stex_persist:e #1 {}
632 }

```

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

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

```

633 \cs_new_protected:Nn \__stex_persist_load_file:n {
634   \file_if_exist:nT{#1} {
635     \group_begin:
636     \cs_set:Npn \stex_persist:n ##1 {}
637     \cs_set:Npn \stex_persist:e ##1 {}
638     \stex_debug:n{persist}{restoring~from~sms~file}
639     \catcode`\ =10\relax
640     \cs:w @ @ input \cs_end:#1\relax
641   \group_end:
642 }
643 }
644 }
645
646 \cs_new_protected:Nn \__stex_persist_write_only: {
647   \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
648   \AtEndDocument{ \iow_close:N \c__stex_persist_sms_iow }
649 }
650
651 \cs_new_protected:Nn \__stex_persist_read_and_write: {
652   \file_if_exist:nTF{\jobname.sms} {
653     \ior_open:Nn \g_tmpa_ior {\jobname.sms}
654     \iow_open:Nn \g_tmpa_iow {\jobname sms2}
655     \ior_str_map_inline:Nn \g_tmpa_ior {
656       \iow_now:Nn \g_tmpa_iow {##1}
657     }
658     \iow_close:N \g_tmpa_iow
659     \ior_close:N \g_tmpa_ior
660     \__stex_persist_write_only:
661     \ior_open:Nn \g_tmpa_ior {\jobname sms2}
662     \ior_str_map_inline:Nn \g_tmpa_ior {
663       \iow_now:Nn \c__stex_persist_sms_iow {##1}
664     }
665     \ior_close:N \g_tmpa_ior
666     \__stex_persist_load_file:n{\jobname sms2}
667   } \__stex_persist_write_only:
668 }
669
670 \cs_new_protected:Nn \_stex_persist_read_now: {
671   \bool_if:NTF \c_stex_persist_mode_bool {
672     \bool_if:NTF \c_stex_persist_write_mode_bool {
673       \__stex_persist_read_and_write:
674       {
675         \__stex_persist_load_file:n{\jobname sms}
676       }
677     }{
678       \bool_if:NT \c_stex_persist_write_mode_bool \__stex_persist_write_only:
679     }
680 }

```

13.2.8 Files, Paths and URIs

681 <@=stex_path>

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

```

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

```

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

```

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

```

```

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

(End of definition for `\stex_file_resolve:Nn`. This function is documented on page [134](#).)

```

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

(End of definition for `\stex_if_file_absolute:NTF`. This function is documented on page [134](#).)

```

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

(End of definition for `\stex_file_use:N`. This function is documented on page [134](#).)

```

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

```

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

```

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

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

```

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

```

(End of definition for `\stex_file_split_off_ext:NN` and `\stex_file_split_off_lang:NN`. These functions are documented on page 134.)

URIs:

`\stex_map_uri:Nnnnn`

```

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

```

```

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

```

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

```

\stex_uri_set:Nn
\stex_uri_set:Nn
\stex_uri_set:Nx
827 \str_set:Nx\__stex_path_colonslash{\cColonStr}
828 \cs_new_protected:Nn \__stex_path_uri_set:NnN {
829   \str_if_empty:nTF {#2} {
830     \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{empty}
831   }{
832     \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn #1 \__stex_path_colonslash { \tl_to_str
833     \seq_pop_left:NN #1 \l__stex_path_auth_str
834     \seq_if_empty:NTF #1 {
835       \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{missing-authority}
836     }{
837       \exp_args:NNnx \seq_set_split:Nnn #1 ? {\exp_args:NNo \seq_use:Nn #1 \__stex_path_colon
838       \seq_pop_left:NN #1 \l__stex_path_path
839       #3 \l__stex_path_path \l__stex_path_path
840       \seq_if_empty:NTF #1 {
841         \exp_args:NNo \__stex_path_uri_set:Nnxnn #1 \l__stex_path_auth_str
842         {\stex_file_use:N \l__stex_path_path} {} {}
843       }{
844         \seq_pop_left:NN #1 \l__stex_path_mod
845         \seq_if_empty:NTF #1 {
846           \exp_args:NNo \__stex_path_uri_set:Nnxon #1 \l__stex_path_auth_str
847           {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod {}
848         }{
849           \seq_pop_left:NN #1 \l__stex_path_name
850           \seq_if_empty:NTF #1 {
851             \exp_args:NNo \__stex_path_uri_set:Nnxon #2 \l__stex_path_auth_str
852             {\stex_file_use:N \l__stex_path_path} \l__stex_path_mod \l__stex_path_name
853           }{
854             \msg_error:nnxx{stex}{error/invalid-uri}{\tl_to_str:n{#2}}{too-many~?s}
855           }
856         }
857       }
858     }
859   }
860   \stex_debug:nn{uris}{Set~\tl_to_str:n{#1}~to~\stex_uri_use:N #1}
861 }
862
863 \cs_new_protected:Nn \__stex_path_uri_set:Nnnnn{
864   \tl_set:Nn #1 {
865     \__stex_path_auth:n{ #2 }
866     \__stex_path_path:n{ #3 }
867     \__stex_path_module:n{ #4 }
868     \__stex_path_name:n{ #5 }
869   }

```

```

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

```

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

`\stex_uri_resolve:Nn`

```

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

```

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

`\stex_uri_use:N`

```

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

```

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

`\stex_uri_from_repo_file>NNNn`

`\stex_uri_from_repo_file_nolang:NNNn`

```

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

```

```

913     }
914 }
915
916 \cs_new_protected:Nn \__stex_path_relativize:N {
917     \seq_set_eq:NN \l__stex_path_seq \l__stex_path_file
918     \seq_map_inline:Nn \c_stex_mathhub_file { % mathhub path
919         \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
920     }
921     \stex_file_set:Nx \l__stex_path_path {\prop_item:Nn \l_stex_current_archive_prop {id} }
922     \seq_map_inline:Nn \l__stex_path_path { % id
923         \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl
924     }
925     \seq_pop_left:NN \l__stex_path_seq \l__stex_path_tl % source
926
927     \stex_uri_set:Nx #1 { \l__stex_path_uri / \stex_file_use:N \l__stex_path_seq }
928 }

```

(End of definition for `\stex_uri_from_repo_file:NNNn` and `\stex_uri_from_repo_file_nolang:NNNn`.
These functions are documented on page 135.)

```

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

```

(End of definition for `\stex_uri_from_current_file:Nn` and `\stex_uri_from_current_file_nolang:Nn`.
These functions are documented on page 136.)

```

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

```

```

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

```

(End of definition for `\stex_uri_add_module:NNn`. This function is documented on page [136](#).)

`\l_stex_current_doc_uri`

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

(End of definition for `\l_stex_current_doc_uri`. This variable is documented on page [136](#).)

`\stex_set_document_uri:`

```

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

(End of definition for `\stex_set_document_uri:`. This function is documented on page [136](#).)

`\stex_set_current_namespace:`

```

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

(End of definition for `\stex_set_current_namespace:`. This function is documented on page [136](#).)

We determine the PWD

```

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

(End of definition for `\c_stex_pwd_file` and `\c_stex_main_file`. These variables are documented on page [135](#).)

13.2.9 File Hooks

keeps track of file changes:

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

`\stex_filestack_push:n`

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

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

`\stex_filestack_pop:`

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

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

Hooks for the current file:

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

```

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

```

13.3 Math Archives

```

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

```

```

1073 \exp_args:NNe \stex_file_set:Nn \c_stex_mathhub_file {\stex_file_use:N \c_stex_home_file \
1074 \seq_clear:N \c_stex_mathhub_file
1075 }{
1076 \stex_file_resolve:No \c_stex_mathhub_file \l_stex_mathhub_str
1077 \stex_if_file_absolute:NF \c_stex_mathhub_file {
1078   \exp_args:NNe \stex_file_resolve:Nn \c_stex_mathhub_file {
1079     \stex_file_use:N \c_stex_main_file / .. / \l_stex_mathhub_str
1080   }
1081 }
1082 }
1083
1084 \exp_args:NNe \str_set:Nn \mathhub {\stex_file_use:N \c_stex_mathhub_file}
1085 \stex_debug:nn{mathhub}{MATHHUB:~\mathhub}

```

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

```

\l_stex_current_archive
\stex_set_current_archive:n
1086 \cs_new_protected:Nn \stex_set_current_archive:n {
1087   \stex_require_archive:n { #1 }
1088   \stex_debug:nn{mathhub}{switching-to-archive-#1}
1089   \prop_set_eq:Nc \l_stex_current_archive_prop {
1090     c_stex_mathhub_#1_manifest_prop
1091   }
1092 }

```

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

```

\stex_in_archive:nn
1093 \cs_new_protected:Nn \stex_in_archive:nn {
1094   \cs_if_exist:NF \l_stex_mathhub_cs {
1095     \cs_set:Npn \l_stex_mathhub_cs ##1 {}
1096   }
1097   \stex_pseudogroup:nn{
1098     \cs_set:Npn \l_stex_mathhub_cs ##1 {#2}
1099     \tl_if_empty:nTF{#1} {
1100       \prop_if_exist:NTF \l_stex_current_archive_prop {
1101         \exp_args:Ne \l_stex_mathhub_cs {\prop_item:Nn \l_stex_current_archive_prop { id }}
1102       }{
1103         \l_stex_mathhub_cs {}
1104       }
1105     }{
1106       \stex_set_current_archive:n{#1}
1107       \l_stex_mathhub_cs {#1}
1108     }
1109   }{
1110     \stex_pseudogroup_restore:N \l_stex_current_archive_prop
1111     \cs_set:Npn \exp_not:N \l_stex_mathhub_cs ##1 {
1112       \exp_args:No \exp_not:n {\l_stex_mathhub_cs {##1}}
1113     }
1114   }
1115 }

```

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

```

\stex_require_archive:n
\stex_require_archive:o
1116 \cs_new_protected:Nn \stex_require_archive:n {
1117     \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
1118         \seq_if_empty:NTF \c_stex_mathhub_file {
1119             \stex_fatal_error:n{warning/nomathhub}
1120         }{
1121             \stex_debug:nn{mathhub}{Opening~archive:~#1}
1122             \__stex_mathhub_do_manifest:n { #1 }
1123         }
1124     }
1125 }
1126 \cs_generate_variant:Nn \stex_require_archive:n {o}

```

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

Code for finding and parsing manifest files:

```

1127 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
1128     \exp_args:Ne \__stex_mathhub_find_manifest:n {\stex_file_use:N \c_stex_mathhub_file / #1}
1129     \str_if_empty:NT \l__stex_mathhub_manifest_str {
1130         \stex_fatal_error:nxx{error/noarchive}
1131         {#1}\stex_file_use:N \c_stex_mathhub_file
1132     }
1133     \__stex_mathhub_parse_manifest:n {#1}
1134 }
1135
1136 \cs_new_protected:Nn \__stex_mathhub_find_manifest:n {
1137     \str_clear:N \l__stex_mathhub_manifest_str
1138     \seq_set_split:Nnn \l__stex_mathhub_seq / {#1}
1139     \bool_set_true:N \l__stex_mathhub_bool
1140     \bool_while_do:Nn \l__stex_mathhub_bool {
1141         \tl_if_eq:NNTF \l__stex_mathhub_seq \c_stex_mathhub_file {
1142             \bool_set_false:N \l__stex_mathhub_bool
1143         }{
1144             \__stex_mathhub_check_manifest:
1145             \bool_if:NT \l__stex_mathhub_bool {
1146                 \seq_pop_right:NN \l__stex_mathhub_seq \l__stex_mathhub_tl
1147             }
1148         }
1149     }
1150 }
1151 \cs_generate_variant:Nn \__stex_mathhub_find_manifest:n {x}
1152
1153 \cs_new_protected:Nn \__stex_mathhub_check_manifest: {
1154     \__stex_mathhub_check_manifest:n {MANIFEST.MF}
1155     \bool_if:NT \l__stex_mathhub_bool {
1156         \__stex_mathhub_check_manifest:n {META-INF/MANIFEST.MF}
1157         \bool_if:NT \l__stex_mathhub_bool {
1158             \__stex_mathhub_check_manifest:n {meta-inf/MANIFEST.MF}
1159         }
1160     }
1161 }
1162
1163 \cs_new_protected:Nn \__stex_mathhub_check_manifest:n {
1164     \stex_debug:nn{mathhub}{Checking~\stex_file_use:N \l__stex_mathhub_seq / #1}
1165     \file_if_exist:nT {\stex_file_use:N \l__stex_mathhub_seq / #1} {

```

```

1166     \bool_set_false:N \l__stex_mathhub_bool
1167     \str_set:Nx \l__stex_mathhub_manifest_str {\stex_file_use:N \l__stex_mathhub_seq / #1}
1168 }
1169 }
1170
1171 \ior_new:N \c__stex_mathhub_manifest_ior
1172 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
1173     \ior_open:Nn \c__stex_mathhub_manifest_ior \l__stex_mathhub_manifest_str
1174     \prop_clear:N \l__stex_mathhub_prop
1175     \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
1176         \exp_args:NNNo \exp_args:NNN
1177             \seq_set_split:Nnn \l__stex_mathhub_seq \c_colon_str {\tl_to_str:n{##1}}
1178         \seq_pop_left:NNT \l__stex_mathhub_seq \l__stex_mathhub_key {
1179             \exp_args:NNo \str_set:Nn \l__stex_mathhub_key \l__stex_mathhub_key
1180             \str_set:Nx \l__stex_mathhub_val {\seq_use:Nn \l__stex_mathhub_seq :}
1181             \str_case:Nn \l__stex_mathhub_key {
1182                 {id} {\prop_put:Nno \l__stex_mathhub_prop { id } \l__stex_mathhub_}
1183                 {narration-base} {\prop_put:Nno \l__stex_mathhub_prop { narr } \l__stex_mathhub_}
1184                 {furl-base} {\prop_put:Nno \l__stex_mathhub_prop { docurl } \l__stex_mathhub_}
1185                 {source-base} {\prop_put:Nno \l__stex_mathhub_prop { ns } \l__stex_mathhub_}
1186                 {ns} {\prop_put:Nno \l__stex_mathhub_prop { ns } \l__stex_mathhub_}
1187             }
1188         }
1189     }
1190     \ior_close:N \c__stex_mathhub_manifest_ior
1191     \prop_gset_eq:cN { c_stex_mathhub_#1_manifest_prop } \l__stex_mathhub_prop
1192     \stex_debug:nn{mathhub}{Result:~\prop_to_keyval:N \l__stex_mathhub_prop}
1193     \stex_persist:e {
1194         \prop_gset_from_keyval:cn {c_stex_mathhub_#1_manifest_prop} {
1195             \prop_to_keyval:N \l__stex_mathhub_prop
1196         }
1197     }
1198 }

```

Current MathHub archive

```

\c_stex_main_archive_prop
\l_stex_current_archive_prop
1199 \cs_new_protected:Nn \stex_main_archive: {
1200     \stex_if_file_starts_with:NNTF \c_stex_pwd_file \c_stex_mathhub_file {
1201         \__stex_mathhub_find_manifest:x { \stex_file_use:N \c_stex_pwd_file }
1202         \str_if_empty:NTF \l__stex_mathhub_manifest_str {
1203             \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~archive}
1204         }{
1205             \__stex_mathhub_parse_manifest:n { main }
1206             \prop_set_eq:NN \c_stex_main_archive_prop \c_stex_mathhub_main_manifest_prop
1207             \cs_undefine:N \c_stex_mathhub_main_manifest_prop
1208             \prop_get:NnN \c_stex_main_archive_prop {id}
1209                 \l__stex_mathhub_str
1210             \prop_set_eq:cN { c_stex_mathhub_\l__stex_mathhub_str _manifest_prop }
1211                 \c_stex_main_archive_prop
1212             \exp_args:No \stex_set_current_archive:n { \l__stex_mathhub_str }
1213             \stex_debug:nn{mathhub}{Current~archive:~}
1214                 \prop_item:Nn \l_stex_current_archive_prop {id}
1215             }
1216             \bool_if:NT \c_stex_persist_write_mode_bool {

```

```

1217   \tl_put_right:Nx \_stex_persist_read_now: {
1218     \stex_persist:n {
1219       \prop_gset_from_keyval:cN {c_stex_mathhub_\l_stex_mathhub_str _manifest_prop}{}
1220         \prop_to_keyval:N \c_stex_main_archive_prop
1221       }
1222       \prop_gset_eq:Nc \exp_not:N \l_stex_current_archive_prop {
1223         c_stex_mathhub_\l_stex_mathhub_str _manifest_prop
1224       }
1225       \prop_gset_eq:Nc \exp_not:N \c_stex_main_archive_prop {
1226         c_stex_mathhub_\l_stex_mathhub_str _manifest_prop
1227       }
1228     }
1229   }
1230 }
1231 }{
1232   \stex_debug:nn{mathhub}{Not~currently~in~the~MathHub~directory}
1233 }
1234 }
1235 }

1236 \%bool_if:NF \c_stex_persist_mode_bool {
1237   \_stex_main_archive:
1238 }
1239 }
```

(End of definition for `\c_stex_main_archive_prop` and `\l_stex_current_archive_prop`. These variables are documented on page 131.)

13.4 Documents

13.4.1 Title

Stores the title, if it exists:

```

1240 <@=stex_doc>
1241 \tl_new:N \g__stex_doc_title_tl
```

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

```

1242 \cs_new_protected:Npn \stexdoctitle #1 {
1243   \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1244   \global\def\stexdoctitle##1{}
1245 }
```

At begin document, we switch to:

```

1246 \cs_new_protected:Nn \__stex_doc_set_title:n {
1247   \stex_if_smsmode:F{
1248     \global\def\stexdoctitle##1{}
1249     \stex_debug:nn{title}{Setting~title~to:\tl_to_str:n##1}
1250     \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1251     \__stex_doc_title_html:
1252   }
1253 }
```

Hooks, changes and HTML:

```

1254 \cs_new_protected:Nn \__stex_doc_title_html: {
1255   \stex_if_do_html:T{\stex_if_html_backend:T{
```

```

1256     \stex_annotation_invisible:nn{shml:doctitle={}}{ \hbox{\g__stex_doc_title_tl} }
1257   }
1258 }
1259
1260 \AtBeginDocument {
1261   \tl_if_empty:NTF \g__stex_doc_title_tl {
1262     \cs_set_eq:NN \stexdoctitle \__stex_doc_set_title:n
1263   }{
1264     \stex_debug:nn{title}{Setting~title~to:\exp_args:No\tl_to_str:n\g__stex_doc_title_tl}
1265     \global\def\stexdoctitle{\#1}
1266     \__stex_doc_title_html:
1267   }
1268
1269   \cs_set_eq:NN \__stex_doc_maketitle: \maketitle
1270   \global\protected\def\maketitle{
1271     \tl_if_empty:NF \@title {
1272       \exp_args:No \stexdoctitle \@title
1273     }
1274     \__stex_doc_maketitle:
1275   }
1276 }

```

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

13.4.2 Sectioning

```

1277 \int_new:N \l_stex_docheader_sect
1278
1279 \tl_set:Nn \stex_current_section_level {document}
1280
1281 \cs_set_protected:Npn \currentsectionlevel {
1282   \stex_if_do_html:TF{
1283     \stex_annotation:nn{shml:currentsectionlevel={},shml:capitalize=false}{}}
1284   }{
1285     \stex_current_section_level
1286   }
1287 \tl_if_exist:NT\xspace\xspace
1288 }
1289
1290 \cs_set_protected:Npn \Currentsectionlevel {
1291   \stex_if_do_html:TF{
1292     \stex_annotation:nn{shml:currentsectionlevel={},shml:capitalize=true}{}}
1293   }{
1294     \exp_args:No \stex_capitalize:n \stex_current_section_level
1295   }
1296 \tl_if_exist:NT\xspace\xspace
1297 }
1298
1299 \stex_if_html_backend:TF {
1300   \cs_new_protected:Nn \sfragment_do_level:nn {
1301     \stexdoctitle{\#2}
1302     \par
1303     \begin{stex_annotation_env}{shml:section={\int_use:N \l_stex_docheader_sect}}
1304       \noindent\stex_annotation:nn{shml:sectiontitle={}}{%

```

```

1305      \_stex_annotation_force_break:n{#2}
1306    }\par
1307  }
1308 \cs_new_protected:Nn \_sfragment_end: {
1309   \end{stex_annotation_env}
1310 }
1311 }{
1312 \cs_new_protected:Nn \_sfragment_do_level:nn {
1313   \stexdoctitle{#2}
1314   \tl_if_empty:NTF \l_stex_key_short_tl {
1315     \use:c{#1}
1316   }{
1317     \exp_args:Nnx \use:nn{\use:c{#1}}{[\exp_args:No \exp_not:n \l_stex_key_short_tl]}
1318   }{#2}
1319   \int_incr:N \l_stex_docheader_sect
1320   \tl_set:Nn \stex_current_section_level{#1}
1321 }
1322 \cs_new_protected:Nn \_sfragment_end: {}
1323 }
1324
1325
1326 \cs_new_protected:Npn \__stex_doc_do_section:n {
1327   \int_case:nnF \l_stex_docheader_sect {
1328     {0}{\cs_if_exist:NTF \thechapter {\_sfragment_do_level:nn{part}}{
1329       \int_incr:N \l_stex_docheader_sect
1330       \__stex_doc_do_section:n
1331     }}
1332     {1}{\cs_if_exist:NTF \thechapter {\_sfragment_do_level:nn{chapter}}{
1333       \int_incr:N \l_stex_docheader_sect
1334       \__stex_doc_do_section:n
1335     }}
1336     {2}{\_sfragment_do_level:nn{section}}
1337     {3}{\_sfragment_do_level:nn{subsection}}
1338     {4}{\_sfragment_do_level:nn{subsubsection}}
1339     {5}{\_sfragment_do_level:nn{paragraph}}
1340   }{\_sfragment_do_level:nn{subparagraph}}
1341 }
1342
1343 \stex_keys_define:nnnn{ sfragment }{
1344   \tl_clear:N \l_stex_key_short_tl
1345 }{
1346   short .tl_set:N = \l_stex_key_short_tl
1347 }{id}
1348
1349 \NewDocumentEnvironment{sfragment}{ O{} m} {
1350   \stex_keys_set:nn{sfragment}{#1}
1351   \__stex_doc_do_section:n{#2}
1352   \stex_do_id:
1353 }{
1354   \_sfragment_end:
1355 }
1356
1357 \%int_incr:N \l_stex_docheader_sect
1358 \NewDocumentEnvironment{blindfragment}{}{

```

```

1359     \__stex_doc_skip_section:
1360 }{
1361     \stex_if_html_backend:T{
1362         \stex_annotate_invisible:n{~}
1363         \end{stex_annotate_env}
1364     }
1365 }
1366
1367 \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
1380 \stex_if_html_backend:TF {
1381     \cs_new_protected:Nn \__stex_doc_skip_section: {
1382         \__stex_doc_skip_section_i:
1383         \begin{stex_annotate_env}{shtml:skipsection={\int_use:N \l_stex_docheader_sect}}
1384         \stex_annotate_invisible:n{~}
1385     }
1386 }
1387 \cs_set_eq:NN \__stex_doc_skip_section: \__stex_doc_skip_section_i:
1388 }
1389
1390
1391 \cs_new_protected:Nn \__stex_doc_skip_fragment:n {
1392     \stepcounter{#1}
1393 }
1394
1395 \cs_new_protected:Npn \skipfragment {
1396     \int_case:nnF \l_stex_docheader_sect {
1397         {0}{\cs_if_exist:NTF \thepart {\__stex_doc_skip_fragment:n{part}}{
1398             \int_incr:N \l_stex_docheader_sect
1399             \skipfragment
1400         }}
1401         {1}{\cs_if_exist:NTF \thechapter {\__stex_doc_skip_fragment:n{chapter}}{
1402             \int_incr:N \l_stex_docheader_sect
1403             \skipfragment
1404         }}
1405         {2}{\__stex_doc_skip_fragment:n{section}}
1406         {3}{\__stex_doc_skip_fragment:n{subsection}}
1407         {4}{\__stex_doc_skip_fragment:n{subsubsection}}
1408         {5}{\__stex_doc_skip_fragment:n{paragraph}}
1409     }{\__stex_doc_skip_fragment:n{subparagraph}}
1410 }

```

\setsectionlevel

```

1411 \cs_new_protected:Npn \setsectionlevel #1 {
1412   \str_case:nnF{#1}{
1413     {part}{\int_set:Nn \l_stex_docheader_sect 0}
1414     {chapter}{\int_set:Nn \l_stex_docheader_sect 1}
1415     {section}{\int_set:Nn \l_stex_docheader_sect 2}
1416     {subsection}{\int_set:Nn \l_stex_docheader_sect 3}
1417     {subsubsection}{\int_set:Nn \l_stex_docheader_sect 4}
1418     {paragraph}{\int_set:Nn \l_stex_docheader_sect 5}
1419   }{
1420     \int_set:Nn \l_stex_docheader_sect 6
1421   }
1422   \cs_if_eq:NNTF \onlypreamble \notprerr {
1423     \stex_annotation_invisible:nn{shml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
1424   }{}
1425 }
1426
1427 \stex_if_html_backend:T{
1428   \cs_new_protected:Nn \__stex_doc_check_topsect: {
1429     \int_case:nnF \l_stex_docheader_sect {
1430       {0}{\cs_if_exist:NTF \thepart {
1431         \stex_annotation_invisible:nn{shml:sectionlevel=0}{}}
1432       }{
1433         \int_incr:N \l_stex_docheader_sect
1434         \__stex_doc_check_topsect:
1435       }
1436       {1}{\cs_if_exist:NTF \thechapter {
1437         \stex_annotation_invisible:nn{shml:sectionlevel=1}{}}
1438       }{
1439         \int_incr:N \l_stex_docheader_sect
1440         \__stex_doc_check_topsect:
1441       }
1442     }{
1443       \stex_annotation_invisible:nn{shml:sectionlevel={\int_use:N\l_stex_docheader_sect}}{}
1444     }
1445   }
1446   \AtBeginDocument{\__stex_doc_check_topsect:}
1447 }
1448
1449 \AtBeginDocument{
1450   \bool_if:NF \c_stex_no_frontmatter_bool {
1451     \cs_if_exist:NTF \frontmatter {
1452       \let\__stex_doc_orig_frontmatter \frontmatter
1453       \let\frontmatter \relax
1454     }{
1455       \tl_set:Nn \__stex_doc_orig_frontmatter {
1456         \clearpage
1457         \%@mainmatterfalse
1458         \pagenumbering{roman}
1459       }
1460     }
1461     \cs_if_exist:NTF \backmatter {
1462       \let\__stex_doc_orig_backmatter \backmatter
1463       \let\backmatter \relax
1464     }

```

```

1465     \tl_set:Nn\__stex_doc_orig_backmatter{
1466         \clearpage
1467         \%@\mainmatterfalse
1468         \pagenumbering{roman}
1469     }
1470 }
1471 \newenvironment{frontmatter} {
1472     \__stex_doc_orig_frontmatter
1473 }{
1474     \cs_if_exist:NTF\mainmatter{
1475         \mainmatter
1476     }{
1477         \clearpage
1478         \%@\mainmattertrue
1479         \pagenumbering{arabic}
1480     }
1481 }
1482 \newenvironment{backmatter} {
1483     \__stex_doc_orig_backmatter
1484 }{
1485     \cs_if_exist:NTF\mainmatter{
1486         \mainmatter
1487     }{
1488         \clearpage
1489         \%@\mainmattertrue
1490         \pagenumbering{arabic}
1491     }
1492 }
1493 }
1494 }

```

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

13.4.3 References

```
1495 <@=stex_refs>
```

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

```

1496 \iow_new:N \c__stex_refs_iow
1497 \AtBeginDocument{\iow_open:Nn \c__stex_refs_iow {\jobname.sref}}
1498 \AtEndDocument{\iow_close:N \c__stex_refs_iow}
```

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

```

1499 \cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}
1500
1501 \cs_new_protected:Npn \STEXInternalSetSrefSymURL #1 #2 {
1502     \str_gset:cn{g_stex_sref_sym_\tl_to_str:n{#1}_target}{#2}
1503 }
1504
```

```
\stex_ref_new_doc_target:n
    \sreflabel
1505 \seq_new:N \g__stex_refs_files_seq
1506 \int_new:N \l__stex_refs_unnamed_counter_int
```

```

1507 \cs_new_protected:Nn \_stex_ref_new_id:n {
1508   \str_if_empty:nTF {#1} {
1509     \int_gincr:N \l__stex_refs_unnamed_counter_int
1510     \str_set:Nx \l__stex_refs_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1511   }{
1512     \str_set:Nn \l__stex_refs_str {#1}
1513   }
1514   \str_set:Nx \l_stex_ref_url_str {\stex_uri_use:N \l_stex_current_doc_uri ? \l__stex_refs_s
1515 }
1516 }

1517 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1518   \stex_ref_new_id:n{#1}
1519   \%stex_uri_add_module:NNo \l__stex_refs_uri \l_stex_current_doc_uri \l__stex_refs_str
1520   \%stex_debug:nn{sref}{New-document-target:~\stex_uri_use:N \l__stex_refs_uri}
1521   \__stex_refs_add_doc_ref:xo {\stex_uri_use:N \l_stex_current_doc_uri} \l__stex_refs_str
1522   \stex_if_smemode:F {
1523     \iow_now:Nx \c__stex_refs_iow {
1524       \STEXInternalSrefRestoreTarget
1525       {\stex_uri_use:N \l_stex_current_doc_uri}
1526       {\l__stex_refs_str}
1527       {\l@currentcounter}
1528       {\l@currentlabel}
1529       {
1530         \tl_if_exist:NT\l@currentlabelname{
1531           \exp_args:No\exp_not:n\l@currentlabelname
1532         }
1533       }
1534     }
1535   }
1536   \exp_args:Nx \label {sref@\l_stex_ref_url_str}
1537   \stex_if_do_html:T {
1538     \pdfdest name "sref@\l_stex_ref_url_str" xyz\relax
1539   }
1540 }
1541 }
1542 \NewDocumentCommand \sreflabel {m} {\stex_ref_new_doc_target:n {#1}}
1543
1544 \cs_new_protected:Nn \__stex_refs_add_doc_ref:nn {
1545   \seq_if_in:NnTF \g__stex_refs_files_seq {#1} {
1546     \seq_if_in:cnF {g__stex_refs_#1_seq}{#2} {
1547       \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1548     }
1549   }{
1550     \seq_gput_right:Nn \g__stex_refs_files_seq {#1}
1551     \seq_new:c{g__stex_refs_#1_seq}
1552     \seq_gput_left:cn{g__stex_refs_#1_seq}{#2}
1553   }
1554 }
1555 \cs_generate_variant:Nn \__stex_refs_add_doc_ref:nn {xo,xx}

```

(End of definition for `\stex_ref_new_doc_target:n` and `\sreflabel`. These functions are documented on page 118.)

\sref Optional arguments:
\extref

```

1556 \stex_keys_define:nnnn{sref / 1}{}{
1557   % TODO get rid of this
1558   fallback .code:n = {},
1559   pre     .code:n = {},
1560   post    .code:n = {}
1561 }{archive file}
1562 \stex_keys_define:nnnn{sref / 2}{}{}{archive file, title}
1563
1564 \str_new:N \l__stex_refs_default_archive_str
1565 \str_new:N \l__stex_refs_default_file_str
1566 \tl_new:N \l__stex_refs_default_title_tl
1567
1568 \cs_set_protected:Nn \__stex_refs_set_keys_b:n {
1569   \tl_if_empty:nTF{#1}{
1570     \str_set_eq:NN \l_stex_key_archive_str \l__stex_refs_default_archive_str
1571     \str_set_eq:NN \l_stex_key_file_str \l__stex_refs_default_file_str
1572     \tl_set_eq:NN \l_stex_key_title_tl \l__stex_refs_default_title_tl
1573   }{
1574     \stex_keys_set:nn{ sref / 2 }{ #1 }
1575   }
1576 }
1577
1578 \newcommand\srefsetin[3][]{%
1579   \str_set:Nx \l__stex_refs_default_archive_str {#1}
1580   \str_set:Nx \l__stex_refs_default_file_str {#2}
1581   \tl_set:Nn \l__stex_refs_default_title_tl {#3}
1582 }
1583

Auxiliary methods:

1584 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1585   \str_clear:N \l__stex_refs_uri_str
1586   \stex_debug:nn{sref}{%
1587     File:~\l_stex_key_file_str^~J
1588     Repo:\l_stex_key_archive_str
1589   }
1590   \str_if_empty:NTF \l_stex_key_file_str {
1591     \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1592     \seq_if_exist:cT{\g__stex_refs_\stex_uri_use:N \l_stex_current_doc_uri _seq}{%
1593       \exp_args:Nnx \__stex_refs_find_uri_in_file:nnn{#1}%
1594         {\l_stex_uri_use:N \l_stex_current_doc_uri}\seq_map_break:
1595     }
1596     \str_if_empty:NT \l__stex_refs_uri_str {
1597       \seq_map_inline:Nn \g__stex_refs_files_seq {
1598         \__stex_refs_find_uri_in_file:nnn{#1}{##1}{\seq_map_break:n{\seq_map_break:}}
1599       }
1600     }
1601   }{%
1602     \str_if_empty:NTF \l_stex_key_archive_str {
1603       \prop_if_exist:NTF \l_stex_current_archive_prop {
1604         \__stex_refs_find_uri_in_prop_file:N \l_stex_current_archive_prop
1605       }{%
1606         \stex_file_resolve:Nx \l__stex_refs_file
1607           { \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str }
1608       }
1609     }
1610   }
1611 }
```

```

1608     \str_set:Nx \l__stex_refs_uri_str { file:/ \stex_file_use:N \l__stex_refs_file }
1609     }
1610     }{
1611         \stex_require_archive:o \l_stex_key_archive_str
1612         \prop_set_eq:Nc \l_stex_refs_prop { c_stex_mathhub_\l_stex_key_archive_str _manifest_
1613             \__stex_refs_find_uri_in_prop_file:N \l__stex_refs_prop
1614         }
1615     }
1616 }
1617
1618 \cs_new_protected:Nn \__stex_refs_find_uri_in_prop_file:N {
1619     \str_set:Nx \l__stex_refs_uri_str {
1620         \stex_file_use:N \c_stex_mathhub_file /
1621         \prop_item:Nn #1 {id} /
1622             source / \l_stex_key_file_str .sref
1623     }
1624     \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_uri_str
1625     \stex_uri_from_repo_file:NNNn \l__stex_refs_uri #1
1626         \l__stex_refs_file {narr}
1627     \str_set:Nx \l__stex_refs_uri_str {\stex_uri_use:N \l__stex_refs_uri}
1628 }
1629
1630 \cs_new_protected:Nn \__stex_refs_find_uri_in_file:nnn {
1631     \stex_debug:nn{sref}{Checking~file~#2}
1632     \seq_map_inline:cn{g__stex_refs_#2_seq}{%
1633         \str_if_eq:nnT{#1}{##1}{%
1634             \str_set:Nx \l__stex_refs_uri_str {\stex_uri_use:N \l_stex_current_doc_uri}
1635             \stex_debug:nn{sref}{Found.}
1636             #3
1637         }
1638     }
1639 }

```

Doing the actual referencing:

```

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

```

```

1660 \stex_debug:nn{sref}{Using~ref~file~\l_stex_refs_uri_str}
1661 \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref_\l_stex_refs_uri_str_seq}{\detokenize{\#1}{}}
1662   \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref~on~\l_stex_refs_uri_str?}
1663   \l_stex_refs_do_autoref:n{\l_stex_refs_uri_str?\#1}
1664 }{
1665   \str_if_empty:NTF \l_stex_key_file_str {
1666     \stex_debug:nn{sref}{in~empty;~autoref~on~\l_stex_refs_uri_str?\#1}
1667     \l_stex_refs_do_autoref:n{\l_stex_refs_uri_str?\#1}
1668   }{
1669     \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l_stex_refs_uri_str?\#1}
1670     \l_stex_refs_set_keys_b:n{ \#2 }
1671     \l_stex_refs_do_sref_in:n{\#1}
1672   }
1673 }
1674 }{
1675   \stex_debug:nn{sref}{No~ref~file~found~for~\l_stex_refs_uri_str}
1676   \str_if_empty:NTF \l_stex_key_file_str {
1677     \stex_debug:nn{sref}{in~empty;~autoref~on~\l_stex_refs_uri_str?\#1}
1678     \l_stex_refs_do_autoref:n{\l_stex_refs_uri_str?\#1}
1679   }{
1680     \stex_debug:nn{sref}{in~non~empty;~srefin~on~\l_stex_refs_uri_str?\#1}
1681     \l_stex_refs_set_keys_b:n{ \#2 }
1682     \l_stex_refs_do_sref_in:n{\#1}
1683   }
1684 }
1685 }
1686 }
1687
1688 \cs_new_protected:Nn \l_stex_refs_do_sref_in:n {
1689   \stex_debug:nn{sref}{In: \l_stex_key_file_str^JRepo:\l_stex_key_archive_str}
1690   \stex_debug:nn{sref}{URI: \l_stex_refs_uri_str?\#1}
1691   \tl_if_exist:cTF{r@sref@\l_stex_refs_uri_str?\#1}){
1692     \l_stex_refs_do_autoref:n{\l_stex_refs_uri_str?\#1}
1693   }{
1694     \%msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1695     \group_begin:\catcode13=9\relax\catcode10=9\relax
1696       \str_if_empty:NTF \l_stex_key_archive_str {
1697         \prop_if_exist:NTF \l_stex_current_archive_prop {
1698           \str_set:Nx \l_stex_refs_file_str {
1699             \stex_file_use:N \c_stex_mathhub_file /
1700             \prop_item:Nn \l_stex_current_archive_prop { id }
1701             / source / \l_stex_key_file_str .sref
1702           }
1703         }{
1704           \str_set:Nx \l_stex_refs_file_str {
1705             \stex_file_use:N \g_stex_current_file / .. / \l_stex_key_file_str . sref
1706           }
1707         }
1708       }{
1709         \str_set:Nx \l_stex_refs_file_str {
1710           \stex_file_use:N \c_stex_mathhub_file / \l_stex_key_archive_str
1711             / source / \l_stex_key_file_str .sref
1712         }
1713     }

```

```

1714     \stex_file_resolve:No \l__stex_refs_file \l__stex_refs_file_str
1715     \str_set:Nx \l__stex_refs_file_str {\stex_file_use:N \l__stex_refs_file }
1716     \stex_debug:nn{sref}{File: \l__stex_refs_file_str }
1717     \exp_args:NNNx \exp_args:No \str_if_eq:nnTF \l__stex_refs_file_str {\stex_file_use:N\l_
1718         \l__stex_refs_do_autoref:n{
1719             \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1720         }
1721     }{
1722         \exp_args:No \IfFileExists \l__stex_refs_file_str {
1723             \tl_clear:N \l__stex_refs_return_tl
1724             \str_set:Nn \l__stex_refs_id_str {#1}
1725             \let\STEXInternalSrefRestoreTarget\l__stex_refs_restore_target:nnnn
1726             \use:c{@ @ input}{\l__stex_refs_file_str}
1727             \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1728                 \exp_args:Nnno \msg_warning:nnnn{sstex}{warning/smslabelmissing}\l__stex_refs_file_
1729                 \l__stex_refs_do_autoref:n{
1730                     \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1731                 }
1732             }{
1733                 \l__stex_refs_return_tl
1734             }
1735         }{
1736             \exp_args:Nnno \msg_warning:nnnn{sstex}{warning/smsmissing}\l__stex_refs_file_str
1737             \l__stex_refs_do_autoref:n{
1738                 \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1739             }
1740         }
1741     }
1742     \group_end:
1743 }
1744 }
1745
1746 \cs_new_protected:Nn \l__stex_refs_do_return:nnnn {
1747     \tl_set:Nn \l__stex_refs_return_tl {
1748         \stex_annotation:nn{shtml:sref={#4},shtml:srefin={\l__stex_refs_file_str}}{
1749             \use:c{#3autorefname}~#1\tl_if_empty:nF{#2}{~(#2)}
1750             \tl_if_empty:NF\l_stex_key_title_tl{
1751                 {}~in~\l_stex_key_title_tl
1752             }
1753         }
1754     }
1755 }
1756
1757 \cs_new_protected:Nn \l__stex_refs_restore_target:nnnn {
1758     \str_if_empty:NTF \l__stex_refs_uri_str {
1759         \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1760             \l__stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
1761         }
1762     }{
1763         \stex_debug:nn{sref}{\l__stex_refs_uri_str{}~ == ~ #1 ~ ?}
1764         \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}){
1765             \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1766             \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1767                 \stex_debug:nn{sref}{success!}

```

```

1768     \__stex_refs_do_return:nnnn{#4}{#5}{#3}{#1?#2}
1769     \endinput
1770   }
1771 }
1772 }
1773 }
```

The actual macros:

```

1774 \NewDocumentCommand \sref { O{} m O{} }{
1775   \stex_keys_set:nn { sref / 1 }{ #1 }
1776   \__stex_refs_find_uri:n { #2 }
1777   \__stex_refs_do_sref:nn{#2}{#3}
1778 }
1779 \NewDocumentCommand \extref { O{} m m }{
1780   \stex_keys_set:nn { sref / 1 }{ #1 }
1781   \__stex_refs_find_uri:n { #2 }
1782   \__stex_refs_set_keys_b:n{ #3 }
1783   \str_if_empty:NT \l_stex_key_file_str {
1784     \msg_error:nn{stex}{error/extrefmissing}
1785   }
1786   \__stex_refs_do_sref_in:n{#2}
1787 }
```

(End of definition for `\sref` and `\extref`. These functions are documented on page 81.)

```

\stex_ref_new_sym_target:n
1788 \cs_new_protected:Nn \stex_ref_new_symbol:n {
1789   \cs_if_exist:cF{r@sref@sym@tl_to_str:n{#1}}{
1790     \__stex_refs_new_symbol:n{#1}
1791   }
1792 }
1793
1794 \cs_new_protected:Nn \stex_sref_do_aux:n {
1795   #1 \iow_now:Nn \auxout {#1}
1796 }
1797
1798 \cs_new_protected:Nn \__stex_refs_new_symbol:n {
1799   \prop_if_exist:NTF \l_stex_current_archive_prop {
1800     \prop_get:NnTF \l_stex_current_archive_prop {docurl} \l__stex_refs_str {
1801       \exp_args:Ne \stex_sref_do_aux:n {
1802         \STEXInternalSetSrefSymURL{#1}{\l__stex_refs_str / symbol? #1}
1803       }
1804     }{
1805       \stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
1806     }
1807   }{
1808     \stex_sref_do_aux:n { \STEXInternalSetSrefSymURL{#1}{} }
1809   }
1810 }
1811
1812 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1813   \cs_if_exist:NT \hypertarget{
1814     \exp_args:Ne \hypertarget{\tl_to_str:n{sref@sym@ #1}}{}
1815     \str_gset:cx{\tl_to_str:n{r@sref@sym@ #1}}{\tl_to_str:n{sref@sym@ #1}}
1816   }
```

```

1817 }
1818
1819 \cs_new_protected:Nn \stex_ref_new_sym_target:nn {
1820   \str_if_eq:nnTF{#1}{#2} {
1821     \stex_ref_new_sym_target:n{#1}
1822   }{
1823     \str_gset:cn{g_stex_sref_sym_ #1 _label}{#2}
1824   }
1825 }

```

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

`\srefsym`

```

1826 \NewDocumentCommand \srefsym { m m }{
1827   \stex_get_symbol:n { #1 }
1828   \exp_args:Ne
1829   \__stex_refs_sym_aux:nn{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
1830 }
1831
1832 \cs_new_protected:Npn \__stex_refs_do_internal_link:nn #1 {
1833   \cs_if_exist:NTF \hyperlink {
1834     \hyperlink{#1}
1835   }\use:n
1836 }
1837
1838 \cs_new_protected:Npn \__stex_refs_do_url_link:nn {
1839   \cs_if_exist:NTF \href \href \use_i:nn
1840 }
1841
1842 \cs_new_protected:Npn \__stex_refs_sym_aux:nn #1 {
1843   \cs_if_exist:cTF{\tl_to_str:n{r@sref@sym@#1}}{
1844     \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1845   }{
1846     \str_if_exist:cTF{g_stex_sref_sym_#1_label}{
1847       \exp_args:Ne \__stex_refs_sym_aux:nn{\use:c{g_stex_sref_sym_#1_label}}
1848     }{
1849       \str_if_empty:cTF{g_stex_sref_sym_#1_target}{
1850         \exp_args:Ne \__stex_refs_do_internal_link:nn{\tl_to_str:n{sref@sym@#1}}
1851       }{
1852         \exp_args:Ne \__stex_refs_do_url_link:nn{\use:c{g_stex_sref_sym_#1_target}}
1853       }
1854     }
1855   }
1856 }

```

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

`\srefsymuri`

```

1857 \cs_new_protected:Npn \srefsymuri #1 {
1858   \__stex_refs_sym_aux:nn{#1}
1859 }

```

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

13.4.4 Inputs

```

1860 <@=stex_inputs>

\stex_resolve_path_pair:Nnn
\stex_resolve_path_pair:Nxx
1861 \cs_new_protected:Nn \stex_resolve_path_pair:Nnn {
1862   \stex_debug:nn{resolving-path}{#3~in-[#2]}
1863   \str_if_empty:nTF{#2} {
1864     \prop_if_exist:NTF \l_stex_current_archive_prop {
1865       \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1866       \prop_item:Nn \l_stex_current_archive_prop { id } / source /
1867       #3}
1868       \stex_debug:nn{resolving-path}{In-current-archive-
1869       \prop_item:Nn \l_stex_current_archive_prop { id }
1870       ;~result:~#1}
1871     }{
1872       \str_set:Nx #1 {\stex_file_use:N \c_stex_pwd_file / .. / #3 }
1873       \stex_debug:nn{resolving-path}{No-current-archive;~result:~#1}
1874     }
1875   }{
1876     \stex_require_archive:n { #2 }
1877     \str_set:Nx #1 {\stex_file_use:N \c_stex_mathhub_file /
1878     \prop_item:cn {c_stex_mathhub_#2_manifest_prop} { id } / source /
1879     #3}
1880     \stex_debug:nn{resolving-path}{result:~#1}
1881   }
1882 }
1883 \cs_generate_variant:Nn \stex_resolve_path_pair:Nnn {Nxx}

```

(End of definition for \stex_resolve_path_pair:Nnn. This function is documented on page 131.)

```

\inputref
\mhinput
\ifinputref
1884 \newif \ifinputref \inputreffalse
1885
1886 \cs_new_protected:Nn \__stex_inputs_mhinput:nn {
1887   \stex_in_archive:nn {#1} {
1888     \ifinputref
1889       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1890     \else
1891       \inputreftrue
1892       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / ##1 / source / #2 }
1893       \inputreffalse
1894     \fi
1895   }
1896 }
1897
1898 \NewDocumentCommand \mhinput { O{} m }{
1899   \exp_args:NNx \exp_args:Nnx \__stex_inputs_mhinput:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2} }
1900 }
1901
1902 \cs_new_protected:Nn \__stex_inputs_inputref_html:nn {
1903   \str_clear:N \l_tmpa_str
1904   \prop_get:NnNF \l_stex_current_archive_prop { narr } \l_tmpa_str {
1905     \prop_get:NnNF \l_stex_current_archive_prop { ns } \l_tmpa_str {}
1906   }

```

```

1907 \tl_if_empty:nTF{ #1 }{
1908   \IfFileExists{#2}{
1909     \ifvmode\noindent\fi\stex_annotation_invisible:nn{shtml:inputref=}
1910     \l_tmpa_str / #2
1911   }{}}
1912 }{
1913   \stex_input_with_hooks:n{#2}
1914 }
1915 }{
1916   \IfFileExists{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }{
1917     \ifvmode\noindent\fi\stex_annotation_invisible:nn{shtml:inputref=}
1918     \l_tmpa_str / #2
1919   }{}}
1920 }{
1921   \input{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1922 }
1923 }
1924 }
1925
1926 \cs_new_protected:Nn \__stex_inputs_inputref_pdf:nn {
1927   \begingroup
1928     \inputreftrue
1929     \tl_if_empty:nTF{ #1 }{
1930       \stex_input_with_hooks:n{#2}
1931     }{
1932       \stex_input_with_hooks:n{ \stex_file_use:N \c_stex_mathhub_file / #1 / source / #2 }
1933     }
1934   \endgroup
1935 }
1936
1937 \cs_new_protected:Nn \__stex_inputs_inputref:nn {
1938   \stex_in_archive:nn {#1} {
1939     \stex_if_html_backend:TF
1940       \__stex_inputs_inputref_html:nn
1941       \__stex_inputs_inputref_pdf:nn
1942       {##1}{#2}
1943     }
1944   }
1945
1946 \NewDocumentCommand \inputref { O{} m }{
1947   \exp_args:NNx \exp_args:Nnx \__stex_inputs_inputref:nn{ \tl_to_str:n{#1} }{ \tl_to_str:n{#2} }
1948 }

```

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

\addmhbibresource

```

1949 \cs_new_protected:Nn \__stex_inputs_bibresource:n {
1950   \__stex_inputs_up_archive:nn{#1}{bib}
1951   \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
1952     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\addmhbibresource}{#1.bib}
1953   }{
1954     \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
1955       \addbibresource{ ##1 }

```

```

1956     }
1957   }
1958 }
1959 \newcommand\addmhbibresource[2][]{%
1960   \tl_if_empty:nTF{#1}{%
1961     \__stex_inputs_bibresource:n{#2}%
1962   }{%
1963     \stex_in_archive:nn{#1}{\__stex_inputs_bibresource:n{#2}}%
1964   }%
1965 }

```

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

\IfInputref

```

1966 \stex_if_html_backend:TF{%
1967   \newcommand \IfInputref[2]{%
1968     \stex_annotate:nn{shtml:ifinputref=true}{#1}%
1969     \stex_annotate:nn{shtml:ifinputref=false}{#2}%
1970   }%
1971 }{%
1972   \newcommand \IfInputref[2]{%
1973     \ifinputref #1 \else #2 \fi%
1974   }%
1975 }

```

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

\libinput

```

1976 \cs_new_protected:Nn \__stex_inputs_up_archive:nn {%
1977   \prop_if_exist:NF \l_stex_current_archive_prop {%
1978     \msg_error:nnn{stex}{error/notinarchive}\libinput%
1979   }%
1980   \prop_get:NnNF \l_stex_current_archive_prop {id} \l__stex_inputs_id_str {%
1981     \msg_error:nnn{stex}{error/notinarchive}\libinput%
1982   }%
1983   \seq_clear:N \l__stex_inputs_libinput_files_seq%
1984   \seq_set_eq:NN \l__stex_inputs_path_seq \c_stex_mathhub_file%
1985   \seq_set_split:NnV \l__stex_inputs_id_seq / \l__stex_inputs_id_str%
1986 %
1987   \bool_while_do:nn { ! \seq_if_empty_p:N \l_stex_inputs_id_seq }{%
1988     \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / meta-i}%
1989     \IfFileExists{ \l__stex_inputs_path_str }{%
1990       \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_s%
1991         \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str%
1992       }%
1993     }{}%
1994     \seq_pop_left:NN \l__stex_inputs_id_seq \l__stex_inputs_path_str%
1995     \seq_put_right:No \l__stex_inputs_path_seq \l__stex_inputs_path_str%
1996   }%
1997 %
1998   \str_set:Nx \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / lib / #1}%
1999   \IfFileExists{ \l__stex_inputs_path_str }{%
2000     \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_s%
2001       \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str%
2002     }%

```

```

2003     }{}
2004 }
2005
2006 \cs_new_protected:Nn \__stex_inputs_libinput:n {
2007   \__stex_inputs_up_archive:nn{#1}{tex}
2008   \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
2009     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
2010   }{
2011     \seq_map_inline:Nn \l__stex_inputs_libinput_files_seq {
2012       \input{ ##1 }
2013     }
2014   }
2015 }
2016
2017 \newcommand \libinput [2] [] {
2018   \tl_if_empty:nTF{#1} {
2019     \__stex_inputs_libinput:n{#2}
2020   }{
2021     \stex_in_archive:nn{#1}{\__stex_inputs_libinput:n{#2}}
2022   }
2023 }

```

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

`\libusepackage`

```

2024 \newcommand\libusepackage[2] [] {
2025   \__stex_inputs_up_archive:nn{#2}{sty}
2026   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2027     \str_set:Nx \l__stex_inputs_tmp_str {\seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
2028     \exp_args:Nne \use:n {\usepackage[#1]} {
2029       \str_range:Nnn\l__stex_inputs_tmp_str 1 {-5}
2030     }
2031   }{
2032     \stex_fatal_error:nnn{error/nofile}{\libusepackage}{#1.sty}
2033   }
2034 }

```

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

```

\mhgraphics
\cmhgraphics
\lstinputmhlisting
\clstinputmhlisting
\mhtikzinput
\cmhtikzinput
2035 \str_new:N \l__stex_inputs_gin_repo_str
2036 \ltx@ifpackageloaded{graphicx}{\use:n}{\AtEndOfPackageFile{graphicx}}{
2037   \define@key{Gin}{archive}{
2038     \tl_set:Nx\Gin@mrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2039   }
2040   \providecommand\mhgraphics[2] []{
2041     \tl_set:Nx\Gin@mrepos{
2042       \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id}
2043     }
2044     \setkeys{Gin}{#1}
2045     \includegraphics[#1]{ \Gin@mrepos #2 }
2046   }
2047   \providecommand\cmhgraphics[2] []{\begin{center}\mhgraphics[#1]{#2}\end{center}}
2048 }
2049

```

```

2050 \ltx@ifpackageloaded{listings}{\use:n}{\AtEndOfPackageFile{listings}}{
2051   \define@key{lst}{archive}{
2052     \def\lst@mrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2053   }
2054 \newcommand\lstinputmhlisting[2][]{%
2055   \def\lst@mrepos{
2056     \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id}
2057   }
2058   \setkeys{lst}{#1}%
2059   \lstinputlisting[#1]{\lst@mrepos #2}}
2060 \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
2061 }
2062
2063 \ltx@ifpackageloaded{tikzinput}{\use:n}{\AtEndOfPackageFile{tikzinput}}{
2064   \define@key{Gin}{archive}{
2065     \str_set:Nn \l__stex_inputs_gin_repo_str {#1}
2066     \def\Gin@mrepos{\stex_file_use:N \c_stex_mathhub_file / #1 / source /}
2067   }
2068 \newcommand\mhtikzinput[2][]{%
2069   \str_clear:N \l__stex_inputs_gin_repo_str
2070   \def\Gin@mrepos{
2071     \stex_file_use:N \c_stex_mathhub_file / \prop_item:Nn \l_stex_current_archive_prop {id}
2072   }
2073   \setkeys{Gin}{#1}%
2074   \exp_args:No \stex_in_archive:nn \l__stex_inputs_gin_repo_str {
2075     \tikzinput[#1]{\Gin@mrepos #2}
2076   }
2077 }
2078 \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
2079 }

```

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

`\libusetikzlibrary`

```

2080 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary_i:nn {
2081   \pgfkeys@spdef\pgf@temp{#1}
2082   \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
2083   \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfutil@empty
2084   \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode`@}
2085   \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode`\|}%
2086   \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode`\$}
2087   \catcode`\@=11
2088   \catcode`\|=12
2089   \catcode`\$=3
2090   \pgfutil@InputIfFileExists{#2}{}{%
2091     \catcode`\@=\csname tikz@library@#1@atcode\endcsname
2092     \catcode`\|= \csname tikz@library@#1@barcode\endcsname
2093     \catcode`\$= \csname tikz@library@#1@dollarcode\endcsname
2094   }
2095
2096 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary:n{
2097   \__stex_inputs_up_archive:nn{tikzlibrary#1}{code.tex}
2098   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2099     \exp_args:Nne \__stex_inputs_usetikzlibrary_i:nn{#1}{\seq_item:Nn \l__stex_inputs_libin

```

```

2100 }{
2101   \stex_fatal_error:n{error/nofile}{\libusetikzlibrary}{tikzlibrary#1.code.tex}
2102 }
2103 }
2104
2105 \newcommand \libusetikzlibrary [2] [] {
2106   \cs_if_exist:N \usetikzlibrary {
2107     \msg_error:nnx{\stex}{error/notikz}{\tl_to_str:n{\libusetikzlibrary}}
2108   }
2109   \tl_if_empty:nTF{#1} {
2110     \__stex_inputs_usetikzlibrary:n{#2}
2111   }{
2112     \stex_in_archive:nn{#1}{\__stex_inputs_usetikzlibrary:n{#2}}
2113   }
2114 }

```

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

13.5 SMS Mode

```
2115 <@=stex_smsmode>
```

Macros and environments allowed in sms mode:

```

2116 \tl_new:N \g__stex_smsmode_allowed_tl
2117 \tl_new:N \g__stex_smsmode_allowed_escape_tl
2118 \seq_new:N \g__stex_smsmode_allowedenvs_seq

```

```
\stex_sms_allow:N
```

```
\stex_sms_allow_escape:N
```

```
\stex_sms_allow_env:n
```

```

2119 \cs_new_protected:Nn \stex_sms_allow:N {
2120   \tl_gput_right:Nn \g__stex_smsmode_allowed_tl {#1}
2121 }
2122
2123 \cs_new_protected:Nn \stex_sms_allow_escape:N {
2124   \tl_gput_right:Nn \g__stex_smsmode_allowed_escape_tl {#1}
2125 }
2126
2127 \cs_new_protected:Nn \stex_sms_allow_env:n {
2128   \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowedenvs_seq {\tl_to_str:n{#1}}
2129 }

```

(End of definition for `\stex_sms_allow:N`, `\stex_sms_allow_escape:N`, and `\stex_sms_allow_env:n`. These functions are documented on page 132.)

Some initial allowed macros:

```

2130 \stex_sms_allow:N \makeatletter
2131 \stex_sms_allow:N \makeatother
2132 \stex_sms_allow:N \ExplSyntaxOn
2133 \stex_sms_allow:N \ExplSyntaxOff
2134 \stex_sms_allow:N \rustexBREAK

```

```
\stex_if_smsmode_p:
```

```
\stex_if_smsmode:TF
```

```

2135 \bool_new:N \g__stex_smsmode_bool
2136 \bool_set_false:N \g__stex_smsmode_bool
2137 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2138   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2139 }

```

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

```
\stex_sms_allow_import:Nn
  \stex_sms_allow_import_env:nn
    2140  \tl_new:N \g__stex_smsmode_allowed_import_tl
    2141  \seq_new:N \g__stex_smsmode_allowed_import_env_seq
    2142  \cs_new_protected:Nn \stex_sms_allow_import:Nn {
    2143    \tl_gput_right:Nn \g__stex_smsmode_allowed_import_tl {#1}
    2144    \tl_gset:cn{\tl_to_str:n{#1}~~~smsmode} {#2}
    2145  }
    2146  \cs_new_protected:Nn \stex_sms_allow_import_env:nn {
    2147    \exp_args:NNx \seq_gput_right:Nn \g__stex_smsmode_allowed_import_env_seq {\tl_to_str:n{#1}
    2148      \tl_gset:cn{\tl_to_str:n{#1}~~~env~~~smsmode} {#2}
    2149  }
    2150
    2151  \tl_new:N \g_stex_sms_import_code
```

(End of definition for \stex_sms_allow_import:Nn and \stex_sms_allow_import_env:nn. These functions are documented on page 133.)

```
\stex_file_in_smsmode:nn
\stex_file_in_smsmode:on
  2152  \cs_new_protected:Nn \__stex_smsmode_in_smsmode:n { \stex_suppress_html:n {
  2153    \vbox_set:Nn \l_tmpa_box {
  2154      \bool_set_true:N \g__stex_smsmode_bool
  2155      \bool_set_false:N \stex_html_do_output_bool
  2156      #1
  2157    }
  2158    \%box_clear:N \l_tmpa_box
  2159  } }
  2160
  2161  \quark_new:N \q__stex_smsmode_break
  2162
  2163  \cs_new_protected:Nn \__stex_smsmode_start_smsmode:n {
  2164    \everyeof{\q__stex_smsmode_break\exp_not:N}
  2165    \let\stex_smsmode_do:\__stex_smsmode_smsmode_do:
  2166    \exp_after:wN \exp_after:wN \exp_after:wN
  2167    \stex_smsmode_do:
  2168    \cs:w @ input\cs_end: "#1" \relax
  2169  }
  2170
  2171  \cs_new_protected:Nn \stex_file_in_smsmode:nn {
  2172    \seq_gclear:N \l__stex_smsmode_importmodules_seq
  2173    \seq_gclear:N \l__stex_smsmode_sigmodules_seq
  2174    \tl_clear:N \g_stex_sms_import_code
  2175    \group_begin:
  2176      \let \l_stex_metatheory_uri \c_stex_default_metatheory
  2177      \cs_set:Npn \stex_check_term:n {#1 {}}
  2178      \seq_clear:N \l_stex_all_modules_seq
  2179      \str_clear:N \l_stex_current_module_str
  2180      #2
  2181      \stex_filestack_push:n{#1}
  2182      \__stex_smsmode_in_smsmode:n {
  2183        \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_imports:N
  2184        \tl_map_inline:Nn \g__stex_smsmode_allowed_import_tl {
  2185          \use:c{\tl_to_str:n{##1}~~~smsmode}
```

```

2186     }
2187     \seq_map_inline:Nn \g__stex_smsmode_allowed_import_env_seq {
2188         \use:c{\tl_to_str:n{##1}~env~~~smsmode}
2189     }
2190     \__stex_smsmode_start_smsmode:n{#1}
2191 }
2192 \__stex_smsmode_in_smsmode:n \g_stex_sms_import_code
2193 \__stex_smsmode_in_smsmode:n {
2194     \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_normal:N
2195     \__stex_smsmode_start_smsmode:n{#1}
2196 }
2197 \stex_filestack_pop:
2198 \group_end:
2199 }
2200 \cs_generate_variant:Nn \stex_file_in_smsmode:nn {on}

```

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

`\stex_smsmode_do:`

```

2201 \cs_new_protected:Nn \__stex_smsmode_smsmode_do: {
2202     \% \stex_if_smsmode:T {
2203         \__stex_smsmode_do:w
2204     %}
2205 }
2206 \let\stex_smsmode_do:\relax
2207
2208
2209 \cs_new:Nn \__stex_smsmode_check_cs:NNn {
2210     \exp_after:wN\if\exp_after:wN\relax\exp_not:N#3
2211     \exp_after:wN#1\exp_after:wN#3\else
2212     \exp_after:wN#2\fi
2213 }
2214
2215 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2216     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2217         \__stex_smsmode_check_cs:NNn \__stex_smsmode_do_aux:N \__stex_smsmode_do:w { #1 }
2218     }{
2219         \__stex_smsmode_do:w
2220     }
2221 }
2222
2223 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2224     \cs_if_eq:NNF #1 \q__stex_smsmode_break {
2225         \__stex_smsmode_do_aux_curr:N #1
2226     }
2227 }
2228
2229 \cs_new_protected:Nn \__stex_smsmode_do_aux_imports:N {
2230     \% \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}~in~import}
2231     \tl_if_in:NnTF \g__stex_smsmode_allowed_import_tl {#1} {
2232         \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}~in~import}
2233         #1
2234     }{
2235         \cs_if_eq:NNTF \begin{ #1 {

```

```

2236     \__stex_smsemode_check_begin:Nn \g__stex_smsemode_allowed_import_env_seq
2237     }{
2238         \cs_if_eq:NNTF \end #1 {
2239             \__stex_smsemode_check_end:Nn \g__stex_smsemode_allowed_import_env_seq
2240         }{
2241             \__stex_smsemode_do:w
2242         }
2243     }
2244 }
2245 }
2246
2247 \cs_new_protected:Nn \__stex_smsemode_do_aux_normal:N {
2248     % \stex_debug:nn{sms}{Checking~\tl_to_str:n{#1}-in-sms-mode}
2249     \tl_if_in:NnTF \g__stex_smsemode_allowed_tl {#1} {
2250         \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}}
2251         #1\__stex_smsemode_do:w
2252     }{
2253         \tl_if_in:NnTF \g__stex_smsemode_allowed_escape_tl {#1} {
2254             \stex_debug:nn{sms}{Executing~escaped~\tl_to_str:n{#1}}
2255             #1
2256         }{
2257             \cs_if_eq:NNTF \begin #1 {
2258                 \__stex_smsemode_check_begin:Nn \g__stex_smsemode_allowedenvs_seq
2259             }{
2260                 \cs_if_eq:NNTF \end #1 {
2261                     \__stex_smsemode_check_end:Nn \g__stex_smsemode_allowedenvs_seq
2262                 }{
2263                     \__stex_smsemode_do:w
2264                 }
2265             }
2266         }
2267     }
2268 }
2269
2270 \cs_new_protected:Nn \__stex_smsemode_check_begin:Nn {
2271     % \stex_debug:nn{sms}{Checking~environment~#2}
2272     \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2273         \stex_debug:nn{sms}{Environment~#2}
2274         \begin{#2}
2275     }{
2276         \__stex_smsemode_do:w
2277     }
2278 }
2279 \cs_new_protected:Nn \__stex_smsemode_check_end:Nn {
2280     % \stex_debug:nn{sms}{Checking~end~environment~#2}
2281     \seq_if_in:NxTF #1 { \tl_to_str:n{#2} }{
2282         \stex_debug:nn{sms}{End-Environment~#2}
2283         \end{#2}\__stex_smsemode_do:w
2284     }{
2285         \%str_if_eq:nnTF{#2}{document} \endinput
2286         \__stex_smsemode_do:w
2287     }
2288 }

```

(End of definition for `\stex_smsemode_do`. This function is documented on page 133.)

13.6 Modules

13.6.1 The smodule-environment

2289 `<@@=stex_modules>`

`\l_stex_current_module_str` The current module:

2290 `\str_new:N \l_stex_current_module_str`

(End of definition for `\l_stex_current_module_str`. This variable is documented on page 119.)

`\l_stex_all_modules_seq` Stores all modules currently in scope

2291 `\seq_new:N \l_stex_all_modules_seq`

(End of definition for `\l_stex_all_modules_seq`. This variable is documented on page 119.)

`\stex_every_module:n`

2292 `<@@=stex_module_setup>`

2293 `\tl_clear:N \g_stex_every_module_tl {`

2294 `}`

2295 `\cs_new_protected:Nn \stex_every_module:n {`

2296 `\tl_gput_right:Nn \g_stex_every_module_tl { #1 }`

2297 `}`

(End of definition for `\stex_every_module:n`. This function is documented on page 119.)

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

2298 `\cs_new_protected:Npn \stex_module_setup:n {`

2299 `\stex_if_in_module:TF __stex_module_setup_setup_nested:n __stex_module_setup_setup_top:n`

2300 `}`

2301 `\cs_new_protected:Nn __stex_module_setup_setup_top:n {`

2302 `__stex_module_setup_get_uri_str:n{#1}`

2303 `\stex_debug:nn{module}{Module~URI:~\l__stex_module_setup_ns_str?#1}`

2304 `\str_if_empty:NTF \l_stex_key_sig_str`

2305 `\stex_module_setup_top_nosig:n __stex_module_setup_setup_top_sig:n {\l__stex_module_setu`

2306 `\stex_metagroup_new:o \l_stex_current_module_str`

2307 `\g_stex_every_module_tl`

2308 `\stex_execute_in_module:x {`

2309 `\stex_do_deprecation:n{#1}`

2310 `}`

2311 `__stex_module_setup_load_meta:`

2312 `}`

2313 `}`

2314

2315 `\cs_new_protected:Nn _stex_module_setup_top_nosig:n {`

2316 `\stex_if_module_exists:nTF{#1}{`

2317 `\stex_debug:nn{modules}{{(already exists)}}`

2318 `}{`

2319 `\tl_gclear:c{c_stex_module_ #1 _code}`

2320 `\prop_gclear:c{c_stex_module_ #1 _morphisms_prop }`

2321 `\prop_gclear:c{c_stex_module_ #1 _symbols_prop }`

2322 `\prop_gclear:c{c_stex_module_ #1 _notations_prop }`

2323 `}`

2324 `\str_set:Nx \l_stex_current_module_str {#1}`

2325 `\seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str`

```

2326 }
2327
2328 \cs_new_protected:Nn \__stex_module_setup_top_sig:n {
2329     \stex_if_module_exists:nTF{#1}{
2330         \stex_debug:nn{modules}{(already exists)}
2331     }{
2332         \stex_debug:nn{modules}{(needs loading)}
2333         \__stex_module_setup_load_sig:
2334     }
2335 \%stex_if_smsmode:F { % WHY?
2336     \stex_activate_module:x {
2337         #1
2338     }
2339 }
2340 \str_set:Nx\l_stex_current_module_str{#1}
2341 }
2342
2343 \cs_new_protected:Nn \__stex_module_setup_load_sig: {
2344     \stex_file_split_off_ext:NN \l__stex_module_setup_sigfile \g_stex_current_file
2345     \stex_file_split_off_lang:NN \l__stex_module_setup_sigfile \l__stex_module_setup_sigfile
2346     \exp_args:Ne \stex_file_in_smsmode:nn {
2347         \stex_file_use:N \l__stex_module_setup_sigfile . \l_stex_key_sig_str . tex
2348     }{}
2349 }
2350
2351 \cs_new_protected:Nn \__stex_module_setup_setup_nested:n {
2352     \exp_after:wN
2353         \__stex_module_setup_split_module:n \l_stex_current_module_str \__stex_module_setup_end:
2354     \stex_debug:nn{module}{Nested~Module~URI:~\l_stex_current_module_str}
2355     \seq_put_right:No \l_stex_all_modules_seq \l_stex_current_module_str
2356     \stex_metagroup_new:o \l_stex_current_module_str
2357 }
2358
2359
2360 \cs_new_protected:Nn \__stex_module_setup_get_uri_str:n {
2361     \str_clear:N \l__stex_module_setup_ns_str
2362     \stex_map_uri:Nnnnn \l_stex_current_ns_uri {
2363         \str_set:Nx \l__stex_module_setup_ns_str{\##1\c_colon_str/}
2364     }{
2365         \seq_set_split:Nnn \l__stex_module_setup_seq / {\##1}
2366         \seq_pop_right:NN \l__stex_module_setup_seq \l__stex_module_setup_seg
2367         \exp_args:No \str_if_eq:nnF \l__stex_module_setup_seg {#1} {
2368             \seq_put_right:No \l__stex_module_setup_seq \l__stex_module_setup_seg
2369         }
2370         \tl_put_right:Nx \l__stex_module_setup_ns_str {\seq_use:Nn \l__stex_module_setup_seq /}
2371     }{}{}
2372 }
2373
2374 \cs_new_protected:Npn \__stex_module_setup_split_module:n #1?#2 \__stex_module_setup_end: #3
2375     \stex_module_setup_top_nosig:n { #1 ? #2 / #3}
2376 }
2377
2378 \bool_new:N \l_stex_in_meta_bool
2379 \bool_set_false:N \l_stex_in_meta_bool

```

```

2380 \cs_new_protected:Nn \__stex_module_setup_load_meta: {
2381   \tl_if_empty:NF \l_stex_metatheory_uri {
2382     \stex_execute_in_module:x{
2383       \stex_pseudogroup_with:nn{\l_stex_in_meta_bool} {
2384         \stex_activate_module:n {\l_stex_uri_use:N \l_stex_metatheory_uri }
2385       }
2386     }
2387   }
2388 }
2389 }
2390
2391 (@@=stex_modules)

```

(End of definition for `\stex_module_setup:n`. This function is documented on page 119.)

`\stex_close_module:`

```

2392 \cs_new:Nn \stex_close_module: {
2393   \bool_if:NT \c_stex_persist_write_mode_bool \__stex_modules_persist_module:
2394   \stex_debug:nn{module} {
2395     Closing~module~\l_stex_current_module_str^~J
2396     Code:~\expandafter\meaning\csname c_stex_module_\l_stex_current_module_str _code\endcsna
2397     Imports:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_str
2398     Declarations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_
2399     Notations:\exp_after:wN \prop_to_keyval:N \cs:w c_stex_module_\l_stex_current_module_
2400   }
2401 }
2402
2403 \cs_new_protected:Nn \__stex_modules_persist_module: {
2404   \stex_persist:e {
2405     \__stex_modules_restore_module:nnnn {\l_stex_current_module_str} {
2406       \exp_after:wN \prop_to_keyval:N \cs:w
2407         c_stex_module_\l_stex_current_module_str _morphisms_prop
2408       \cs_end:
2409     }{
2410       \exp_after:wN \prop_to_keyval:N \cs:w
2411         c_stex_module_\l_stex_current_module_str _symbols_prop
2412       \cs_end:
2413     }{
2414       \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
2415       \exp_after:wN \exp_after:wN \exp_after:wN
2416       { \cs:w c_stex_module_\l_stex_current_module_str _code \cs_end: }
2417     }{}
2418     \prop_map_function:cN{c_stex_module_\l_stex_current_module_str _notations_prop}
2419       \__stex_modules_persist_nots_i:nn
2420     \exp_not:N \STEXRestoreNotsEnd {}
2421   }
2422 }
2423
2424 \cs_new_protected:Nn \__stex_modules_restore_module:nnnn {
2425   \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_morphisms_prop}{#2}
2426   \cs_set:Npn \__stex_modules_tl {#3}
2427   \exp_args:Nno \prop_gset_from_keyval:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}\__stex_
2428   \prop_map_inline:cn{c_stex_module_\tl_to_str:n{#1}_symbols_prop}{%
2429     \stex_ref_new_symbol:n{#1?##1}

```

```

2430 }
2431 \cs_gset:cpn{c_stex_module_ \tl_to_str:n{#1}_code}{#4}
2432 \prop_gclear:c{c_stex_module_ \tl_to_str:n{#1} _notations_prop}
2433 \str_set:Nn \l_stex_modules_restore_mod_str {#1}
2434 \group_begin:
2435   \catcode`_=8\relax
2436   \catcode`:=12\relax
2437   \__stex_modules_restore_nots:n
2438 }
2439
2440 \cs_new:Nn \__stex_modules_persist_nots_i:nn {
2441   \exp_not:n{#2}
2442 }
2443
2444 \quark_new:N \STEXRestoreNotsEnd
2445
2446 \cs_new_protected:Nn \__stex_modules_restore_nots:n {
2447   \__stex_modules_restore_nots_i:n
2448 }
2449
2450 \cs_new_protected:Nn \__stex_modules_restore_nots_i:n {
2451   \tl_if_eq:nnTF{#1}{\STEXRestoreNotsEnd}{
2452     \group_end:
2453   }{
2454     \__stex_modules_restore_nots_ii:nnnn {#1}
2455   }
2456 }
2457
2458 \cs_new_protected:Nn \__stex_modules_restore_nots_ii:nnnn {
2459   \cs_set:Npn \l_stex_modules_tl {{#4}{#5}}
2460   \exp_args:NNe\use:nn\prop_gput:cnn{
2461     {c_stex_module_ \l_stex_modules_restore_mod_str _notations_prop}
2462     {\tl_to_str:n{#1!#2}}{
2463       {\tl_to_str:n{#1}}{\tl_to_str:n{#2}}{#3}
2464       \exp_args:No \exp_not:n \l_stex_modules_tl
2465     }
2466   }
2467   \__stex_modules_restore_nots_i:n
2468 }

```

(End of definition for `\stex_close_module`. This function is documented on page [119](#).)

`\l_stex_metatheory_uri`

```
2469 \tl_new:N \l_stex_metatheory_uri
```

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

`\setmetatheory`

```

2470 \cs_new_protected:Nn \__stex_modules_set_matatheory:nn {
2471   \group_begin:
2472     \stex_debug:nn{metatheory}{Setting-metatheory~[#1]#2}
2473     \stex_import_module_uri:nn { #1 } { #2 }
2474     \stex_debug:nn{metatheory}{Here:^^J
2475       \l_stex_import_archive_str^^J
2476       \l_stex_import_path_str^^J

```

```

2477     \l_stex_import_name_str^^J
2478   }
2479   \stex_import_require_module:ooo
2480     \l_stex_import_archive_str
2481     \l_stex_import_path_str
2482     \l_stex_import_name_str
2483 \stex_debug:nn{metatheory}{Found:~\l_stex_import_ns_str}
2484 \exp_args:Nne \use:nn {
2485   \group_end: \stex_uri_resolve:Nn \l_stex_metatheory_uri
2486 }{\l_stex_import_ns_str}
2487 }
2488
2489 \NewDocumentCommand \setmetatheory {O{} m}{
2490   \__stex_modules_set_metatheory:nn {#1}{#2}
2491   \stex_smsmode_do:
2492 }
2493 \stex_sms_allow_escape:N \setmetatheory

```

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

Keys and key handling:

```

2494 \stex_keys_define:nnnn{smodule}{
2495   \str_clear:N \l_stex_key_sig_str
2496 }{
2497   meta .code:n = {
2498     \str_if_empty:nTF {#1} {
2499       \tl_clear:N \l_stex_metatheory_uri
2500     }{
2501       \stex_uri_resolve:Nx \l_stex_metatheory_uri {#1}
2502     }
2503   },
2504   ns .code:n = {
2505     \stex_uri_resolve:Nx \l_stex_current_ns_uri {#1}
2506   },
2507   lang .code:n = {
2508     \stex_set_language:n {#1}
2509   },
2510   sig .str_set_x:N = \l_stex_key_sig_str ,
2511   creators .code:n = {} , % todo ?
2512   contributors .code:n = {} , % todo ?
2513   srccite .code:n = {} % todo ?
2514 }{id, title, style, deprecate}

```

smodule (env.)

```

2515 \stex_new_stylable_env:nnnnnnn {module} {O{} m} {
2516   \stex_keys_set:nn { smodule }{ #1 }
2517   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
2518   \tl_if_empty:NF \thistitle {
2519     \exp_args:No \stexdoctitle \thistitle
2520   }
2521   \exp_args:Nx \stex_module_setup:n { \tl_to_str:n{ #2 } }
2522
2523 \stex_if_do_html:T {
2524   \exp_args:Nne \begin{stex_annotation_env} {
2525     shtml:theory={\l_stex_current_module_str},

```

```

2526     shtml:language={ \l_stex_current_language_str},
2527     shtml:signature={\l_stex_key_sig_str}
2528     \tl_if_empty:NF \l_stex_metatheory_uri {
2529         shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
2530     }
2531 }
2532 \stex_annotation_invisible:n{}
2533 }
2534 \stex_if_smsmode:F {
2535     \str_set_eq:NN \thismoduleuri \l_stex_current_module_str
2536     \tl_set:Nn \thismodulename {#2}
2537     \stex_style_apply:
2538 }
2539 \stex_smsmode_do:
2540 }{
2541     \stex_close_module:
2542     \stex_if_smsmode:F \stex_style_apply:
2543     \stex_if_do_html:T{ \end{stex_annotation_env} }
2544 }{}{}{s}
2545
2546 \stex_sms_allow_env:n{smodule}

```

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

\stex_if_in_module:TF

```

2547 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
2548     \str_if_empty:NTF \l_stex_current_module_str
2549     \prg_return_false: \prg_return_true:
2550 }

```

(End of definition for `\stex_if_in_module:TF`. This function is documented on page 119.)

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

\stex_if_module_exists:nTF

```

2551 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
2552     \tl_if_exist:cTF { c_stex_module_#1_code }
2553     \prg_return_true: \prg_return_false:
2554 }

```

(End of definition for `\stex_if_module_exists:nTF`. This function is documented on page 119.)

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

```

2555 \cs_new_protected:Nn \stex_do_up_to_module:n {
2556     \exp_args:No \stex_metagroup_do_in:nn \l_stex_current_module_str {#1}
2557 }
2558 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}

```

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

\stex_module_add_code:n

\stex_module_add_code:x

```

2559 \cs_new_protected:Nn \stex_module_add_code:n {
2560     \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
2561 }
2562 \cs_generate_variant:Nn \stex_module_add_code:n {x}

```

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

```

\stex_execute_in_module:n
\stex_execute_in_module:x 2563 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:TF {
2564     \stex_module_add_code:n { #1 }
2565     \stex_do_up_to_module:n { #1 }
2566 }{ #1 }
2567 \cs_generate_variant:Nn \stex_execute_in_module:n {x}

```

(End of definition for `\stex_execute_in_module:n`. This function is documented on page 119.)

\STEXexport

```

2568 \NewDocumentCommand \STEXexport {} {
2569     \ExplSyntaxOn
2570     \__stex_modules_export:n
2571 }
2572 \cs_new_protected:Nn \__stex_modules_export:n {
2573     \stex_ignore_spaces_and_pars:#1\ExplSyntaxOff
2574     \stex_module_add_code:n { \stex_ignore_spaces_and_pars:#1}
2575     \stex_smsmode_do:
2576 }

```

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

```

2577 \stex_deactivate_macro:Nn \STEXexport {module~environments}
2578 \stex_sms_allow_escape:N \STEXexport
2579 \stex_every_module:n {\stex_reactivate_macro:N \STEXexport}

```

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

\stex_module_add_morphism:nnnn

```

\stex_module_add_morphism:nonn
\stex_module_add_morphism:ooox
2580 \cs_new_protected:Nn \stex_module_add_morphism:nnnn {
2581     \exp_args:Nne \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _morphisms_prop}{%
2582         \tl_if_empty:nTF{#1}{[#2]}{#1}
2583     }{[#1]{#2}{#3}{#4}}
2584 }
2585 \cs_generate_variant:Nn \stex_module_add_morphism:nnnn {nonn,ooox}

```

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

\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

2586 \cs_new_protected:Nn \stex_module_add_symbol:nnnnnnN {
2587     \stex_debug:nn{declaration}{New~declaration:~\l_stex_current_module_str?#2^J
2588         Macro:#1^JArity:#3~(#4)^J
2589         Def:~\tl_to_str:n{#5}^J
2590         Type:~\tl_to_str:n{#6}^J
2591         Returns:~\tl_to_str:n{#7}
2592     }
2593     \%prop_gput:cnx{c_stex_module_\l_stex_current_module_str _symbols_prop}
2594     %{#2}{\exp_not:n{#1}{#2}{#3}{#4}{#5}{#6}{#7}\exp_not:n{#8}}

```

```

2595 \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
2596 {##2}{##1}{##2}{##3}{##4}{##5}{##6}{##7}{##8}}
2597 \tl_if_empty:nF{#1} {
2598   \stex_execute_in_module:n {
2599     \__stex_modules_activate_sym:n {#2}
2600   }
2601 }
2602 }
2603
2604 \cs_new_protected:Nn \__stex_modules_activate_sym:n {
2605   \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _symbols_prop}{%
2606     \str_if_eq:nnT{#1}{##1}{%
2607       \__stex_modules_activate_i:nnnnnnnn ##2
2608     }
2609   }
2610 }
2611 \cs_new_protected:Nn \__stex_modules_activate_i:nnnnnnnn {
2612   \stex_debug:nn{activating}{#1:\l_stex_current_module_str^~J}
2613   \tl_to_str:n{##2}{##3}{##4}{##5}{##6}{##7}{##8}
2614 }
2615 \cs_set:cp{#1} {
2616   \stex_invoke_symbol:nnnnnnnN
2617   {\l_stex_current_module_str}
2618   \exp_not:n{##2}{##3}{##4}{##5}{##6}{##7}{##8}
2619 }
2620 \stex_debug:nn{activating}{done}
2621 \prop_map_break:
2622 }

```

(End of definition for `\stex_module_add_symbol: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

2623 \cs_new_protected:Nn \stex_module_add_notation:nnnnn {
2624   \stex_debug:nn{notations}{Adding~notation:~^~J}
2625   #1~\c_hash_str#2~#3~^~J
2626   to~\l_stex_current_module_str
2627 }
2628 \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _notations_prop}
2629 {##1!##2}{##1}{##2}{##3}{##4}{##5}
2630 \stex_execute_in_module:n {
2631   \__stex_modules_activate_not:nn{##1}{##2}
2632 }
2633 }
2634 \cs_generate_variant:Nn \stex_module_add_notation:nnnnn {eoeoo}
2635
2636
2637 \cs_new_protected:Nn \__stex_modules_activate_not:nn {
2638   \prop_map_inline:cn{c_stex_module_\l_stex_current_module_str _notations_prop}{%
2639     \str_if_eq:nnT{##1!##2}{##1}{%
2640       \prop_map_break:n{\stex_set_notation_macro:nnnnn ##2 } }}
```

```

2641     }
2642   }
2643 }
```

(End of definition for \stex_module_add_notation:nnnnn and \stex_set_notation_macro:nnnnn. These functions are documented on page 122.)

```
\stex_set_notation_macro:nnnnn
\stex_set_notation_macro:eoexo
2644 \cs_new_protected:Nn \stex_set_notation_macro:nnnnn {
2645   \tl_set:cn {l_stex_notation_#1#2_cs}{#4}
2646   \cs_if_exist:cF{l_stex_notation_#1__cs} {
2647     \tl_set:cn {l_stex_notation_#1__cs}{#4}
2648   }
2649   \tl_if_empty:nF{#5} {
2650     \tl_set:cn{l_stex_notation_#1_op_#2_cs}{#5}
2651     \cs_if_exist:cF{l_stex_notation_#1_op__cs} {
2652       \cs_set_eq:cc {l_stex_notation_#1_op__cs}{l_stex_notation_#1_op_#2_cs}
2653     }
2654   }
2655 }
2656 \cs_generate_variant:Nn \stex_set_notation_macro:nnnnn {eoexo}
```

(End of definition for \stex_set_notation_macro:nnnnn. This function is documented on page 123.)

```
\stex_activate_module:n
\stex_activate_module:o
\stex_activate_module:x
2657 \cs_new_protected:Nn \stex_activate_module:n {
2658   \seq_if_in:Nnf \l_stex_all_modules_seq { #1 } {
2659     \stex_debug:nnf{modules}{Activating-module~#1^J\expandafter\meaning\csname c_stex_module
2660     \seq_put_right:Nn \l_stex_all_modules_seq { #1 }
2661     \stex_pseudogroup:nn{
2662       \str_set:Nn \l_stex_current_module_str {#1}
2663       \use:c{ c_stex_module_#1_code }
2664     }{
2665       \stex_pseudogroup_restore:N \l_stex_current_module_str
2666     }
2667   }
2668 }
2669 \cs_generate_variant:Nn \stex_activate_module:n {o,x}
```

(End of definition for \stex_activate_module:n. This function is documented on page 119.)

Iterating:

```
2670 <@=stex_iterate>
```

\stex_iterate_symbols:n

```
2671 \cs_new_protected:Nn \stex_iterate_symbols:n {
2672   \stex_pseudogroup_with:nn{\_\stex_iterate_sym cs:nnnnnnnnN\stex_iterate_break:\stex_iterat
2673   \cs_set:Npn \_\stex_iterate_sym cs:nnnnnnnnN
2674   ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
2675   \cs_set:Npn \stex_iterate_break: {
2676     \prop_map_break:n{\seq_map_break:}
2677   }
2678   \cs_set:Npn \stex_iterate_break:n ##1 {
2679     \prop_map_break:n{\seq_map_break:n{##1}}
2680   }
```

```

2681   \seq_map_inline:Nn \l_stex_all_modules_seq {
2682     \prop_map_inline:cn{c_stex_module_##1_symbols_prop}{
2683       \__stex_iterate_sym_cs:nnnnnnnnN {##1} ####2
2684     }
2685   }
2686 }
2687 }
```

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

\stex_iterate_symbols:nn

```

2688 \cs_new_protected:Nn \stex_iterate_symbols:nn {
2689   \seq_clear:N \l_stex_iterate_mods_seq
2700   \stex_pseudogroup_with:nn{\l_stex_iterate_sym_cs:nnnnnnnnN}{
2701     \cs_set:Npn \__stex_iterate_sym_cs:nnnnnnnnN
2702       ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #2 }
2703     \clist_map_function:nN {#1} \__stex_iterate_it_decl_i:n
2704   }
2705 }
```

2696

```

2697 \cs_new_protected:Nn \__stex_iterate_it_decl_i:n {
2698   \seq_if_in:NnF \l_stex_iterate_mods_seq {#1} {
2699     \seq_put_left:Nn \l_stex_iterate_mods_seq {#1}
2700     \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2701       \__stex_iterate_it_decl_check:nnnn ##2
2702     }
2703     \prop_map_inline:cn{c_stex_module_#1_symbols_prop}{
2704       \__stex_iterate_sym_cs:nnnnnnnnN {##1} ##2
2705     }
2706   }
2707 }
```

2708

```

2708 \cs_new_protected:Nn \__stex_iterate_it_decl_check:nnnn {
2709   \tl_if_empty:nT{#1}{%
2710     \__stex_iterate_it_decl_i:n {#2}
2711   }
2712 }
```

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

\stex_iterate_notations:nn

```

2713 \cs_new_protected:Nn \stex_iterate_notations:nn {
2714   \seq_clear:N \l_stex_iterate_mods_seq
2715   \stex_pseudogroup_with:nn{\l_stex_iterate_not_cs:nnnn}{%
2716     \cs_set:Npn \__stex_iterate_not_cs:nnnn
2717       ##1 ##2 ##3 ##4 ##5 { #2 }
2718     \clist_map_function:nN {#1} \__stex_iterate_it_not_i:n
2719   }
2720 }
```

2721

```

2722 \cs_new_protected:Nn \__stex_iterate_it_not_i:n {
2723   \seq_if_in:NnF \l_stex_iterate_mods_seq {#1} {
2724     \seq_put_left:Nn \l_stex_iterate_mods_seq {#1}
2725     \prop_map_inline:cn{c_stex_module_#1_notations_prop}{
2726       \__stex_iterate_not_cs:nnnn ##2
2727     }
2728 }
```

```

2728   \prop_map_inline:cn{c_stex_module_#1_morphisms_prop}{
2729     \__stex_iterate_it_not_check:nnnn ##2
2730   }
2731 }
2732 }
2733 \cs_new_protected:Nn \__stex_iterate_it_not_check:nnnn {
2734   \tl_if_empty:nT{#1}{%
2735     \__stex_iterate_it_not_i:n {#2}
2736   }
2737 }
```

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

`\stex_iterate_morphisms:nn`

```

2738 \cs_new_protected:Nn \stex_iterate_morphisms:nn {
2739   \seq_clear:N \l__stex_iterate_mods_seq
2740   \bool_set_true:N \l__stex_iterate_continue_bool
2741   \cs_set:Npn \__stex_iterate_morphism_cs:nnnn ##1 ##2 ##3 ##4 ##5 {
2742     #2
2743     \bool_if:NT \l__stex_iterate_continue_bool {
2744       \str_if_eq:nnTF{##1}{##2}{%
2745         \tl_put_right:Nn \l__stex_iterate_todo_tl {
2746           \__stex_iterate_iterate_morphism:nn{##5}{##2}
2747         }
2748       }{
2749         \tl_put_right:Nn \l__stex_iterate_todo_tl {
2750           \__stex_iterate_iterate_morphism:nn{##5 / ##1}{##2}
2751         }
2752       }
2753     }
2754   }
2755   \cs_set:Npn \stex_iterate_break:n ##1 {
2756     \bool_set_false:N \l__stex_iterate_continue_bool
2757     \prop_map_break:n{##1}
2758   }
2759   \__stex_iterate_iterate_morphism:nn{}{##1}
2760 }
```

```

2761
2762 \cs_new_protected:Nn \__stex_iterate_iterate_morphism:nn {
2763   \tl_clear:N \l__stex_iterate_todo_tl
2764   \seq_if_in:NnF \l__stex_iterate_mods_seq {##1 ##2}{%
2765     \seq_put_right:Nn \l__stex_iterate_mods_seq {##1 ##2}
2766     \prop_map_inline:cn{c_stex_module_#2_morphisms_prop}{
2767       \__stex_iterate_morphism_cs:nnnn ##2 {##1}
2768       % TODO
2769       % ##1: name or [mpath]
2770       % ##2 = {##1}{##2}{##3}{##4}
2771       % ##1 = name
2772       % ##2 = mpath
2773       % ##3 = type
2774       % ##4 = {origname}{newname}*
2775     }
2776     \bool_if:NT \l__stex_iterate_continue_bool \l__stex_iterate_todo_tl
2777   }
2778 }
```

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

13.6.2 Structural Features

```
2779 <@@=stex_features>

\stex_structural_feature_module:nn
\stex_structural_feature_module_end:
2780 \cs_new_protected:Nn \stex_structural_feature_module:nn {
2781   \stex_if_do_html:TF {
2782     \exp_args:Nne \begin{stex_annotation_env} {
2783       \shtml:feature-#2={
2784         \l_stex_current_module_str/#1}
2785       \str_if_empty:NF \l_stex_mroname_str {
2786         \shtml:macroname={\l_stex_mroname_str}
2787       }
2788     }
2789     \stex_annotation_invisible:n{}
2790   }\group_begin:
2791   \stex_module_setup:n {#1-module}
2792 }
2793
2794 \cs_new_protected:Nn \stex_structural_feature_module_end: {
2795   \tl_gset_eq:NN \g_stex_last_feature_str \l_stex_current_module_str
2796   \stex_close_module:
2797   \stex_if_do_html:TF{
2798     \end{stex_annotation_env}
2799   }\group_end:
2800 }
```

(End of definition for \stex_structural_feature_module:nn and \stex_structural_feature_module_end:. These functions are documented on page 125.)

```
\stex_structural_feature_morphism:nnn
\stex_structural_feature_morphism_end:
2801 \bool_new:N \l__stex_features_implicit_bool
2802 \cs_new_protected:Nn \stex_structural_feature_morphism:nnnnn {
2803   \str_clear:N \l_stex_current_domain_str
2804   \tl_if_empty:nT{#3} {
2805     \stex_get_mathstructure:n{#4}
2806     \str_set_eq:NN \l_stex_current_domain_str \l_stex_get_structure_module_str
2807   }
2808   \str_if_empty:NT \l_stex_current_domain_str {
2809     \stex_import_module_uri:nn { #3 }{ #4 }
2810     \group_begin:
2811     \stex_import_require_module:ooo
2812       \l_stex_import_archive_str
2813       \l_stex_import_path_str
2814       \l_stex_import_name_str
2815     \exp_args:Nnx \use:nn \group_end: {
2816       \str_set:Nn \exp_not:N\l_stex_current_domain_str {\l_stex_import_ns_str}
2817     }
2818   }
2819   \tl_if_empty:nTF{#1} {
2820     \bool_set_true:N \l__stex_features_implicit_bool
2821     \str_set:Nx \l_tmpa_str {[ \l_stex_current_domain_str ] }
2822   }{
```

```

2823   \bool_set_false:N \l__stex_features_implicit_bool
2824   \str_set:Nn \l_tmpa_str {#1}
2825 }
2826
2827 \stex_if_do_html:TF {
2828   \begin{stex_annotation_env} {
2829     \shtml:feature-#2={\l_stex_current_module_str?\l_tmpa_str},
2830     \shtml:domain={\l_stex_current_domain_str}
2831     #5
2832   }
2833   \stex_annotation_invisible:n{}
2834 } \group_begin:
2835 \str_set:Nn \l__stex_features_feature_str {#2}
2836 \str_set_eq:NN \l_stex_feature_name_str \l_tmpa_str
2837 \__stex_features_setup:
2838 \__stex_features_reactivate:
2839 %^A\stex_activate_module:o \l_stex_current_domain_str
2840 \exp_args:Ne \stex_metagroup_new:n {\l_stex_current_module_str / \l_stex_feature_name_str}
2841 }
2842
2843 \cs_new_protected:Nn \__stex_features_do_for_list: {
2844   \seq_clear:N \l_stex_fors_seq
2845   \clist_map_inline:Nn \l_stex_key_for_clist {
2846     \exp_args:Ne \stex_get_in_morphism:n{\tl_to_str:n{##1}}
2847     \seq_put_right:Nx \l_stex_fors_seq
2848     {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
2849   }
2850 }
2851
2852 \cs_new_protected:Nn \__stex_features_add_definiens:nn {
2853   \__stex_features_set_definiens_macros: #1\__stex_features_break:
2854   \stex_assign_do:n{#2}
2855   #2
2856 }
2857 \cs_new_protected:Npn \__stex_features_set_definiens_macros: #1?#2?#3\__stex_features_break:
2858   \str_set:Nn \l_stex_get_symbol_mod_str {#1?#2}
2859   \str_set:Nn \l_stex_get_symbol_name_str {#3}
2860   \exp_args:Nne \use:nn{\__stex_features_set_definiens_macros_i:nnnnnnn}{%
2861     \prop_item:Nn \l_stex_morphism_symbols_prop {[#1?#2]/[#3]}}
2862   }
2863 }
2864 \cs_new_protected:Nn \__stex_features_set_definiens_macros_i:nnnnnnn {
2865   \tl_set:Nn \l_stex_get_symbol_def_tl{#4}
2866 }
2867
2868 \cs_new_protected:Nn \stex_structural_feature_morphism_end: {
2869   \str_gset_eq:NN \l_stex_feature_name_str \l_stex_feature_name_str
2870   \str_gset_eq:NN \l_stex_current_domain_str \l_stex_current_domain_str
2871   \seq_gset_eq:NN \l_stex_morphism_symbols_prop \l_stex_morphism_symbols_prop
2872   \seq_gset_eq:NN \l_stex_morphism_renames_prop \l_stex_morphism_renames_prop
2873   \seq_gset_eq:NN \l_stex_morphism_morphisms_seq \l_stex_morphism_morphisms_seq
2874   \__stex_features_do_elaboration:
2875   \stex_if_do_html:TF{
2876     \end{stex_annotation_env}

```

```

2877   }\group_end:
2878 }
2879
2880 \cs_new_protected:Nn \__stex_features_setup: {
2881   \prop_clear:N \l_stex_morphism_symbols_prop
2882   \prop_clear:N \l_stex_morphism_renames_prop
2883   \seq_clear:N \l_stex_morphism_morphisms_seq
2884   \__stex_features_do_decls:
2885   \exp_args:No \__stex_features_do_morphisms:n \l_stex_current_domain_str
2886 }
2887
2888 \cs_new_protected:Nn \__stex_features_rename_all: {
2889 }
2890
2891 \cs_new:Npn \__stex_features_clean:nnw [#1] / #2 \_stex_end: {
2892   [#1]/[#2]
2893 }
2894
2895 \cs_new_protected:Nn \__stex_features_do_decls: {
2896   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_domain_str {
2897     \stex_if_starts_with:nnTF{##3}[{
2898       \exp_args:NNe \prop_put:Nnn \l_stex_morphism_symbols_prop {
2899         \__stex_features_clean:nnw ##3 \_stex_end:
2900       }
2901     }{
2902       \prop_put:Nnn \l_stex_morphism_symbols_prop
2903         {[##1]/[##3]}
2904     }{
2905       {[##2}{##4}{##5}{##6}{##7}{##8}##9
2906     }
2907   }
2908 }
2909
2910 \cs_new_protected:Nn \stex_structural_feature_morphism_check_total: {
2911   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2912     \__stex_features_total_check: ##1 ##2
2913   }
2914 }
2915
2916 \cs_new_protected:Npn \__stex_features_total_check: [#1]/[#2] #3 #4 #5 #6 #7 #8 #9 {
2917   \tl_if_empty:nT{#6}{%
2918     \msg_error:nnxx{\stex}{error/needsdefiniens}{##1##2}{total-morphism}
2919   }
2920 }
2921
2922 \cs_new:Npn \__stex_features_split_qm:w #1 ? #2 ? #3 { #3 }
2923 \cs_new_protected:Nn \__stex_features_do_elaboration: {
2924   \stex_debug:nn{morphisms}{%
2925     Elaborating:^^J\prop_to_keyval:N \l_stex_morphism_symbols_prop
2926     ^^J
2927     Renamings:^^J
2928     \prop_to_keyval:N \l_stex_morphism_renames_prop
2929   }
2930 }
```

```

2931 \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
2932   \__stex_features_elab_check: ##1 ##2
2933 }
2934 \exp_args:No\stex_iterate_notations:nn\l_stex_current_domain_str{
2935   \prop_get:NnNTF \l_stex_morphism_renames_prop {##1}\l__stex_features_tmp {
2936     \exp_args:Ne \stex_module_add_notation:nnnnn
2937     {\l_stex_current_module_str ? \exp_after:wN \use_i:nn \l__stex_features_tmp}
2938   }{
2939     \exp_args:Ne \stex_module_add_notation:nnnnn
2940     {\l_stex_current_module_str ? \l_stex_feature_name_str
2941       / \__stex_features_split_qm:w ##1}
2942     }{##2}{##3}{##4}{##5}
2943   }
2944 \stex_module_add_morphism:ooox
2945   \l_stex_feature_name_str
2946   \l_stex_current_domain_str
2947   \l_stex_features_feature_str
2948   {\prop_map_function:NN \l_stex_morphism_renames_prop \__stex_features_rename:nn}
2949 }
2950
2951 \cs_new:Nn \__stex_features_rename:nn{
2952   {#1}{\use_i:nn#2}
2953 }
2954
2955 \cs_new_protected:Npn \__stex_features_elab_check: [#1]/[#2] #3 {
2956   \prop_get:NnNTF \l_stex_morphism_renames_prop {#1##2} \l__stex_features_tmp {
2957     \stex_debug:nn{morphisms}{Generating~\l_stex_features_tmp}
2958     \exp_after:wN \stex_module_add_symbol:nnnnnnnnN \l_stex_features_tmp
2959   }{
2960     \bool_if:NTF \l_stex_features_implicit_bool {
2961       \stex_debug:nn{morphisms}{Generating~#3:~\l_stex_feature_name_str / #2}
2962       \exp_args:Nno \stex_module_add_symbol:nnnnnnnnN {#3}{\l_stex_feature_name_str / #2}
2963     }{
2964       \stex_debug:nn{morphisms}{Generating~\l_stex_feature_name_str / #2}
2965       \exp_args:Nno \stex_module_add_symbol:nnnnnnnnN {}{\l_stex_feature_name_str / #2}
2966     }
2967   }
2968 }
2969
2970 \cs_new_protected:Nn \__stex_features_do_morphisms:n {
2971   \prop_map_inline:cn {c_stex_module_#1_morphisms_prop} {
2972     \__stex_features_do_morph:nnnn ##2
2973   }
2974 }
2975
2976 \cs_new_protected:Nn \__stex_features_do_morph:nnnn {
2977   \tl_if_empty:nF{#3} {
2978     \seq_put_right:Nn \l_stex_morphism_morphisms_seq {{#1}{#2}{#3}}
2979   }
2980   \__stex_features_do_morphisms:n{#2}
2981 }
2982
2983 \cs_new_protected:Npn \__stex_features_reactivate: {
2984   \stex_deactivate_macro:Nn \symdecl {module~environments}

```

```

2985 \stex_deactivate_macro:Nn \textsymdecl {module-environments}
2986 \stex_deactivate_macro:Nn \symdef {module-environments}
2987 \stex_deactivate_macro:Nn \notation {module-environments}
2988 \stex_deactivate_macro:Nn \importmodule {module-environments}
2989 \stex_deactivate_macro:Nn \requiremodule {module-environments}
2990 \stex_deactivate_macro:Nn \smodule {outside-of-morphisms}
2991 \stex_reactivate_macro:N \assign
2992 \stex_reactivate_macro:N \assignMorphism
2993 \stex_reactivate_macro:N \renamedecl
2994 \cs_set_eq:NN \stex_do_for_list: \stex_features_do_for_list:
2995 \cs_set_eq:NN \stex_add_definiens:nn \stex_features_add_definiens:nn
2996 }

```

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

\stex_get_in_morphism:n

```

2997 \cs_new_protected:Nn \stex_get_in_morphism:n {
2998   \str_clear:N \l_stex_get_symbol_name_str
2999   \prop_map_inline:Nn \l_stex_morphism_symbols_prop {
3000     \exp_args:Nx \stex_features_get_check:nnnn{\tl_to_str:n{#1}}##1##2
3001   }
3002   \str_if_empty:NT \l_stex_get_symbol_name_str {
3003     \prop_map_inline:Nn \l_stex_morphism_renames_prop {
3004       \stex_features_renamed_check:nnnnn{#1}##1=##2
3005     }
3006     \str_if_empty:NT \l_stex_get_symbol_name_str {
3007       \msg_error:nnxx{\stex}{error/unknownsymbolin}{#1}{
3008         morphism-\l_stex_feature_name_str
3009       }
3010     }
3011   }
3012 }
3013
3014 \cs_new_protected:Npn \stex_features_renamed_check:nnnnn #1##2##3##4##5##6 {
3015   \str_if_eq:nnTF{#1}{#5} {
3016     \exp_args:Nnx \use:nn{\stex_features_check_break:nnnnnnnn{#2##3}{#4}}{
3017       \prop_item:Nn \l_stex_morphism_symbols_prop {[#2##3]/[#4]}
3018     }
3019   }{
3020     \str_if_eq:nnT{#1}{#6} {
3021       \exp_args:Nnx \use:nn{\stex_features_check_break:nnnnnnnn{#2##3}{#4}}{
3022         \prop_item:Nn \l_stex_morphism_symbols_prop {[#2##3]/[#4]}
3023       }
3024     }
3025   }
3026 }
3027
3028 \cs_new_protected:Npn \stex_features_get_check:nnnn #1[#2]/[#3]##4 {
3029   \str_if_eq:nnTF{#1}{#3} {
3030     \stex_features_check_break:nnnnnnnn{#2}{#3}{#4}
3031   }{
3032     \str_if_eq:nnTF{#1}{#4} {
3033       \stex_features_check_break:nnnnnnnn{#2}{#3}{#4}

```

```

3034     }{
3035         \use_none:nnnnnnn
3036     }
3037 }
3038 }
3039
3040 \cs_new_protected:Nn \__stex_features_check_break:nnnnnnnn {
3041     \prop_map_break:n{
3042         \str_set:Nn \l_stex_get_symbol_mod_str{\#1}
3043         \str_set:Nn \l_stex_get_symbol_name_str{\#2}
3044         \str_set:Nn \l_stex_get_symbol_macro_str{\#3}
3045         \int_set:Nn \l_stex_get_symbol_arity_int {\#4}
3046         \tl_set:Nn \l_stex_get_symbol_args_tl {\#5}
3047         \tl_set:Nn \l_stex_get_symbol_def_tl {\#6}
3048         \tl_set:Nn \l_stex_get_symbol_type_tl {\#7}
3049         \tl_set:Nn \l_stex_get_symbol_return_tl {\#8}
3050         \tl_set:Nn \l_stex_get_symbol_invoke_cs {\#9}
3051     }
3052 }

```

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

13.7 Inheritance

13.7.1 `\importmodule/\usemodule`

```

3053 <@=stex_importmodule>
\usemodule
3054 \stex_new_styable_cmd:nnnn {usemodule} { 0{} m } {
3055     \stex_import_module_uri:nn { #1 }{ #2 }
3056     \stex_import_require_module:ooo
3057         \l_stex_import_archive_str
3058         \l_stex_import_path_str
3059         \l_stex_import_name_str
3060     \stex_if_do_html:T {
3061         \hbox{\stex_annotation_invisible:nn
3062             {shtml:usemodule=\l_stex_import_ns_str} {}}
3063     }
3064     \stex_if_smsmode:F{
3065         \group_begin:
3066         \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3067         \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3068         \tl_clear:N \thisstyle
3069         \stex_style_apply:
3070         \group_end:
3071     }
3072 }{}

```

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

```
\stex_import_module_uri:nn
3073 \cs_new_protected:Nn \stex_import_module_uri:nn {
3074     \stex_debug:nn{importmodule}{URI:~>#1<~>#2<}
```

```

3075 \exp_args:NNnx \seq_set_split:Nnn \__stex_importmodule_seq ? { \tl_to_str:n{ #2 } }
3076 \seq_pop_right:NN \__stex_importmodule_seq \l_stex_import_name_str
3077 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \__stex_importmodule_seq ? }
3078 \tl_if_empty:nTF { #1 } {
3079   \stex_debug:nn{importmodule}{No~archive}
3080   \prop_if_exist:NTF \l_stex_current_archive_prop {
3081     \stex_debug:nn{importmodule}{Picking~current~archive}
3082     \str_set:Nx \l_stex_import_archive_str {
3083       \prop_item:Nn \l_stex_current_archive_prop { id }
3084     }
3085   }{
3086     \str_clear:N \l_stex_import_archive_str
3087     \str_set:Nn \l_stex_import_uri_str {file:}
3088     \str_if_empty:NTF \l_stex_import_path_str {
3089       \stex_debug:nn{importmodule}{Empty~Path}
3090       \stex_file_split_off_ext:NN \l__stex_importmodule_path_seq \g_stex_current_file
3091       \stex_file_split_off_lang:NN \l__stex_importmodule_path_seq \l__stex_importmodule_pa
3092       \str_set:Nx \l_stex_import_path_str {
3093         \stex_file_use:N \l__stex_importmodule_path_seq
3094       }
3095     }{
3096       \stex_debug:nn{importmodule}{Resolving~path~\l_stex_import_path_str~relative~to~\ste
3097       \stex_file_resolve:Nx \l__stex_importmodule_seq { \stex_file_use:N \g_stex_current_f
3098       \str_set:Nx \l_stex_import_path_str {
3099         \stex_file_use:N \l__stex_importmodule_seq
3100       }
3101       \stex_debug:nn{importmodule}{...yields~\l_stex_import_path_str}
3102     }
3103   }
3104 }{
3105   \stex_debug:nn{importmodule}{Archive~#1}
3106   \str_set:Nx \l_stex_import_archive_str { #1 }
3107   \stex_require_archive:o \l_stex_import_archive_str
3108   \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ \l_stex_import_archi
3109 }
3110 }
3111 }
```

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

```

\stex_import_require_module:nnn
\stex_import_require_module:ooo
3111 \cs_new_protected:Npn \stex_import_require_module:nnn #1 {
3112   \tl_if_empty:nTF { #1 } {
3113     \str_clear:N \l__stex_importmodule_archive_str
3114     \str_set:Nn \l_stex_import_uri_str {file:}
3115     \__stex_importmodule_get_module:nnn {}
3116   }{
3117     \stex_require_archive:n { #1 }
3118     \str_set:Nx \l__stex_importmodule_archive_str {#1}
3119     \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ #1 _manifest_prop}{}
3120     \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
3121     \exp_args:No \__stex_importmodule_get_module:nnn \l__stex_importmodule_str
3122   }
3123 }
3124 \cs_generate_variant:Nn \stex_import_require_module:nnn {ooo}
```

```

3125 \cs_new_protected:Npn \stex_import_require_module_safe:nnn #1 {
3126   \tl_if_empty:nTF { #1 } {
3127     \str_clear:N \l__stex_importmodule_archive_str
3128     \str_set:Nn \l_stex_import_uri_str {file:}
3129     \__stex_importmodule_get_module_safe:nnn {}
3130   }{
3131     \stex_require_archive:n { #1 }
3132     \str_set:Nx \l__stex_importmodule_archive_str {#1}
3133     \str_set:Nx \l_stex_import_uri_str { \prop_item:cnd{ c_stex_mathhub_ #1 _manifest_prop}{}
3134     \str_set:Nx \l__stex_importmodule_str { \stex_file_use:N \c_stex_mathhub_file / #1 / sou
3135     \exp_args:No \__stex_importmodule_get_module_safe:nnn \l__stex_importmodule_str
3136   }
3137 }
3138 }
3139
3140 \cs_new_protected:Nn \__stex_importmodule_get_module_uri:nnn {
3141   \tl_if_empty:nF {#2} {
3142     \str_set:Nx \l_stex_import_uri_str { \l_stex_import_uri_str / #2}
3143   }
3144   \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
3145     Current~file:\stex_file_use:N \g_stex_current_file^^J
3146     Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3147   }
3148   \stex_if_module_exists:nTF { \l_stex_import_uri_str?#3} {
3149     \str_set:Nx \l_stex_import_ns_str { \l_stex_import_uri_str?#3}
3150   }{
3151     \stex_if_module_exists:nTF{ \stex_uri_use:N \l_stex_current_ns_uri ? #3} {
3152       \str_set:Nx \l_stex_import_ns_str { \stex_uri_use:N \l_stex_current_ns_uri ? #3}
3153     }{
3154       \__stex_importmodule_get_from_file:nnn{#1}{#2}{#3}
3155       \str_set:Nx \l_stex_import_ns_str { \l_stex_import_uri_str?#3}
3156     }
3157   }
3158 }
3159 \cs_new_protected:Nn \__stex_importmodule_get_module_uri_safe:nnn {
3160   \tl_if_empty:nF {#2} {
3161     \str_set:Nx \l_stex_import_uri_str { \l_stex_import_uri_str / #2}
3162   }
3163   \stex_debug:nn{importmodule}{~>#1<^^J>#2<^^J>#3<^^J>\l_stex_import_uri_str<^^J
3164     Current~file:\stex_file_use:N \g_stex_current_file^^J
3165     Current~namespace:\stex_uri_use:N \l_stex_current_ns_uri
3166   }
3167   \stex_if_module_exists:nTF { \l_stex_import_uri_str?#3} {
3168     \str_set:Nx \l_stex_import_ns_str { \l_stex_import_uri_str?#3}
3169   }{
3170     \stex_if_module_exists:nTF{ \stex_uri_use:N \l_stex_current_ns_uri ? #3} {
3171       \str_set:Nx \l_stex_import_ns_str { \stex_uri_use:N \l_stex_current_ns_uri ? #3}
3172     }{
3173       \__stex_importmodule_get_from_file_safe:nnn{#1}{#2}{#3}
3174       \str_set:Nx \l_stex_import_ns_str { \l_stex_import_uri_str?#3}
3175     }
3176   }
3177 }
3178

```

```

3179 \cs_new_protected:Nn \__stex_importmodule_get_module:nnn {
3180   \stex_debug:nn{importmodule}{Requiring~>[#1]#2?#3<}
3181   \__stex_importmodule_get_module_uri:nnn{#1}{#2}{#3}
3182   \stex_activate_module:o \l_stex_import_ns_str
3183 }
3184
3185 \cs_new_protected:Nn \__stex_importmodule_get_module_safe:nnn {
3186   \stex_debug:nn{importmodule}{Requiring~>[#1]#2?#3<}
3187   \__stex_importmodule_get_module_uri_safe:nnn{#1}{#2}{#3}
3188 }
3189
3190 \cs_new_protected:Nn \__stex_importmodule_get_from_file:nnn {
3191   \stex_file_resolve:Nx \l_stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3192   \str_set:Nx \l_stex_importmodule_str {\stex_file_use:N \l_stex_importmodule_seq}
3193   \stex_debug:nn{imports}{Looking-for~\l_stex_import_uri_str?#3...}
3194   \__stex_importmodule_check_file:nn{ /#3.tex }{
3195     \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}){
3196       \__stex_importmodule_check_file:nn{/#3.en.tex} {
3197         \__stex_importmodule_check_file:nn{.tex} {
3198           \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex} {
3199             \__stex_importmodule_check_file:nn{.en.tex} {
3200               \msg_error:nnx{\stex}{error/unknownmodule}{\l_stex_import_uri_str?#3}
3201             }
3202           }
3203         }
3204       }
3205     }
3206   }
3207 \stex_if_smsmode:TF{
3208   \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l_stex_importmodule_str}{\stex_file_use:N
3209     \stex_debug:nn{imports}{Skipping-current-file}
3210   }{
3211     \__stex_importmodule_load_file:n{#3}
3212   }
3213 }{
3214   \__stex_importmodule_load_file:n{#3}
3215 }
3216 }
3217 \cs_new_protected:Nn \__stex_importmodule_get_from_file_safe:nnn {
3218   \stex_file_resolve:Nx \l_stex_importmodule_seq { \tl_if_empty:nF{ #1 }{ #1 / } #2 }
3219   \str_set:Nx \l_stex_importmodule_str {\stex_file_use:N \l_stex_importmodule_seq}
3220   \stex_debug:nn{imports}{Looking-for~\l_stex_import_uri_str?#3...}
3221   \__stex_importmodule_check_file:nn{ /#3.tex }{
3222     \__stex_importmodule_check_file:nn{/#3.\l_stex_current_language_str .tex}{
3223       \__stex_importmodule_check_file:nn{/#3.en.tex} {
3224         \__stex_importmodule_check_file:nn{.tex} {
3225           \__stex_importmodule_check_file:nn{.\l_stex_current_language_str .tex} {
3226             \__stex_importmodule_check_file:nn{.en.tex} {
3227               }
3228             }
3229           }
3230         }
3231       }
3232     }

```

```

3233 \stex_if_smsmode:TF{
3234   \exp_args:NNo \exp_args:Nnx \str_if_eq:nnTF{\l__stex_importmodule_str}{\stex_file_use:N
3235     \stex_debug:nn{imports}{Skipping-current-file}
3236   }{
3237     \IfFileExists{ \l__stex_importmodule_str }{
3238       \__stex_importmodule_load_file:n{#3}
3239     }{}
3240   }
3241 }{
3242   \IfFileExists{ \l__stex_importmodule_str }{
3243     \__stex_importmodule_load_file:n{#3}
3244   }{}
3245 }
3246 }
3247
3248 \cs_new_protected:Nn \__stex_importmodule_load_file:n {
3249   \stex_file_in_smsmode:on \l__stex_importmodule_str {
3250     \str_if_empty:NF \l__stex_importmodule_archive_str {
3251       \stex_set_current_archive:n \l__stex_importmodule_archive_str
3252     }
3253     \stex_debug:nn{modules}{Loading~\l__stex_importmodule_str}
3254   }
3255   \stex_if_module_exists:nF {\l_stex_import_uri_str?#1} {
3256     \msg_error:nnx{stex}{error/unknownmodule}{\l_stex_import_uri_str?#1}
3257   }
3258 }
3259
3260 \cs_new_protected:Npn \__stex_importmodule_check_file:nn #1 {
3261   \stex_debug:nn{imports}{Checking~\l__stex_importmodule_str #1}
3262   \IfFileExists{ \l__stex_importmodule_str #1 }{
3263     \stex_debug:nnf{imports}{Success}
3264     \str_set:Nx \l__stex_importmodule_str { \l__stex_importmodule_str #1 }
3265   }
3266 }

```

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

`\importmodule`

```

3267 \stex_new_stylable_cmd:nnnn{importmodule} { O{} m } {
3268   \__stex_importmodule_import_module:nn {#1}{#2}
3269   \stex_smsmode_do:
3270 }{}
3271 \stex_deactivate_macro:Nn \importmodule {module-environments}
3272
3273 \cs_new_protected:Nn \__stex_importmodule_import_module:nn {
3274   \stex_import_module_uri:nn { #1 }{ #2 }
3275   \stex_import_require_module:ooo
3276     \l_stex_import_archive_str
3277     \l_stex_import_path_str
3278     \l_stex_import_name_str
3279   \stex_execute_in_module:x{
3280     \stex_activate_module:n{\l_stex_import_ns_str}
3281   }
3282   \stex_module_add_morphism:nonn

```

```

3283     {}{\l_stex_import_ns_str}{import}{}}
3284 \stex_if_do_html:T {
3285   \stex_annotation_invisible:nn
3286   {shtml:import=\l_stex_import_ns_str} {}
3287 }
3288 \stex_if_smsmode:F{
3289   \group_begin:
3290   \tl_set:Nn \thisarchive {\#1}
3291   \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3292   \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3293   \tl_clear:N \thisstyle
3294   \stex_style_apply:
3295   \group_end:
3296 }
3297 }
3298
3299 \cs_new_protected:Nn \__stex_importmodule_import_module_presms:nn {
3300   \stex_import_module_uri:nn { #1 }{ #2 }
3301   \tl_gput_right:Nx \g_stex_sms_import_code {
3302     \stex_import_require_module_safe:nnn
3303     {\l_stex_import_archive_str}
3304     {\l_stex_import_path_str}
3305     {\l_stex_import_name_str}
3306   }
3307 }
3308
3309 \stex_sms_allow_escape:N \importmodule
3310 \stex_every_module:n {\stex_reactivate_macro:N \importmodule}
3311 \stex_sms_allow_import:Nn \importmodule {
3312   \stex_reactivate_macro:N \importmodule
3313   \let \__stex_importmodule_import_module:nn \__stex_importmodule_import_module_presms:nn
3314 }
3315
3316 \stex_new_stylable_cmd:nnnn[requiremodule] { 0{} m } {
3317   \stex_import_module_uri:nn { #1 }{ #2 }
3318   \stex_import_require_module:ooo
3319   \l_stex_import_archive_str
3320   \l_stex_import_path_str
3321   \l_stex_import_name_str
3322   \stex_do_up_to_module:x{
3323     \stex_activate_module:n{\l_stex_import_ns_str}
3324   }
3325   \stex_if_do_html:T {
3326     \stex_annotation_invisible:nn
3327     {shtml:import=\l_stex_import_ns_str} {}
3328   }
3329 \stex_if_smsmode:F{
3330   \group_begin:
3331   \tl_set:Nn \thisarchive {\#1}
3332   \tl_set_eq:NN \thismoduleuri \l_stex_import_ns_str
3333   \tl_set_eq:NN \thismodulename \l_stex_import_name_str
3334   \tl_clear:N \thisstyle
3335   \stex_style_apply:
3336   \group_end:

```

```

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

```

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

13.7.2 Theory Morphisms

```

3341 <@=stex_morphisms>
3342 \stex_new_stylable_cmd:nnnn {assign} { m m }{
3343   \stex_get_in_morphism:n{#1}
3344   \stex_assign_do:n{#2}
3345   \stex_smsmode_do:
3346 }{}
3347 \stex_sms_allow_escape:N\assign
3348
3349 \cs_new_protected:Nn \stex_assign_do:n{
3350   \stex_debug:nn{assign}{Assigning~\l_stex_get_symbol_name_str~to~\tl_to_str:n{#1}}
3351   \tl_if_empty:NF \l_stex_get_symbol_def_tl {
3352     \%msg_error:nnxx{stex}{error/symbolalreadydefined}{\l_stex_get_symbol_name_str}{%
3353       % morphism~\l_stex_feature_name_str
3354       %}
3355   }
3356   \stex_check_term:n{#1}
3357   \stex_debug:nn{HERE!}{%
3358     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str^J
3359     \tl_to_str:n{#1}
3360   }
3361   \stex_if_do_html:T{
3362     \stex_annotation_invisible:nn{shtml:assign={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_
3363       \stex_annotation_force_break:n{
3364         \mode_if_math:T\hbox{\$ \stex_annotation:nn{shtml:definiens={}{}{#1} \$}
3365       }
3366     }
3367   }
3368   \exp_args:Ne \stex_metagroup_do_in:nx{
3369     \l_stex_current_module_str / \l_stex_feature_name_str
3370   }{
3371     \prop_put:Nnn \exp_not:N \l_stex_morphism_symbols_prop
3372     {[ \l_stex_get_symbol_mod_str ] / [ \l_stex_get_symbol_name_str ]}
3373     {
3374       {\l_stex_get_symbol_macro_str}
3375       {\int_use:N \l_stex_get_symbol_arity_int}
3376       {\l_stex_get_symbol_args_tl}
3377       {\exp_not:n{#1}}
3378       {\exp_args:No \exp_not:n \l_stex_get_symbol_type_tl}
3379       {\exp_args:No \exp_not:n \l_stex_get_symbol_return_tl}
3380       {\l_stex_get_symbol_invoke_cs}
3381     }
3382   }
3383 }
3384
3385

```

```

3386 \stex_new_stylable_cmd:nnnn {renamedecl} { m 0{} m }{
3387   \stex_get_in_morphism:n{#1}
3388   \_stex_renamedecl_do:nn{#2}{#3}
3389   \stex_smsmode_do:
3390 }{}
3391 \stex_sms_allow_escape:N\renamedecl
3392
3393 \cs_new_protected:Nn \_stex_renamedecl_do:nn {
3394   \stex_debug:nn{renamedecl}{Renaming~\l_stex_get_symbol_name_str~to~[#1]{#2}}
3395   \stex_if_do_html:T{
3396     \exp_args:Ne \stex_annotation_invisible:nn{
3397       shtml:rename={\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str},
3398       shtml:macroname={#2}
3399       \str_if_empty:nF{#1}{ ,shtml:to={#1} }
3400     }{}
3401   }
3402   \exp_args:Ne \stex_metagroup_do_in:nx{
3403     \l_stex_current_module_str / \l_stex_feature_name_str
3404   }{
3405     \prop_put:Nnn \exp_not:N \l_stex_morphism_renames_prop
3406     {\l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str}{##2}{%
3407       \tl_if_empty:nTF{#1}{\l_stex_feature_name_str/\l_stex_get_symbol_name_str}{#1}
3408     }%
3409   }
3410 }
3411
3412 \stex_new_stylable_cmd:nnnn {assignMorphism} { m m }{
3413   \str_clear:N \l__stex_morphisms_morphism_dom_str
3414   \stex_iterate_morphisms:nn\l_stex_current_domain_str{
3415     \stex_debug:nn{assignMorphism}{%
3416       Checking:~#1-vs:^^J##1^^J##2^^J##3^^J##4
3417     }
3418     \str_if_eq:nnTF{#1}{##1}{%
3419       \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3420     }{
3421       \stex_str_if_ends_with:nnT{##2}{#1}{%
3422         \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3423       }
3424     }
3425   }
3426   \str_if_empty:NT \l__stex_morphisms_morphism_dom_str {
3427     \msg_error:nnn{\stex}{error/nomorphism}{#1}
3428   }
3429   \bool_set_false:N \l_tmpa_bool
3430   \stex_iterate_morphisms:nn \l_stex_current_module_str {
3431     \stex_debug:nn{assignMorphism}{%
3432       Checking:~#2-vs:^^J##1^^J##2^^J##3^^J##4
3433     }
3434     \str_if_eq:nnTF{#2}{##1}{%
3435       \stex_debug:nn{assignMorphism}{match!}
3436       \stex_iterate_break:n{
3437         \stex_annotation_invisible:nn{
3438           shtml:assignMorphismFrom={\l__stex_morphisms_morphism_dom_str}
3439           ahtml:assignMorphismTo={\l_stex_current_module_str?##1}

```

```

3440     }{}
3441     \bool_set_true:N \l_tmpa_bool
3442   }
3443 {
3444   \stex_if_ends_with:nnT{##2}{#2}{
3445     \stex_debug:nn{assignMorphism}{match!}
3446     \stex_iterate_break:n{
3447       \stex_annotation_invisible:nn{
3448         shtml:assignMorphismFrom={\l_stex_morphisms_morphism_dom_str},
3449         shtml:assignMorphismTo={\l_stex_current_module_str?##1}
3450       }{}
3451       \bool_set_true:N \l_tmpa_bool
3452     }
3453   }
3454 }
3455 }
3456 \bool_if:NF \l_tmpa_bool {
3457   \msg_error:nnn{\stex}{error/nomorphism}{#2}
3458 }
3459 }{}
3460 \cs_new_protected:Nn \__stex_morphisms_do_morph_assign:nnn {
3461   \stex_iterate_break:n{
3462     \str_set:Nx \l_stex_morphisms_morphism_dom_str { \l_stex_current_domain_str ? #1 }
3463     \stex_debug:nn{assignMorphism}{match!}
3464     \stex_iterate_symbols:nn{#2}{{
3465       \stex_debug:nn{assignMorphism}{removing~##1?##3}
3466       % TODO: non-trivial assignments
3467       \prop_remove:NN \l_stex_morphism_symbols_prop {
3468         [##1]/[##3]
3469       }
3470     }
3471   }
3472 }
3473 }
3474 \stex_deactivate_macro:Nn \assign {morphism~environments}
3475 \stex_deactivate_macro:Nn \renamedecl {morphism~environments}
3476 \stex_deactivate_macro:Nn \assignMorphism {morphism~environments}
3477 \stex_new_stylable_env:nnnnnnn {copymodule}{m 0{} m}{%
3478
3479   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3480
3481   \stex_if_smsmode:F {
3482     \tl_set:Nn \thiscopyname { #1 }
3483     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3484     \stex_style_apply:
3485   }
3486   \stex_smsmode_do:
3487 }{
3488   \stex_if_smsmode:F {
3489     \stex_style_apply:
3490   }
3491   \stex_structural_feature_morphism_end:
3492 }{}{}{}%
3493 \stex_deactivate_macro:Nn \copymodule {module~environments}

```

```

3494 \stex_every_module:n {
3495   \stex_reactivate_macro:N \copymodule
3496 }
3497 \stex_sms_allow_env:n{copymodule}
3498
3499 \stex_new_stylable_env:nnnnnn {interpretmodule}{m 0{} m}{
3500   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3501   \stex_if_smsmode:F {
3502     \tl_set:Nn \thiscopyname { #1 }
3503     \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3504     \stex_style_apply:
3505   }
3506   \stex_smsmode_do:
3507 }{
3508   \stex_structural_feature_morphism_check_total:
3509   \stex_if_smsmode:F {
3510     \stex_style_apply:
3511   }
3512   \stex_structural_feature_morphism_end:
3513 }{}{}{}
3514 \stex_deactivate_macro:Nn \interpretmodule {module~environments}
3515 \stex_every_module:n {
3516   \stex_reactivate_macro:N \interpretmodule
3517 }
3518 \stex_sms_allow_env:n{interpretmodule}
3519 \stex_new_stylable_env:nnnnnn {realization}{0{} m}{

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

3543 \cs_new_protected:Nn \__stex_morphisms_parse_assign:n {
3544   \str_clear:N \l__stex_morphisms_name_str
3545   \str_clear:N \l__stex_morphisms_newname_str
3546   \tl_clear:N \l__stex_morphisms_ass_tl

```

```

3547 \stex_debug:nn{morphisms}{Parsing-#1}
3548 \exp_args:NNe \seq_set_split:Nnn \l__stex_morphisms_seq {\tl_to_str:n{0}} {#1}
3549 \int_compare:nNnTF {\seq_count:N \l__stex_morphisms_seq} = 1 {
3550   \stex_debug:nn{morphisms}{No~@}
3551   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_next_tl
3552 }{
3553   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3554   \stex_debug:nnf{morphisms}{Name:~\l__stex_morphisms_name_str}
3555   \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3556   \tl_set:Nx \l__stex_morphisms_next_tl {\seq_use:Nn \l__stex_morphisms_seq @}
3557 }
3558 \exp_args:NNNo \seq_set_split:Nnn \l__stex_morphisms_seq = \l__stex_morphisms_next_tl
3559 \str_if_empty:NTF \l__stex_morphisms_name_str {
3560   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3561   \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3562   \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3563 }{
3564   \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_newname_str
3565   \exp_args:NNo \str_set:Nn \l__stex_morphisms_newname_str \l__stex_morphisms_newname_str
3566   \tl_set:Nx \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3567 }
3568 \__stex_morphisms_do_parsed_assign:
3569 }
3570
3571 \cs_new_protected:Nn \__stex_morphisms_do_parsed_assign: {
3572   \exp_args:No \stex_get_in_morphism:n \l__stex_morphisms_name_str
3573   \str_if_empty:NF \l__stex_morphisms_newname_str {
3574     \exp_after:wN \__stex_morphisms_do_parsed_newname: \l__stex_morphisms_newname_str \__ste
3575   }
3576   \tl_if_empty:NF \l__stex_morphisms_ass_tl {
3577     \exp_args:No \stex_assign_do:n \l__stex_morphisms_ass_tl
3578   }
3579 }
3580
3581 \cs_new_protected:Nn \__stex_morphisms_do_parsed_newname: {
3582   \peek_charcode:NTF [ {
3583     \__stex_morphisms_do_parsed_newname:w
3584   }{
3585     \__stex_morphisms_do_parsed_newname:w []
3586   }
3587 }
3588
3589 \cs_new_protected:Npn \__stex_morphisms_do_parsed_newname:w [#1] #2 \__stex_morphisms_end: {
3590   \stex_renamedecl_do:nn{#1}{#2}
3591 }
3592
3593 \stex_new_stylable_cmd:nnnn{copymod}{m 0{} m m}{
3594   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=false}
3595
3596 \clist_map_function:nN{#4}\__stex_morphisms_parse_assign:n
3597
3598 \stex_if_smsmode:F {
3599   \tl_set:Nn \thiscopyname { #1 }
3600   \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str

```

```

3601     \stex_style_apply:
3602 }
3603 \stex_structural_feature_morphism_end:
3604 \stex_smsmode_do:
3605 }{}
3606 \stex_deactivate_macro:Nn \copymod {module~environments}
3607 \stex_every_module:n {
3608   \stex_reactivate_macro:N \copymod
3609 }
3610 \stex_sms_allow_escape:N\copymod
3611
3612
3613 \stex_new_stylable_cmd:nnnn{interpretmod}{0{} m m}{
3614   \stex_structural_feature_morphism:nnnnn{#1}{morphism}{#2}{#3}{,shtml:total=true}
3615
3616 \clist_map_function:nN{#4}\_stex_morphisms_parse_assign:n
3617
3618 \stex_if_smsmode:F {
3619   \tl_set:Nn \thiscopynname { #1 }
3620   \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3621   \stex_style_apply:
3622 }
3623 \stex_structural_feature_morphism_check_total:
3624 \stex_structural_feature_morphism_end:
3625 \stex_smsmode_do:
3626 }{}
3627 \stex_deactivate_macro:Nn \interpretmod {module~environments}
3628 \stex_every_module:n {
3629   \stex_reactivate_macro:N \interpretmod
3630 }
3631 \stex_sms_allow_escape:N\interpretmod
3632
3633
3634 \stex_new_stylable_cmd:nnnn{realize}{0{} m m}{
3635   \stex_structural_feature_morphism:nnnnn{}{morphism}{#1}{#2}{,shtml:total=true}
3636
3637 \clist_map_function:nN{#3}\_stex_morphisms_parse_assign:n
3638
3639 \stex_if_smsmode:F {
3640   \tl_set:Nn \thiscopynname { #1 }
3641   \tl_set_eq:NN \thismoduleuri \l_stex_current_domain_str
3642   \stex_style_apply:
3643 }
3644 \stex_structural_feature_morphism_check_total:
3645 \stex_structural_feature_morphism_end:
3646 \stex_smsmode_do:
3647 }{}
3648 \stex_deactivate_macro:Nn \realize {module~environments}
3649 \stex_every_module:n {
3650   \stex_reactivate_macro:N \realize
3651 }
3652 \stex_sms_allow_escape:N\realize

```

13.8 Symbols

13.8.1 Declarations

3653 `<@@=stex_symdecl>`

Some setup:

```
\stex_if_check_terms_p:
\stex_if_check_terms:TF
3654 \stex_if_html_backend:TF {
3655   \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3656     \prg_return_false:
3657   }
3658 }{
3659   \stex_get_env:Nn\__stex_symdecl_env_str{STEX_CHECKTERMS}
3660   \str_if_empty:NF\__stex_symdecl_env_str{
3661     \exp_args:No \str_if_eq:nnF \__stex_symdecl_env_str{false}{
3662       \bool_set_true:N \c_stex_check_terms_bool
3663     }
3664   }
3665   \bool_if:NTF \c_stex_check_terms_bool {
3666     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3667       \prg_return_true:
3668     }
3669   }{
3670     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
3671       \prg_return_false:
3672     }
3673   }
3674 }
```

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

```
\stex_check_term:n
3675 \stex_if_check_terms:TF{
3676   \cs_new_protected:Nn \stex_check_term:n {
3677     \hbox_set:Nn \l_tmpa_box {
3678       \group_begin:
3679         $$#1$$
3680       \group_end:
3681     }
3682   }
3683 }{
3684   \cs_new_protected:Nn \stex_check_term:n {}
3685 }
```

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

```
3686 \stex_keys_define:nnnn{symargs}{
3687   \str_clear:N \l_stex_key_args_str
3688   \str_clear:N \l_stex_key_role_str
3689   \str_clear:N \l_stex_key_reorder_str
3690   \str_clear:N \l_stex_key_assoc_str
3691 }{
3692   args      .str_set:N  = \l_stex_key_args_str ,
3693   reorder   .str_set:N  = \l_stex_key_reorder_str ,
```

```

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

\symdecl
3719 \str_new:N \l_stex_mroname_str
3720 \stex_new_stylable_cmd:nnnn {symdecl} { s m O{} } {
3721   \stex_keys_set:nn{decl}{#3}
3722   \IfBooleanTF #1 {
3723     \str_clear:N \l_stex_mroname_str
3724   }{
3725     \str_set:Nx \l_stex_mroname_str { #2 }
3726   }
3727 \stex_symdecl_top:n{#2}
3728
3729 \stex_if_smsmode:F{
3730   \group_begin:
3731   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3732   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3733   \tl_set_eq:NN \thistype \l_stex_key_type_tl
3734   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
3735   \tl_set_eq:NN \thisargs \l_stex_key_args_str
3736   \tl_clear:N \thisstyle
3737   \stex_style_apply:
3738   \group_end:
3739 }
3740 \stex_smsmode_do:
3741 }{}
3742 \stex_deactivate_macro:Nn \symdecl {module~environments}
3743 \stex_every_module:n {\stex_reactivate_macro:N \symdecl}
3744 \stex_sms_allow_escape:N \symdecl

```

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

```

\stex_symdecl_top:n
3745 \cs_new_protected:Nn \stex_symdecl_top:n {
3746   \str_if_empty:NT \l_stex_key_name_str {
3747     \str_set:Nx \l_stex_key_name_str { #1 }
3748   }
3749   \stex_symdecl_do:
3750   \stex_symdecl_check_terms:
3751   \__stex_symdecl_add_decl:
3752   \stex_if_do_html:T {
3753     \stex_symdecl_html:
3754   }
3755 }
3756
3757 \cs_new_protected:Nn \__stex_symdecl_add_decl: {
3758   \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnnN} {
3759     {\l_stex_mroname_str}
3760     {\l_stex_key_name_str}
3761     {\int_use:N \l_stex_get_symbol_arity_int}
3762     {\l_stex_get_symbol_args_tl}
3763     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }%{\exp_args:No \exp_not:n \l_stex_key_def_tl
3764     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
3765     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
3766     \stex_invoke_symbol:
3767   }
3768   \exp_args:Ne \stex_ref_new_symbol:n
3769   {\l_stex_current_module_str?\l_stex_key_name_str}
3770 }
3771
3772 \cs_new:Nn \stex_return_args:nn {
3773   \svar{ARGUMENT_#1}\stex_eat_exclamation_point:
3774 }
3775
3776 \cs_new_protected:Nn \stex_symdecl_html: {
3777   \exp_args:Ne \stex_annotation_invisible:nn {
3778     \shtml:symdecl = {\l_stex_current_module_str ? \l_stex_key_name_str},
3779     \shtml:args = {\l_stex_key_args_str}
3780     \str_if_empty:NF \l_stex_mroname_str {
3781       \shtml:macroname={\l_stex_mroname_str}
3782     }
3783     \str_if_empty:NF \l_stex_key_assoc_str {
3784       \shtml:assocotype={\l_stex_key_assoc_str}
3785     }
3786     \str_if_empty:NF \l_stex_key_reorder_str {
3787       \shtml:reorderargs={\l_stex_key_reorder_str}
3788     }
3789     \str_if_empty:NF \l_stex_key_role_str {
3790       \shtml:role={\l_stex_key_role_str}
3791     }
3792   }{\hbox\bgroup\stex_annotation_force_break:n{
3793     \bool_set_true:N \stex_in_invisible_html_bool
3794     \tl_if_empty:NF \l_stex_key_type_tl {
3795       $ \stex_annotation:nn{\shtml:type={}{}{\l_stex_key_type_tl}}$}
3796     }
3797     \tl_if_empty:NF \l_stex_key_def_tl {

```

```

3798     $\\stex_annotate:nn{shtml:definiens={}}{\\l_stex_key_def_t1}$
3799   }
3800   \\tl_if_empty:NF \\l_stex_key_return_t1{
3801     \\exp_args:Nno \\use:n{
3802       \\cs_generate_from_arg_count:NNnn \\l__stex_symdecl_cs
3803       \\cs_set:Npn \\l_stex_get_symbol_arity_int} \\l_stex_key_return_t1
3804       \\tl_set:Nx \\l__stex_symdecl_args_t1 {\\_stex_map_args:N \\_stex_return_args:nn}
3805       $\\stex_annotate:nn{shtml:returntype={}}{
3806         \\exp_after:wN \\l__stex_symdecl_cs \\l__stex_symdecl_args_t1!
3807       }$}
3808   }
3809   \\clist_if_empty:NF \\l_stex_key_argtypes_clist {
3810     \\stex_annotate:nn{shtml:argtypes={}}{\\_stex_annotate_force_break:n{
3811       \\clist_map_inline:Nn \\l_stex_key_argtypes_clist {
3812         $\\stex_annotate:nn{shtml:type={}}{##1}$
3813       }
3814     }}
3815   }
3816 }\\egroup}
3817 }
```

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

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

```

3818   \\cs_new_protected:Nn \\stex_symdecl_do: {
3819     \\_stex_do_deprecation:n \\l_stex_key_name_str
3820     \\__stex_symdecl_parse_arity:
3821     \\__stex_symdecl_do_args:
3822   }
3823
3824   \\int_new:N \\l_stex_assoc_args_count
3825
3826   \\cs_new_protected:Nn \\__stex_symdecl_parse_arity: {
3827     \\int_zero:N \\l_stex_get_symbol_arity_int
3828     \\int_zero:N \\l_stex_assoc_args_count
3829     \\str_map_inline:Nn \\l_stex_key_args_str {
3830       \\str_case:nnF ##1 {
3831         0 { \\str_map_break: }
3832         1 { \\str_map_break:n{
3833           \\int_set:Nn \\l_stex_get_symbol_arity_int {1}
3834           \\str_set:Nn \\l_stex_key_args_str {i}
3835         } }
3836         2 { \\str_map_break:n{
3837           \\int_set:Nn \\l_stex_get_symbol_arity_int {2}
3838           \\str_set:Nn \\l_stex_key_args_str {ii}
3839         } }
3840         3 { \\str_map_break:n{
3841           \\int_set:Nn \\l_stex_get_symbol_arity_int {3}
3842           \\str_set:Nn \\l_stex_key_args_str {iii}
3843         } }
3844         4 { \\str_map_break:n{
3845           \\int_set:Nn \\l_stex_get_symbol_arity_int {4}
3846           \\str_set:Nn \\l_stex_key_args_str {iiii}
3847         } }
```

```

3848      5 { \str_map_break:n{
3849          \int_set:Nn \l_stex_get_symbol_arity_int {5}
3850          \str_set:Nn \l_stex_key_args_str {iiiii}
3851      } }
3852      6 { \str_map_break:n{
3853          \int_set:Nn \l_stex_get_symbol_arity_int {6}
3854          \str_set:Nn \l_stex_key_args_str {iiiiii}
3855      } }
3856      7 { \str_map_break:n{
3857          \int_set:Nn \l_stex_get_symbol_arity_int {7}
3858          \str_set:Nn \l_stex_key_args_str {iiiiiii}
3859      } }
3860      8 { \str_map_break:n{
3861          \int_set:Nn \l_stex_get_symbol_arity_int {8}
3862          \str_set:Nn \l_stex_key_args_str {iiiiiiii}
3863      } }
3864      9 { \str_map_break:n{
3865          \int_set:Nn \l_stex_get_symbol_arity_int {9}
3866          \str_set:Nn \l_stex_key_args_str {iiiiiiiii}
3867      } }
3868      i {\int_incr:N \l_stex_get_symbol_arity_int}
3869      b {\int_incr:N \l_stex_get_symbol_arity_int}
3870      a {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3871      B {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3872  }{
3873      \msg_error:nnnx{stex}{error/wrongargs}{
3874          \l_stex_current_module_str ? \l_stex_key_name_str
3875      }{##1}
3876  }
3877 }
3878 }

3879 \cs_new_protected:Nn \__stex_symdecl_do_args: {
3880     \tl_clear:N \l_stex_get_symbol_args_tl
3881     \int_step_inline:nn \l_stex_get_symbol_arity_int {
3882         \tl_put_right:Nn \l_stex_get_symbol_args_tl {##1}
3883         \tl_put_right:Nx \l_stex_get_symbol_args_tl {
3884             \str_item:Nn \l_stex_key_args_str {##1}
3885         }
3886     }
3887 }
3888 }

```

(End of definition for `\stex_symdecl_do:`. This function is documented on page [121](#).)

`_stex_symdecl_check_terms:`

```

3889 \cs_new_protected:Nn \_stex_symdecl_check_terms: {
3890     \stex_check_term:n{
3891         \stex_debug:nn{check_terms}{Checking~type...}
3892         \group_begin:\l_stex_key_type_tl\group_end:
3893         \stex_debug:nn{check_terms}{Checking~definiens...}
3894         \group_begin:\l_stex_key_def_tl\group_end:
3895         \stex_debug:nn{check_terms}{Checking~return...}
3896         \group_begin:\l_stex_key_return_tl!\group_end:
3897         \stex_debug:nn{check_terms}{Checking~argument~types...}

```

```

3898     \group_begin:\l_stex_key_argtypes_clist\group_end:
3899   }
3900 }

```

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

\textsymdecl

```

3901
3902 \stex_keys_define:nnnn{textsymdecl}{
3903   \str_clear:N \l_stex_key_name_str
3904   \tl_clear:N \l_stex_key_type_tl
3905   \tl_clear:N \l_stex_key_def_tl
3906 }{
3907   name      .str_set:N  = \l_stex_key_name_str ,
3908   type      .tl_set:N   = \l_stex_key_type_tl ,
3909   def       .tl_set:N   = \l_stex_key_def_tl
3910 }{style,deprecate}
3911
3912 \stex_new_stylable_cmd:nnnn {textsymdecl} {m 0{} m} {
3913   \stex_keys_set:nn{symdef}{}%
3914   \stex_keys_set:nn{textsymdecl}{#2}
3915   \str_set:Nx \l_stex_macroname_str { #1 }
3916   \str_if_empty:NT \l_stex_key_name_str {
3917     \str_set:Nn \l_stex_key_name_str {#1}
3918   }%
3919   % \str_set:Nx \l_stex_key_name_str {\l_stex_key_name_str-sym}
3920   %
3921   \str_set:Nn \l_stex_key_role_str {textsymdecl}
3922
3923 \stex_symdecl_do:
3924 \_stex_symdecl_check_terms:
3925 \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnN}{
3926   {\l_stex_macroname_str}
3927   {\l_stex_key_name_str}
3928   {0}{}%
3929   {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }%
3930   {}% type
3931   {\use:c{\#1name_nospace}}% return
3932   \stex_invoke_text_symbol:
3933 }
3934 \exp_args:Ne \stex_ref_new_symbol:n
3935   {\l_stex_current_module_str?\l_stex_key_name_str}
3936 \stex_if_do_html:T {
3937   \_stex_symdecl_html:
3938 }
3939
3940 \int_set:Nn \l_stex_get_symbol_arity_int 0
3941 \tl_clear:N \l_stex_key_op_tl
3942 \str_clear:N \l_stex_key_intent_str
3943 \str_clear:N \l_stex_key_prec_str
3944 \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
3945 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
3946 \stex_notation_parse:n{\hbox{#3}}
3947 \_stex_notation_add:

```

```

3948 \stex_if_do_html:T {
3949   \def\comp{\_comp}
3950   \l_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
3951 }
3952 \stex_execute_in_module:x{
3953   \__stex_symdecl_set_textsymdecl_macro:nnn{#1}{\l_stex_current_module_str?\l_stex_key_name}
3954   \exp_not:n{#3}
3955 }
3956
3957 \stex_if_smsmode:F{
3958   \group_begin:
3959   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
3960   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3961   \tl_clear:N \thisstyle
3962   \stex_style_apply:
3963   \group_end:
3964 }
3965 \stex_smsmode_do:
3966 }{}}
3967 \stex_deactivate_macro:Nn \textsymdecl {module~environments}
3968 \stex_every_module:n {\stex_reactivate_macro:N \textsymdecl}
3969 \stex_sms_allow_escape:N \textsymdecl
3970
3971 \cs_new_protected:Nn \l_stex_symdecl_set_textsymdecl_macro:nnn {
3972   \cs_set_protected:cpn{#1name_nospace}{#3}
3973   \cs_set_protected:cpn{#1name} {
3974     \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3975     \mode_if_math:T\hbox{\let\xspace\relax #3}
3976     \mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3977   }
3978 }
3979
3980 \cs_new_protected:Nn \stex_invoke_text_symbol: {
3981   \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3982   \l_stex_term_oms_or_omv:nnn{}{}{\maincomp{\let\xspace\relax\l_stex_current_return_tl}}
3983   \group_end:\mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3984 }

```

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

```

\stex_get_symbol:n
3985 \cs_new_protected:Nn \stex_get_symbol:n {
3986   \l_stex_get_symbol:n{ #1 }
3987   \str_if_empty:NT \l_stex_get_symbol_name_str {
3988     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3989   }
3990 }
3991
3992 \cs_new_protected:Nn \l_stex_get_symbol:n {
3993   \str_clear:N \l_stex_get_symbol_mod_str
3994   \str_clear:N \l_stex_get_symbol_name_str
3995   \cs_if_exist:cTF { #1 }{
3996     \cs_set_eq:Nc \l_stex_symdecl_cs { #1 }
3997     % command name

```

```

3998 \exp_args:Ne \tl_if_empty:nTF { \cs_argument_spec:N \l__stex_symdecl_cs }{
3999   % ...that takes no arguments
4000   \exp_args:Ne \cs_if_eq:NNTF {\tl_head:N \l__stex_symdecl_cs}
4001     \l_stex_invoke_symbol:nnnnnnnN
4002     \l__stex_symdecl_get_symbol_from_cs:
4003     {\l__stex_symdecl_get_symbol_from_string:n { #1 }}
4004   }{
4005     \l__stex_symdecl_get_symbol_from_string:n { #1 }
4006   }
4007   }{
4008     \l__stex_symdecl_get_symbol_from_string:n { #1 }
4009   }
4010 }
4011
4012 \int_new:N \l_stex_get_symbol_arity_int
4013 \cs_new_protected:Nn \l_stex_symdecl_get_symbol_from_cs: {
4014   \stex_debug:nn{symbols}{Getting~from~cs...}
4015   \stex_pseudogroup_with:nn{\l_stex_invoke_symbol:nnnnnnnnN}{
4016     \cs_set:Npn \l_stex_invoke_symbol:nnnnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {
4017       \str_set:Nn \l_stex_get_symbol_mod_str {##1}
4018       \str_set:Nn \l_stex_get_symbol_name_str {##2}
4019       \int_set:Nn \l_stex_get_symbol_arity_int {##3}
4020       \tl_set:Nn \l_stex_get_symbol_args_tl {##4}
4021       \tl_set:Nn \l_stex_get_symbol_def_tl {##5}
4022       \tl_set:Nn \l_stex_get_symbol_type_tl {##6}
4023       \tl_set:Nn \l_stex_get_symbol_return_tl {##7}
4024       \tl_set:Nn \l_stex_get_symbol_invoke_cs {##8}
4025     }
4026     \l__stex_symdecl_cs
4027   }
4028 }
4029
4030 \cs_new_protected:Nn \l_stex_symdecl_get_symbol_from_string:n {
4031   \stex_debug:nn{symbols}{Getting~from~string~#1...}
4032   \seq_set_split:Nnn \l_stex_symdecl_seq ? {#1}
4033   \seq_pop_right:NN \l_stex_symdecl_seq \l_stex_symdecl_name
4034   \seq_if_empty:NTF \l_stex_symdecl_seq {
4035     \exp_args:No \l_stex_symdecl_get_from_one_string:n {#1}
4036   }{
4037     \exp_args:NNe \exp_args:Nno \l_stex_symdecl_get_symbol_from_modules:nn {
4038       \seq_use:Nn \l_stex_symdecl_seq ?
4039     } \l_stex_symdecl_name
4040   }
4041 }
4042
4043 \cs_new_protected:Nn \l_stex_symdecl_sym_from_str_i:nnnn {
4044   \bool_lazy_any:nTF{
4045     {\str_if_eq_p:nn{#2}{#3}}
4046     {\str_if_eq_p:nn{#2}{#4}}
4047     {\str_if_eq_p:nn{#2}{#5}}
4048   }{
4049     \l_stex_symdecl_sym_i_finish:nnnnnnnN{#1}{#4}
4050   }{
4051     \l_stex_symdecl_sym_i_gobble:nnnnnn

```

```

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

```

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

13.8.2 Notations

```

4102 <@@=stex_notations>

\_stex_map_args:N
\stex_map_notation_args:N
4103 \cs_new:Nn \stex_map_args:N {
4104     \tl_if_empty:NF \l_stex_get_symbol_args_tl {
4105         \exp_after:wN \__stex_notations_map_args_i:w \exp_after:wN
4106         #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
4107     }
4108 }
4109 \cs_new:Npn \__stex_notations_map_args_i:w #1 #2 #3 #4 \__stex_notations_args_end: {
4110     #1 #2 #3
4111     \tl_if_empty:nF{#4} {
4112         \__stex_notations_map_args_i:w #1 #4 \__stex_notations_args_end:
4113     }
4114 }
4115
4116 \cs_new:Nn \stex_map_notation_args:N {
4117     \tl_if_empty:NF \l_stex_notation_args_tl {
4118         \exp_after:wN \__stex_notations_map_args_ii:w \exp_after:wN
4119         #1 \l_stex_get_symbol_args_tl \__stex_notations_args_end:
4120     }
4121 }
4122 \cs_new:Npn \__stex_notations_map_args_ii:w #1 #2 #3 #4 #5 #6 \__stex_notations_args_end: {
4123     #1 #2 #3 #4 #5
4124     \tl_if_empty:nF{#6} {
4125         \__stex_notations_map_args_ii:w #1 #6 \__stex_notations_args_end:
4126     }
4127 }

```

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

notation arguments:

```

4128 \stex_keys_define:nnnn{notation}{
4129     \str_clear:N \l_stex_key_variant_str
4130     \str_clear:N \l_stex_key_prec_str
4131     \str_clear:N \l_stex_key_op_tl
4132     \str_clear:N \l_stex_key_intent_str
4133     \clist_clear:N \l_stex_key_intent_args_clist
4134 }{
4135     variant      .str_set_x:N = \l_stex_key_variant_str ,
4136     prec        .str_set_x:N = \l_stex_key_prec_str ,
4137     op          .tl_set:N    = \l_stex_key_op_tl ,
4138     intent       .str_set:N = \l_stex_key_intent_str ,
4139     argnames     .clist_set:N = \l_stex_key_intent_args_clist ,
4140     unknown      .code:n    = {
4141         \str_if_empty:NTF \l_keys_key_str {
4142             \str_set:Nx \l_stex_key_variant_str {\l_keys_key_tl}
4143         }{
4144             \str_set_eq:NN \l_stex_key_variant_str \l_keys_key_str
4145         }
4146     }
4147 }{style}

```

```

\notation
4148 \stex_new_stylable_cmd:nnnn {notation} { s m 0{} m} {
4149   \stex_keys_set:nn{notation}{#3}
4150   \stex_get_symbol:n{#2}
4151   \stex_notation_parse:n{#4}
4152   \stex_if_check_terms:T{ \_stex_notation_check: }
4153   \_stex_notation_add:
4154   \stex_if_do_html:T {
4155     \def\comp{\_comp}
4156     \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4157   }
4158   \IfBooleanTF#1{
4159     \_stex_notation_set_default:n{
4160       \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4161     }
4162   }{}
4163   \stex_if_smsmode:F{
4164     \group_begin:
4165     \_\stex_notations_styledefs:
4166     \stex_style_apply:
4167     \group_end:
4168   }
4169   \stex_smsmode_do:
4170 }{}
4171
4172 \cs_new_protected:Nn \_\stex_notations_styledefs: {
4173   \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4174   \str_set:Nn \thisdeclname \l_stex_get_symbol_name_str
4175   \tl_set:Nx \thisdecluri {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4176   \def\thisnotation{
4177     $
4178     \tl_set_eq:NN \l_stex_current_symbol_str \thisdecluri
4179     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs} {
4180       \_stex_notation_make_args:
4181     }$
4182   }
4183 }
4184
4185 \stex_deactivate_macro:Nn \notation {module~environments}
4186 \stex_every_module:n {\stex_reactivate_macro:N \notation}
4187 \stex_sms_allow_escape:N \notation

```

(End of definition for `\notation`. This function is documented on page 87.)

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

```

4188 \cs_new_protected:Nn \stex_notation_parse:n {
4189   \tl_if_empty:NF \l_stex_key_op_tl {
4190     \tl_set:Nx \l_stex_key_op_tl { \exp_not:N\maincomp {
4191       \exp_args:No \exp_not:n \l_stex_key_op_tl
4192     } }
4193   }
4194   \seq_clear:N \l_\stex_notations_precs_seq
4195   \tl_clear:N \l_stex_notation_args_tl

```

```

4196 \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0 {
4197   \__stex_notations_const_precs:
4198   \tl_if_empty:NT \l_stex_key_op_tl {
4199     \tl_set:Nn \l_stex_key_op_tl { \maincomp{#1} }
4200   }
4201 }{
4202   \__stex_notations_fun_precs:
4203   \str_set:Nn \l__stex_notations_missing_str {#1}
4204   \tl_clear:N \l__stex_notations_missing_tl
4205   \stex_map_args:N \__stex_notations_add_missing_args:nn
4206   \tl_if_empty:NT \l_stex_key_op_tl {
4207     \hbox_set:Nn \l_tmpa_box {
4208       \str_set:Nn \l_stex_current_symbol_str {}
4209       \cs_set:Npn \l_tmpa_cs ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
4210       \cs_set:Npn \maincomp ##1 {
4211         \tl_gset:Nn \l_stex_key_op_tl { \maincomp{##1} }
4212         ##1
4213       }
4214       \cs_set:Npn \argsep ##1 ##2 {##1 ##2}
4215       \cs_set:Npn \argmap ##1 ##2 ##3 {##1 ##3}
4216       \cs_set:Npn \argarraymap ##1 ##2 ##3 ##4 {
4217         ##1 ##2
4218       }
4219       \stex_suppress_html:n{$\l_tmpa_cs abcdefghj$}
4220     }
4221   }
4222 }
4223 \exp_args:NNe
4224 \tl_set:Nn \l_stex_notation_macrocode_cs {
4225   \STEXInternalNotation
4226   { \l_stex_key_variant_str }
4227   { \l__stex_notations_oppref_tl }
4228   { \l_stex_key_intent_str }
4229   { \l_stex_notation_args_tl }
4230   {
4231     \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0
4232     { \exp_not:n { \maincomp{ #1 } } }
4233     { \exp_not:n { #1 } \l__stex_notations_missing_tl }
4234   }
4235 }
4236 \stex_debug:nn{notation}{Notation:~\meaning\l_stex_notation_macrocode_cs}
4237 }
4238
4239 \cs_new_protected:Nn \__stex_notations_const_precs: {
4240   \str_if_empty:NTF \l_stex_key_prec_str {
4241     \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4242   }{
4243     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4244       \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4245     }{
4246       \tl_set_eq:NN \l__stex_notations_oppref_tl \l_stex_key_prec_str
4247     }
4248   }
4249 }

```

```

4250
4251 \cs_new_protected:Nn \__stex_notations_fun_precs: {
4252   \str_if_empty:NTF \l_stex_key_prec_str {
4253     \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4254   }{
4255     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4256       \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4257     }{
4258       \tl_set_eq:NN \l__stex_notations_oppref_tl \l_stex_key_prec_str
4259     }
4260   }
4261 \str_if_empty:NTF \l_stex_key_prec_str {
4262   \tl_set:Nn \l__stex_notations_oppref_tl { 0 }
4263   \int_step_inline:nn \l_stex_get_symbol_arity_int {
4264     \seq_put_right:Nn \l__stex_notations_precs_seq {0}
4265   }
4266 }{
4267   \str_if_eq:onTF \l_stex_key_prec_str {nobrackets} {
4268     \stex_debug:nn{notation}{No~brackets}
4269     \tl_set:No \l__stex_notations_oppref_tl { \neginfpref }
4270     \int_step_inline:nn \l_stex_get_symbol_arity_int {
4271       \exp_args:NNo \seq_put_right:Nn \l__stex_notations_precs_seq \infpref
4272     }
4273   } \__stex_notations_parse_precs:
4274 }
4275 \__stex_notations_do_argnames:
4276 }
4277
4278 \cs_new_protected:Nn \__stex_notations_parse_precs: {
4279   \stex_debug:nn{notation}{parsing~precedence~\l_stex_key_prec_str}
4280   \seq_set_split:NnV \l__stex_notations_seq ; \l_stex_key_prec_str
4281   \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4282     \tl_set_eq:NN \l__stex_notations_oppref_tl \l__stex_notations_str
4283     \seq_pop_left:NNT \l__stex_notations_seq \l__stex_notations_str {
4284       \exp_args:NNo \seq_set_split:NnV \l__stex_notations_seq
4285         {\tl_to_str:n{x}} \l__stex_notations_str
4286     }
4287   }{
4288     \tl_set:No \l__stex_notations_oppref_tl { 0 }
4289   }
4290   \int_step_inline:nn \l_stex_get_symbol_arity_int {
4291     \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
4292       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_str
4293     }{
4294       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_oppref_tl
4295     }
4296   }
4297 }
4298
4299 \cs_new_protected:Nn \__stex_notations_do_argnames: {
4300   \tl_clear:N \l_stex_notation_args_tl
4301   \stex_map_args:N \__stex_notations_do_argname:nn
4302 }
4303

```

```

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

```

(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:
4325 \cs_new_protected:Nn \__stex_notation_check: {
4326   \stex_check_term:n{
4327     \str_set:Nn \l_stex_current_symbol_str {test}
4328     \cs_set:Npn \comp ##1 {##1}
4329     \stex_debug:nn{check_terms}{Checking~notation...}
4330     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{{
4331       \__stex_notation_make_args:
4332     }
4333   }
4334 }
4335
4336 \cs_new_protected:Nn \__stex_notation_add: {
4337   \stex_module_add_notation:eooeo{
4338     \l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str
4339     }\l_stex_key_variant_str
4340     {\int_use:N \l_stex_get_symbol_arity_int}
4341     \l_stex_notation_macrocode_cs
4342     \l_stex_key_op_tl
4343 }
4344
4345 \cs_new_protected:Nn \__stex_notation_do_html:n {
4346   \hbox{\stex_annotate_invisible:nn {
4347     \shtml:notation={#1},
4348     \shtml:notationfragment={\l_stex_key_variant_str},
4349     \shtml:precedence={\l_stex_notations_opprec_tl},
4350     \shtml:argprecs={\seq_use:Nn \l_stex_notations_precs_seq ,}
4351   }{
4352     \cs_set_protected:Npn \argsep ##1 ##2 {
4353       \stex_annotate:nn{\shtml:argsep={}}{

```

```

4354     ##1 ##2
4355 }
4356 }
4357 \cs_set_protected:Npn \argmap ##1 ##2 ##3 {
4358     \cs_set:Npn \__stex_notations_map_cs: #####1 { ##2 }
4359     \stex_annotate:nn{shtml:argmap={}}{
4360         \__stex_notations_map_cs:{##1} ##3
4361     }
4362 }
4363 \cs_set_protected:Npn \maincomp {
4364     \do_comp:nNn {maincomp}\compemph@uri
4365 }
4366 $
4367 \str_set:Nx \l_stex_current_symbol_str {#1}
4368 \stex_annotate:nn{shtml:notationcomp={}}{
4369     \exp_args:Nne \use:nn {
4370         \l_stex_notation_macrocode_cs {}
4371     }{
4372         \l_stex_map_args:N \__stex_notations_make_arg_html:nn
4373     }
4374 }
4375 $
4376 \tl_if_empty:NF \l_stex_key_op_tl {
4377     $
4378     \str_set:Nx \l_stex_current_symbol_str {#1}
4379     \stex_annotate:nn{shtml:notationopcomp={}}{
4380         \l_stex_term_oms:nnn{\l_stex_key_variant_str}{\l_stex_key_op_tl}
4381     }
4382     $
4383 }
4384 }
4385 }
4386
4387 \cs_new:Nn \__stex_notations_make_arg_html:nn {
4388 % \str_case:nnF #2 {
4389 %     a {{%
4390 %         \stex_annotate:nn{shtml:argnum=#1a}{x},
4391 %         \stex_annotate:nn{shtml:argnum=#1b}{x}
4392 %     }%
4393 %     B {{%
4394 %         \stex_annotate:nn{shtml:argnum=#1a}{x},
4395 %         \stex_annotate:nn{shtml:argnum=#1b}{x}
4396 %     }%
4397 % }{%
4398 %     {
4399 %         \stex_annotate:nn{shtml:argnum=#1}{x}
4400 %     }
4401 % }
4402 }
4403
4404 \cs_new:Nn \stex_notation_make_args: {
4405     \l_stex_map_notation_args:N \__stex_notations_make_arg:nnnn
4406 }
4407

```

```

4408 \cs_new:Nn \__stex_notations_make_arg:nnnn {
4409   \str_case:nnF #2 {
4410     a {
4411       a\c_math_subscript_token{#1,1},
4412       a\c_math_subscript_token{#1,2}
4413     }
4414     B {
4415       B\c_math_subscript_token{#1,1},
4416       B\c_math_subscript_token{#1,2}
4417     }
4418   }
4419   \{
4420     \__stex_term_arg:nnnnn{#1}{#2}{#3}{#4}
4421     {{#2}\c_math_subscript_token{#1}}
4422   }
4423 }

```

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

\setnotation

```

\_stex_notation_set_default:n
4424 \cs_new_protected:Npn \setnotation #1 #2 {
4425   \stex_get_symbol:n{#1}
4426   \cs_if_exist:cTF{l_stex_notation_%
4427     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4428     _#2_cs
4429   }{
4430     \tl_set_eq:Nc \l_stex_notation_macrocode_cs {l_stex_notation_%
4431       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4432       _#2_cs
4433   }
4434   \cs_if_exist:cTF{l_stex_notation_%
4435     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4436     _op_#2_cs
4437   }{
4438     \tl_set_eq:Nc \l_stex_key_op_tl {l_stex_notation_%
4439       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4440       _op_#2_cs
4441   }
4442   }{
4443     \tl_clear:N \l_stex_key_op_tl
4444   }
4445   \_stex_notation_set_default:n{
4446     \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4447   }
4448   }{
4449     \msg_error:nnxx{stex}{unknownnotation}{#2}%
4450       \l_stex_get_symbol_mod_str?\l_stex_get_symbol_name_str
4451   }
4452 }
4453 }
4454 \cs_new_protected:Nn \_stex_notation_set_default:n{
4455   \stex_module_add_notation:eooeo{#1}{}

```

```

4457   {\int_use:N \l_stex_get_symbol_arity_int}
4458   \l_stex_notation_macrocode_cs
4459   \l_stex_key_op_tl
4460 }

```

(End of definition for `\setnotation` and `_stex_notation_set_default:n`. These functions are documented on page 88.)

\varnotation

```

4461 \stex_new_stylable_cmd:nnnn {varnotation} { s m 0{} m} {
4462   \stex_keys_set:nn{notation}{#3}
4463   \stex_get_var:n{#2}
4464   \str_set_eq:NN \l_stex_key_name_str \l_stex_get_symbol_name_str
4465   \stex_notation_parse:n{#4}
4466   \stex_if_check_terms:T{ \_stex_notation_check: }
4467   \_stex_vardecl_notation_macro:
4468   \IfBooleanTF#1{
4469     \_stex_notation_set_default:n{\l_stex_get_symbol_name_str}
4470   }{}
4471   \group_begin:
4472   \tl_set_eq:NN \thisvarname \l_stex_get_symbol_name_str
4473   \tl_clear:N \thisstyle
4474   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4475   \def\thisnotation{
4476     $ \let\l_stex_current_symbol_str\thisvarname
4477     \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}
4478     \_stex_notation_make_args:
4479   }$}
4480 }
4481 \stex_style_apply:
4482 \group_end:
4483 }{}

```

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

\symdef

```

4484 \stex_keys_define:nnnn{symdef}{}{}{decl,notation}
4485
4486 \cs_new_protected:Nn \_stex_symdef_styledefs: {
4487   \tl_set:Nx \thisdecluri {\l_stex_current_module_str ? \l_stex_key_name_str}
4488   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
4489   \tl_set_eq:NN \thistype \l_stex_key_type_tl
4490   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
4491   \tl_set_eq:NN \thisargs \l_stex_key_args_str
4492   \tl_clear:N \thisstyle
4493   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4494   \def\thisnotation{
4495     $ \let\l_stex_current_symbol_str\thisdecluri
4496     \def\comp{\_comp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}
4497     \_stex_notation_make_args:
4498   }$}
4499 }
4500 }
4501 \stex_new_stylable_cmd:nnnn {symdef} { m 0{} m} {

```

```

4503 \stex_keys_set:nn{symdef}{#2}
4504 \str_set:Nx \l_stex_macroname_str { #1 }
4505 \stex_symdecl_top:n{#1}
4506 \stex_debug:nn{symdef}{Doing~\l_stex_current_module_str ? \l_stex_key_name_str}
4507 \str_set_eq:NN \l_stex_get_symbol_mod_str \l_stex_current_module_str
4508 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
4509 \stex_notation_parse:n{#3}
4510 \stex_debug:nn{Here!}{\meaning\l_stex_notation_args_tl}
4511 \_stex_notation_check:
4512 \_stex_notation_add:
4513 \stex_if_do_html:T{
4514   \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
4515 }
4516 \stex_if_smsmode:F{
4517   \group_begin:
4518   \_stex_symdef_styledefs:
4519   \stex_style_apply:
4520   \group_end:
4521 }
4522 \stex_smsmode_do:
4523 }{}
4524
4525 \stex_deactivate_macro:Nn \symdef {module~environments}
4526 \stex_every_module:n {\stex_reactivate_macro:N \symdef}
4527 \stex_sms_allow_escape:N \symdef

```

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

```

\stex_do_default_notation_op:
4528 \cs_new_protected:Nn \stex_do_default_notation: {
4529   \stex_do_default_notation_op:
4530   \tl_if_empty:NTF \l_stex_current_args_tl {
4531     \tl_clear:N \l__stex_notations_args_tl
4532   }{
4533     \__stex_notations_make_name:
4534     \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
4535     \tl_set:Nx \l__stex_notations_args_tl {
4536       \stex_map_args:N \__stex_notations_augment_arg:nn
4537     }
4538     \tl_put_right:Nn \l_stex_default_notation {\comp{}}
4539     \seq_clear:N \l_tmpa_seq
4540     \int_step_inline:nn \l_stex_current_arity_str {
4541       \seq_put_right:Nn \l_tmpa_seq {#### ##1}
4542     }
4543     \tl_put_right:Nx \l_stex_default_notation {
4544       \seq_use:Nn \l_tmpa_seq {\mathpunct{\comp{,}}}}
4545     }
4546     \tl_put_right:Nn \l_stex_default_notation {\comp{}}
4547   }
4548   \tl_set:Nx \l_stex_default_notation {\STEXInternalNotation{}{0}{}{\l__stex_notations_args_}
4549   \exp_args:No \exp_not:n \l_stex_default_notation
4550 }
4551 }
4552

```

```

4553 \cs_new:Nn \__stex_notations_augment_arg:nn {
4554   #1#2{0}{}
4555 }
4556
4557 \cs_new_protected:Nn \__stex_notations_make_name: {
4558   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq ? \l_stex_current_symbol_str
4559   \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
4560   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq / \l__stex_notations_name_str
4561   \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
4562 }
4563
4564 \cs_new_protected:Nn \stex_do_default_notation_op: {
4565   \__stex_notations_make_name:
4566   \tl_set:Nx \l_stex_default_notation {\exp_not:N \maincomp{ \exp_not:N \mathrm {\l__stex_no
4567 }

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

\STEXInternalNotation

```

4568 % 1: variant 2: operator precedence 3: intent 4: arguments 5: code 6: next
4569
4570 \cs_new_protected:Npn \STEXInternalNotation #1 #2 #3 #4 #5 #6 {
4571   \__stex_notations_process_notation:nnnnnn{\#1}{\#2}{\#3}{\#4}{\#5}{\#6}
4572   \l__stex_notations_code_tl
4573   #6
4574 }
4575
4576
4577 \cs_new_protected:Npn \__stex_notations_process_notation:nnnnnn #1 #2 #3 #4 {
4578   \tl_if_empty:nTF{\#4}{
4579     \__stex_notations_simple:nnnnn{\#1}{\#2}{\#3}
4580   }{
4581     \__stex_notations_complex:nnnnnn{\#1}{\#2}{\#3}{\#4}
4582   }
4583 }
4584
4585 \cs_new_protected:Nn \__stex_notations_simple:nnnnn {
4586   \stex_debug:nn{Notation~code}{\tl_to_str:n{\#4}}
4587   \tl_set:Nn \l__stex_notations_code_tl {
4588     \cs_set:Npn \l__stex_notations_cs {
4589       \__stex_maybe_brackets:nn{\#2}{\#3}{\#4}
4590       \__stex_term_oms_or_omv:nnn{\#1}{\#3}{\#4}
4591     }
4592   }
4593   \l__stex_notations_cs
4594 }
4595 \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4596 #5
4597 }
4598
4599 \cs_new_protected:Nn \__stex_notations_complex:nnnnnn {
4600   \stex_debug:nn{Notation~code}{\tl_to_str:n{\#5}}
4601   \int_zero:N \l_tmpa_int
4602   \tl_set:Nn \l__stex_notations_pre_tl {\cs_set_eq:NN \l__stex_term_oma_or_omb:nnn \l__stex_term
```

```

4603 \tl_set:Nn \l__stex_notations_code_tl {
4604   \cs_generate_from_arg_count:NNnn \l__stex_notations_cs \cs_set:Npn \l_tmpa_int
4605   {
4606     \l_stex_maybe_brackets:nn{#2}{
4607       \l_stex_term_oma_or_omb:nnn{#1}{#3}{%
4608         \bool_set_false:N \l_stex_brackets_dones_bool
4609         #5
4610       }
4611     }
4612   }
4613   \l__stex_notations_cs
4614 }
4615 \tl_set:Nn \l__stex_notations_after_tl{
4616   \exp_args:NNo
4617   \tl_put_left:Nn \l__stex_notations_code_tl \l__stex_notations_pre_tl
4618   \tl_put_left:Nx \l__stex_notations_code_tl {
4619     \int_set:Nn \l_tmpa_int {\int_use:N \l_tmpa_int}
4620   }
4621   \stex_debug:nn{Here:Notation}{\meaning \l__stex_notations_code_tl }
4622   #6
4623 }
4624 \__stex_notations_parse_notation_args:nnnw #4 \__stex_notations_args_end:
4625 }
4626
4627 \cs_new_protected:Npn \__stex_notations_parse_notation_args:nnnw #1 #2 #3 #4 #5 \__stex_no
4628   \tl_if_empty:nTF{#5}{%
4629     \__stex_notations_add_last:nnnnn{#1}{#2}{#3}{#4}{#5}
4630   }{%
4631     \__stex_notations_add_next:nnnnnn{#1}{#2}{#3}{#4}{#5}{#6}
4632   }
4633 }
4634
4635 \cs_new_protected:Nn \__stex_notations_add_next:nnnnnn {
4636   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#6}
4637   \__stex_notations_parse_notation_args:nnnw #5 \__stex_notations_args_end:
4638 }
4639
4640 \cs_new_protected:Nn \__stex_notations_add_last:nnnnn {
4641   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#5}
4642   \l__stex_notations_after_tl
4643 }
4644
4645 \cs_new_protected:Nn \__stex_notations_add:nnnnn {
4646   \int_incr:N \l_tmpa_int
4647   \str_case:nn{#2}{%
4648     i {
4649       \tl_put_right:Nn \l__stex_notations_code_tl {
4650         {\l_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4651       }
4652     }
4653     b {
4654       \tl_set:Nn \l__stex_notations_pre_tl {
4655         \cs_set_eq:NN \l_stex_term_oma_or_omb:nnn \l_stex_term_omb:nnn
4656       }

```

```

4657     \tl_put_right:Nn \l__stex_notations_code_tl {
4658         {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
4659     }
4660 }
4661 a {
4662     \tl_put_right:Nn \l__stex_notations_code_tl {
4663         {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4664     }
4665 }
4666 B {
4667     \tl_set:Nn \l__stex_notations_pre_tl {
4668         \cs_set_eq:NN \_stex_term_oma_or_omb:nnn \_stex_term_omb:nnn
4669     }
4670     \tl_put_right:Nn \l__stex_notations_code_tl {
4671         {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
4672     }
4673 }
4674 }
4675 }

```

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

a/B-mode argument handling

`\argsep`

```

4676 \cs_new_protected:Nn \__stex_notations_check_aB_arg:Nn {
4677     \exp_args:Ne \cs_if_eq:NNF {\tl_head:n{#2}}
4678     \_stex_term_arg_aB:nnnnn {
4679         \msg_error:nnx{\stex}{error/assocarg}{\tl_to_str:n{#1}}
4680     }
4681 }
4682
4683 \cs_new_protected:Npn \argsep #1 #2 {
4684     \__stex_notations_check_aB_arg:Nn\argsep{#1}
4685     \stex_pseudogroup_with:nn{\_stex_term_do_aB_clist:} {
4686         \tl_set:Nn \_stex_term_do_aB_clist: {
4687             \seq_use:Nn \l_stex_aB_args_seq {#2}
4688         }
4689         #1
4690     }
4691 }

```

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

`\argmap`

```

4692 \cs_new_protected:Npn \argmap #1 #2 #3 {
4693     \__stex_notations_check_aB_arg:Nn\argmap{#1}
4694     \stex_pseudogroup_with:nn{
4695         \_stex_term_do_aB_clist:
4696         \__stex_notations_map_cs:
4697     }{
4698         \cs_set:Npn \__stex_notations_map_cs: ##1 { #2 }
4699         \tl_set:Nn \_stex_term_do_aB_clist: {
4700             \seq_clear:N \l_tmpa_seq

```

```

4701     \seq_map_inline:Nn \l_stex_aB_args_seq {
4702         \tl_if_eq:nnTF{##1}{\ellipses}{
4703             \seq_put_right:Nn \l_tmpa_seq \ellipses
4704         }{
4705             \seq_put_right:Nx \l_tmpa_seq {
4706                 \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
4707             }
4708         }
4709     }
4710     \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4711     \seq_use:Nn \l_stex_aB_args_seq {#3}
4712 }
4713 #1
4714 }
4715 }

```

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

`\argarraymap`

```

4716 \int_new:N \l_stex_notations_clist_count_int
4717 \cs_new_protected:Npn \argarraymap #1 #2 #3 #4 {
4718     \__stex_notations_check_aB_arg:Nn\argarraymap{#1}
4719     \stex_pseudogroup_with:nn{
4720         \_stex_term_do_aB_clist:
4721         \__stex_notations_map_cs:
4722     }{
4723         \cs_set:Npn \__stex_notations_map_cs: ##1 { #3 }
4724         \int_set:Nn \l_stex_notations_clist_count_int {\exp_args:No\clist_count:n{\tl_to_str:n}
4725         \tl_set:Nn \l_stex_term_do_aB_clist: {
4726             \tl_clear:N \l_tmpa_tl
4727             \int_zero:N \l_tmpa_int
4728             \seq_map_inline:Nn \l_stex_aB_args_seq {
4729                 \int_incr:N \l_tmpa_int
4730                 \int_compare:nNnT \l_tmpa_int > \l_stex_notations_clist_count_int {
4731                     \int_set:Nn \l_tmpa_int 1
4732                 }
4733                 \tl_put_right:Nx \l_tmpa_tl {
4734                     \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
4735                     \clist_item:nn{#4}\l_tmpa_int
4736                 }
4737             }
4738             \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
4739             \begin{array}{#2}
4740                 \l_tmpa_tl
4741             \end{array}
4742         }
4743         #1
4744     }
4745 }

```

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

13.8.3 Variables

4746 `\@@=stex_vars`

```

\vardef
4747 \tl_new:N \l_stex_variables_prop
4748 \bool_new:N \l__stex_vars_bind_bool
4749 \cs_new_protected:Nn \_stex_variable:nnnnnnnN {}
4750 \stex_keys_define:nnnn{vardef}{
4751   \bool_set_false:N \l__stex_vars_bind_bool
4752 }{
4753   bind .bool_set:N = \l__stex_vars_bind_bool
4754 }{symdef}
4755
4756 \stex_new_stylable_cmd:nnn {vardef} { m 0{} m} {
4757   \stex_keys_set:nn{vardef}{#2}
4758   \str_set:Nx \l_stex_macroname_str { #1 }
4759   \str_if_empty:NT \l_stex_key_name_str {
4760     \str_set:Nx \l_stex_key_name_str { #1 }
4761   }
4762
4763 \stex_symdecl_do:
4764 \_stex_symdecl_check_terms:
4765 \_stex_vars_add:
4766 \_stex_vars_macro:
4767 \stex_if_do_html:T \_stex_vars_html:
4768
4769 \int_set:Nn \l_stex_get_symbol_arity_int {\l_stex_get_symbol_arity_int}
4770 \stex_debug:nn{vardef}{Doing~\l_stex_key_name_str}
4771 \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4772 \stex_notation_parse:n{#3}
4773 \stex_if_check_terms:T{ \_stex_notation_check: }
4774 \_stex_vardecl_notation_macro:
4775 \stex_if_do_html:T {
4776   \def\comp{\_varcomp}
4777   \_stex_notation_do_html:n \l_stex_key_name_str
4778 }
4779 \group_begin:
4780 \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4781 \tl_clear:N \thisstyle
4782 \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4783 \def\thisnotation{
4784   \$\let\l_stex_current_symbol_str\thisvarname
4785   \def\comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}
4786   \_stex_notation_make_args:
4787   }$
4788 }
4789 \stex_style_apply:
4790 \group_end:\ignorespaces
4791 }{}}
4792
4793 \cs_new_protected:Nn \_stex_vars_add: {
4794   \exp_args:NNNo \exp_args:NNnx
4795   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4796     {\l_stex_macroname_str}
4797     {\l_stex_key_name_str}
4798     {\int_use:N \l_stex_get_symbol_arity_int}
4799     {\l_stex_get_symbol_args_tl}

```

```

4800   {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4801   {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4802   {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4803   \stex_invoke_symbol:
4804 }
4805 }
4806
4807 \cs_new_protected:Nn \__stex_vars_macro: {
4808   \tl_set:cx{\l_stex_mroname_str}{
4809     \stex_invoke_variable:nnnnnN
4810     {\l_stex_key_name_str}
4811     {\int_use:N \l_stex_get_symbol_arity_int}
4812     {\l_stex_get_symbol_args_tl}
4813     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4814     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4815     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4816     \stex_invoke_symbol:
4817   }
4818 }
4819
4820 \cs_new_protected:Nn \__stex_vars_html: {
4821   \stex_if_do_html:T {
4822     \hbox\bgroup\exp_args:Ne \stex_annotation_invisible:nn {
4823       \shtml:vardef = {\l_stex_key_name_str},
4824       \shtml:args = {\l_stex_key_args_str}
4825       \str_if_empty:NF \l_stex_mroname_str {
4826         \shtml:macroname={\l_stex_mroname_str}
4827       }
4828       \str_if_empty:NF \l_stex_key_assoc_str {
4829         \shtml:assocotype={\l_stex_key_assoc_str}
4830       }
4831       \str_if_empty:NF \l_stex_key_role_str {
4832         \shtml:role={\l_stex_key_role_str}
4833       }
4834       \str_if_empty:NF \l_stex_key_reorder_str {
4835         \shtml:reorderargs={\l_stex_key_reorder_str}
4836       }
4837       \bool_if:NT \l_stex_vars_bind_bool {
4838         \shtml:bind={}
4839       }
4840     }{
4841       \stex_annotation_force_break:n{
4842         \bool_set_true:N \stex_in_invisible_html_bool
4843         \tl_if_empty:NF \l_stex_key_type_tl {
4844           \stex_annotation:nn{\shtml:type={}}{$\l_stex_key_type_tl$}
4845         }
4846         \tl_if_empty:NF \l_stex_key_def_tl {
4847           \stex_annotation:nn{\shtml:definiens={}}{$\l_stex_key_def_tl$}
4848         }
4849         \tl_if_empty:NF \l_stex_key_return_tl{
4850           \exp_args:Nno \use:n{
4851             \cs_generate_from_arg_count:NNnn \l__stex_vars_cs
4852             \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4853             \tl_set:Nx \l__stex_vars_args_tl {\stex_map_args:N \stex_return_args:nn}

```

```

4854     $\\stex_annotate:nn{shtml:returnrntype={}){
4855         \\exp_after:wN \\l__stex_vars_cs \\l__stex_vars_args_tl!}$
4856     }
4857     \\tl_if_empty:NF \\l_stex_key_argtypes_clist {
4858         \\stex_annotate:nn{shtml:argtypes={}){
4859             \\_stex_annotate_force_break:n{
4860                 \\clist_map_inline:Nn \\l_stex_key_argtypes_clist {
4861                     $\\stex_annotate:nn{shtml:type={}{{##1}}$}
4862                 }
4863             }
4864         }
4865     }
4866 }
4867 }\\egroup
4868 }
4869 }
```

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

`_stex_vardecl_notation_macro:`

```

4870 \\cs_new_protected:Nn \\_stex_vardecl_notation_macro: {
4871     \\tl_set_eq:cN {l_stex_notation_
4872         \\l_stex_key_name_str _ 
4873         \\l_stex_key_variant_str _cs
4874     }\\l_stex_notation_macrocode_cs
4875     \\cs_if_exist:cF {l_stex_notation_\\l_stex_key_name_str __cs} {
4876         \\tl_set_eq:cN{l_stex_notation_\\l_stex_key_name_str __cs}
4877             \\l_stex_notation_macrocode_cs
4878     }
4879     \\tl_if_empty:NF \\l_stex_key_op_tl {
4880         \\tl_set_eq:cN {l_stex_notation_\\l_stex_key_name_str _op_
4881             \\l_stex_key_variant_str _cs}\\l_stex_key_op_tl
4882         \\cs_if_exist:cF {l_stex_notation_\\l_stex_key_name_str _op__cs} {
4883             \\cs_set_eq:cN{l_stex_notation_\\l_stex_key_name_str _op__cs}
4884             \\l_stex_key_op_tl
4885         }
4886     }
4887 }
```

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

`\stex_get_symbol_or_var:n`

`\stex_get_var:n`

```

4888 \\cs_new_protected:Nn \\_stex_vars_set_vars:nnnnnnN {
4889     \\stex_debug:nn{symbols}{Variable~#1~found}
4890     \\cs_set:Npn \\_stex_variable:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {}
4891     \\str_clear:N \\l_stex_get_symbol_mod_str
4892     \\str_set:Nn \\l_stex_get_symbol_name_str {#1}
4893     \\int_set:Nn \\l_stex_get_symbol_arity_int {#2}
4894     \\tl_set:Nn \\l_stex_get_symbol_args_tl {#3}
4895     \\tl_set:Nn \\l_stex_get_symbol_def_tl {#4}
4896     \\tl_set:Nn \\l_stex_get_symbol_type_tl {#5}
4897     \\tl_set:Nn \\l_stex_get_symbol_return_tl {#6}
4898     \\tl_set:Nn \\l_stex_get_symbol_invoke_cs {#7}
4899 }
```

4900

```

4901 \cs_new_protected:Nn \__stex_vars_get_var:n {
4902     \prop_map_inline:Nn \l_stex_variables_prop {
4903         \__stex_vars_check_var:nnnnnnnnN {#1} ##2
4904     }
4905 }
4906
4907 \cs_new_protected:Nn \__stex_vars_check_var:nnnnnnnnN {
4908     \str_if_eq:nnTF{#1}{#2} {
4909         \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}}
4910     }{
4911         \str_if_eq:nnT{#1}{#3} {
4912             \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}}
4913         }
4914     }
4915 }
4916
4917 \cs_new_protected:Nn \stex_get_var:n {
4918     \str_clear:N \l_stex_get_symbol_name_str
4919     \__stex_vars_get_var:n{#1}
4920     \str_if_empty:NT \l_stex_get_symbol_name_str {
4921         \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4922     }
4923 }
4924
4925 \cs_new_protected:Nn \stex_get_symbol_or_var:n {
4926     \str_clear:N \l_stex_get_symbol_name_str
4927     \__stex_vars_get_var:n{#1}
4928     \str_if_empty:NT \l_stex_get_symbol_name_str {
4929         \stex_debug:nn{symbols}{No~variable~#1~found}
4930         \stex_get_symbol:n{#1}
4931     }
4932 }

```

(End of definition for `\stex_get_symbol_or_var:n` and `\stex_get_var:n`. These functions are documented on page 122.)

\svar

```

4933 \NewDocumentCommand \svar {O{} m} {
4934     \group_begin:
4935         \tl_if_empty:nTF{#1} {
4936             \str_set:Nn \l_stex_current_symbol_str {#2}
4937         }{
4938             \str_set:Nn \l_stex_current_symbol_str {#1}
4939         }
4940         \bool_if:NTF \l_stex_allow_semantic_bool {
4941             \tl_clear:N \l_stex_current_term_tl
4942             \stex_term_omv:nnn{}{}{\l_varcomp{#2}}
4943         }{
4944             \msg_error:nnx{stex}{error/notallowed}{Variable}{\l_stex_current_symbol_str}
4945         }
4946     \group_end:
4947 }

```

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

13.8.4 Sequences

```
4948 <@=stex_seqs>
\varseq
4949 \stex_new_stylable_cmd:nnnn {varseq}{m O{} m m} {
4950   \stex_keys_set:nn{symdef}{#2}
4951   \str_set:Nx \l_stex_mroname_str { #1 }
4952   \str_if_empty:NT \l_stex_key_name_str {
4953     \str_set:Nx \l_stex_key_name_str { #1 }
4954   }
4955   \str_if_empty:NT \l_stex_key_args_str {
4956     \str_set:Nn \l_stex_key_args_str {1}
4957   }
4958 \stex_symdecl_do:
4959
4960 \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4961 \clist_set:Nn \l_stex_seqs_range_clist {#3}
4962 \tl_if_empty:NTF \l_stex_key_op_tl {
4963   \stex_notation_parse:n{#4}
4964   \tl_clear:N \l_stex_key_op_tl
4965 }{
4966   \stex_notation_parse:n{#4}
4967 }
4968 \stex_if_do_html:T \__stex_seqs_html:
4969 \stex_if_check_terms:T \__stex_seqs_check_terms:
4970 \__stex_seqs_add:
4971 \__stex_seqs_macro:
4972 \stex_if_check_terms:T \stex_notation_check:
4973 \stex_vardecl_notation_macro:
4974 \group_begin:
4975 \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4976 \tl_clear:N \thisstyle
4977 \str_set_eq:NN \thisnotationvariant \l_stex_key_variant_str
4978 \def \thisnotation{
4979   \$\let \l_stex_current_symbol_str \thisvarname
4980   \def \comp{\_varcomp}\exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs{}}
4981   \__stex_seqs_make_args:
4982 }$
4983 }
4984 \stex_style_apply:
4985 \group_end:\ignorespaces
4986 }{}
4987
4988 \cs_new_protected:Nn \__stex_seqs_add: {
4989   \exp_args:NNNo \exp_args:NNnx
4990   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4991     {\l_stex_mroname_str}
4992     {\l_stex_key_name_str}
4993     {\int_use:N \l_stex_get_symbol_arity_int}
4994     {\l_stex_get_symbol_args_tl}
4995     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4996     {\exp_args:No \exp_not:n \l_stex_seqs_range_clist}
4997     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4998   \stex_invoke_sequence:
```

```

4999     }
5000 }
5001
5002 \cs_new_protected:Nn \__stex_seqs_macro: {
5003   \tl_set:cx{\l_stex_mroname_str}{
5004     \stex_invoke_variable:nnnnnN
5005       {\l_stex_key_name_str}
5006       {\int_use:N \l_stex_get_symbol_arity_int}
5007       {\l_stex_get_symbol_args_tl}
5008       {\exp_args:No \exp_not:n \l_stex_key_def_tl}
5009       {\exp_args:No \exp_not:n \l_stex_seqs_range_clist}
5010       {\exp_args:No \exp_not:n \l_stex_key_return_tl}
5011       \stex_invoke_sequence:
5012   }
5013 }
5014
5015 \cs_new_protected:Nn \__stex_seqs_make_args: { \TODO }
5016 \cs_new_protected:Nn \__stex_seqs_check_terms: { \TODO }
5017
5018 \cs_new_protected:Nn \__stex_seqs_html: {
5019   \exp_args:Ne \stex_annotation_invisible:nn {
5020     \shtml:vareq = {\l_stex_key_name_str},
5021     \shtml:args = {\l_stex_key_args_str}
5022     \str_if_empty:NF \l_stex_mroname_str {,
5023       \shtml:macroname={\l_stex_mroname_str}
5024     }
5025     \str_if_empty:NF \l_stex_key_assoc_str {,
5026       \shtml:assocotype={\l_stex_key_assoc_str}
5027     }
5028     \str_if_empty:NF \l_stex_key_role_str {,
5029       \shtml:role={\l_stex_key_role_str}
5030     }
5031     \str_if_empty:NF \l_stex_key_reorder_str {,
5032       \shtml:reorderargs={\l_stex_key_reorder_str}
5033     }
5034   }{\hbox\bgroup
5035     \stex_annotation_force_break:n{
5036       \tl_if_empty:NF \l_stex_key_type_tl {
5037         \stex_annotation:nn{\shtml:type={}}{$\l_stex_key_type_tl$}
5038       }
5039       \tl_if_empty:NF \l_stex_key_def_tl {
5040         \stex_annotation:nn{\shtml:definiens={}}{$\l_stex_key_def_tl$}
5041       }
5042       \tl_if_empty:NF \l_stex_key_return_tl{
5043         \exp_args:Nno \use:n{
5044           \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs
5045           \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
5046           \tl_set:Nx \l_stex_seqs_args_tl {\l_stex_map_args:N \stex_return_args:nn}
5047           \stex_annotation:nn{\shtml:returntype={}}{
5048             $\exp_after:wN \l_stex_seqs_cs \l_stex_seqs_args_tl!$}
5049       }
5050       \tl_if_empty:NF \l_stex_key_argtypes_clist {
5051         \stex_annotation:nn{\shtml:argtypes={}}{
5052           \stex_annotation_force_break:n{

```

```

5053           \clist_map_inline:Nn \l_stex_key_argtypes_clist {
5054             \stex_annotation:nn{shml:type={}}{$##1$}
5055           }
5056         }
5057       }
5058     }
5059   }
5060 \egroup
5061 }
```

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

```

\stex_invoke_sequence:
5062 \cs_new_protected:Nn \stex_invoke_sequence: {
5063   \peek_charcode_remove:NTF ! {
5064     \peek_charcode:NTF [ \__stex_seqs_do_op:w { \__stex_seqs_do_op:w [] } ]
5065   }\__stex_seqs_do_first:
5066 }
5067
5068 \cs_new_protected:Npn \__stex_seqs_do_op:w [#1] {
5069   \cs_if_exist:cTF {l_stex_notation_\l_stex_current_symbol_str _op_#1_cs} {
5070     \stex_maybe_brackets:nn{\neginfpref}{%
5071       \stex_term_oms_or_omv:nnn{#1}{%
5072         {\use:c{l_stex_notation_\l_stex_current_symbol_str _op_#1_cs}}%
5073       }%
5074     \group_end:
5075   }{%
5076     \__stex_seqs_get_index_notation:n{#1}%
5077     \peek_charcode:NTF [ \__stex_seqs_doop_range:w { \__stex_seqs_doop_range:w[] } ]
5078   }%
5079 }
5080
5081 \cs_new_protected:Npn \__stex_seqs_doop_range:w [#1] {
5082   \bool_set_true:N \l_stex_allow_semantic_bool
5083   \clist_clear:N \l_stex_seqs_clist
5084   \clist_map_function:NN \l_stex_current_type_tl \__stex_seqs_doop_arg:n
5085     \stex_annotation:nn{
5086       shml:term=OMV,
5087       shml:head={\l_stex_current_symbol_str},
5088       shml:notationid={}
5089     }{%
5090       \l_stex_seqs_clist
5091     }%
5092   \group_end:
5093 }
5094
5095 \cs_new_protected:Nn \__stex_seqs_doop_arg:n {
5096   \tl_if_eq:nnTF{#1}{\ellipses}{%
5097     \clist_put_right:Nn \l_stex_seqs_clist {
5098       \ellipses
5099     }%
5100   }{%
5101     \clist_put_right:Nn \l_stex_seqs_clist {
5102       \exp_args:No \str_if_eq:nnTF \l_stex_current_arity_str {1}{%
```

```

5103   \group_begin:
5104     \l__stex_seqs_cs \group_end: {#1}
5105 
5106   }{
5107     \group_begin:
5108       \l__stex_seqs_cs \group_end: #1
5109     }
5110   }
5111 }
5112 }
5113 
5114 \cs_new_protected:Nn \__stex_seqs_get_index_notation:n {
5115   \cs_if_exist:cTF {l_stex_notation} {\l_stex_current_symbol_str _#1_cs} {
5116     \cs_set_eq:Nc \l__stex_seqs_cs {l_stex_notation} {\l_stex_current_symbol_str _#1_cs}
5117   }{
5118     \stex_do_default_notation:
5119     \cs_set_eq:NN \l__stex_seqs_cs \l_stex_default_notation
5120   }
5121 }
5122 
5123 
5124 \cs_new:Nn \__stex_seqs_do_first_arg:n {{\exp_not:n{## #1}}}
5125 
5126 \cs_new_protected:Nn \__stex_seqs_do_first: {
5127   \exp_args:Nnx \use:nn{
5128     \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs \cs_set:Npn
5129     \l_stex_current_arity_str }{
5130       \tl_set:Nn \exp_not:N \l__stex_seqs_first_args_tl {
5131         \int_step_function:nN \l_stex_current_arity_str \__stex_seqs_do_first_arg:n
5132       }
5133       \exp_not:N \__stex_seqs_do_first_next:
5134     }
5135   \l__stex_seqs_cs
5136 }
5137 
5138 \cs_new_protected:Nn \__stex_seqs_do_first_next: {
5139   \peek_charcode_remove:NTF ! {
5140     \peek_charcode:NTF [ \__stex_seqs_do_one:w {\__stex_seqs_do_one:w []}]
5141   }{
5142     \peek_charcode:NTF [ \__stex_seqs_do_all:w {\__stex_seqs_do_all:w []}]
5143   }
5144 }
5145 
5146 \cs_new_protected:Npn \__stex_seqs_do_one:w [#1] {
5147   \__stex_seqs_get_index_notation:n{#1}
5148   \stex_debug:nn{HERE~seq~one}{\meaning\l__stex_seqs_cs^~J\meaning\l__stex_seqs_first_args_t1}
5149   \exp_args:Nno\use:nn{\l__stex_seqs_cs\group_end:}\l__stex_seqs_first_args_t1
5150 }
5151 
5152 \cs_new_protected:Npn \__stex_seqs_do_all:w [#1] {
5153   \stex_debug:nn{HERE~seq~all}{\meaning\l__stex_seqs_first_args_t1}
5154   \exp_args:Nno\use:nn{\l_stex_invoke_notation:w [#1]}\l__stex_seqs_first_args_t1
5155 }

```

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

```
\seqmap
5156 \cs_new_protected:Npn \seqmap #1 #2 {
5157   \symuse{Metatheory?sequence-expression}{\seqmap{#1}{#2}}%\l_tmpa_t1 {#2}
5158 }
```

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

13.8.5 Expressions

```
5159 <@=stex_expr>
      Various variables:
5160 \bool_new:N \l_stex_allow_semantic_bool
5161 \bool_set_true:N \l_stex_allow_semantic_bool
5162 \tl_new:N \l_stex_current_term_tl
5163 \tl_set:Nn \l_stex_every_symbol_tl {
5164   \bool_set_false:N \l_stex_allow_semantic_bool
5165 }
```

```
\_stex_next_symbol:n
5166 \tl_new:N \l__stex_expr_reset_tl
5167 \cs_new_protected:Nn \_stex_next_symbol:n {
5168   \tl_set:Nx \l_stex_every_symbol_tl {
5169     \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5170       \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
5171         \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5172       }
5173       \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
5174         \exp_args:No \exp_not:n \l__stex_expr_reset_tl
5175       }
5176     }
5177     \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
5178       \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5179     }
5180     \exp_not:n{ \aftergroup \l__stex_expr_reset_tl }
5181     \exp_not:N \l_stex_every_symbol_tl
5182     \exp_not:n{ #1 }
5183   }
5184 }
5185 \cs_generate_variant:Nn \_stex_next_symbol:n {e}
```

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

\STEXinvisible

```
5186 \cs_new_protected:Npn \STEXinvisible #1 {
5187   \stex_annotation_invisible:n { #1 }
5188 }
```

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

Invoking Semantic Macros

```
\_stex_invoke_symbol:nnnnnnN
5189 \cs_new_protected:Nn \_stex_invoke_symbol:nnnnnnN {
5190   \bool_if:NTF \l_stex_allow_semantic_bool{
5191     \stex_if_html_backend:T{ifvmode\indent\fi}
5192     \__stex_expr_setup:nnnnnf{\_comp}{#1?#2}{#3}{#4}{#7}{#6}
5193     \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
5194     \tl_put_right:Nn \l_stex_current_redo_tl{
5195       \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_oms:nnn
5196     }
5197     #8
5198   }{
5199     \msg_error:nnxx{stex}{error/notallowed}{#1?#2}{\l_stex_current_symbol_str}
5200   }
5201 }
5202 \cs_generate_variant:Nn \_stex_invoke_symbol:nnnnnnN {ooxooooN}
5203
5204 \cs_new_protected:Nn \__stex_expr_setup:nnnnnn {
5205   \group_begin:
5206   \tl_clear:N \l_stex_return_notation_tl
5207   \tl_set:Nn \l_stex_current_redo_tl {
5208     \let \this \stex_current_this:
5209     \def\comp{#1}
5210     \def\maincomp{\comp}
5211     \str_set:Nn \l_stex_current_symbol_str {#2}
5212     \str_set:Nn \l_stex_current_arity_str{ #3 }
5213     \tl_set:Nn \l_stex_current_args_tl{ #4 }
5214     \tl_set:Nn \l_stex_current_return_tl{ #5 }
5215     \tl_set:Nn \l_stex_current_type_tl{ #6 }
5216     \tl_clear:N \l_stex_current_term_tl
5217   }
5218   \tl_put_right:Nx \l_stex_current_redo_tl {
5219     \exp_args:No \exp_not:n \l_stex_every_symbol_tl
5220   }
5221   \l_stex_current_redo_tl
5222 }
```

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

```
\_stex_invoke_variable:nnnnnn
5223 \cs_new_protected:Nn \_stex_invoke_variable:nnnnnn {
5224   \bool_if:NTF \l_stex_allow_semantic_bool{
5225     \stex_if_html_backend:T{ifvmode\indent\fi}
5226     \__stex_expr_setup:nnnnnf{\_varcomp}{#1}{#2}{#3}{#6}{#5}
5227     \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
5228     \tl_put_right:Nn \l_stex_current_redo_tl {
5229       \cs_set_eq:NN \_stex_term_oms_or_omv:nnn \_stex_term_omv:nnn
5230     }
5231     #7
5232   }{
5233     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
5234   }
5235 }
```

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

\symuse

```
5236 \cs_new_protected:Npn \symuse #1 {
5237   \stex_get_symbol:n{#1}
5238   \exp_args:Nno \use:n {\_stex_invoke_symbol:oooooN
5239   \l_stex_get_symbol_mod_str
5240   \l_stex_get_symbol_name_str
5241   {\int_use:N \l_stex_get_symbol_arity_int}
5242   \l_stex_get_symbol_args_tl
5243   \l_stex_get_symbol_def_tl
5244   \l_stex_get_symbol_type_tl
5245   \l_stex_get_symbol_return_tl}
5246   \l_stex_get_symbol_invoke_cs
5247 }
```

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

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

```
5248 \cs_new_protected:Nn \stex_invoke_symbol: {
5249   \stex_debug:nn{expressions}{Invoking-\l_stex_current_symbol_str}
5250   \mode_if_math:TF \_stex_expr_invoke_math: \_stex_expr_invoke_text:
5251 }
5252
5253 \cs_new_protected:Nn \__stex_expr_invoke_text: {
5254   \stex_debug:nn{expressions}{text-mode}
5255   \peek_charcode_remove:NTF ! \_stex_expr_invoke_op_custom:n \__stex_expr_invoke_custom:n
5256 }
5257
5258 \cs_new_protected:Nn \__stex_expr_invoke_math: {
5259   \stex_debug:nn{expressions}{math-mode}
5260   \peek_charcode_remove:NTF !
5261     % operator
5262     \peek_charcode_remove:NTF * \_stex_expr_invoke_op_custom:n {
5263       % op notation
5264       \peek_charcode:NTF [ \__stex_expr_invoke_op_notation:w {
5265         \_stex_expr_invoke_op_notation:w []
5266       }
5267     }
5268   }{
5269     \peek_charcode_remove:NTF * \_stex_expr_invoke_custom:n {
5270       % normal
5271       \peek_charcode:NTF [ \_stex_invoke_notation:w {
5272         \_stex_invoke_notation:w []
5273       }
5274     }
5275   }
5276 }
```

Notations:

```
5277 \cs_new_protected:Npn \_stex_invoke_notation:w [#1] {
5278   \stex_debug:nn{expressions}{using-notiation-#1-for-\l_stex_current_symbol_str}
5279   \cs_if_exist:cTF{\l_stex_notation_\l_stex_current_symbol_str _#1_cs}{
```

```

5280 \tl_if_empty:NTF \l_stex_current_return_tl {
5281   \stex_debug:nn{expressions}{return~empty}
5282   \use:c{l_stex_notation_}\l_stex_current_symbol_str _#1_cs}{\group_end:\_stex_eat_exclam
5283 }{
5284   \stex_debug:nn{expressions}{return?}
5285   \exp_after:wN\exp_after:wN\exp_after:wN
5286   \_\_stex_expr_invoke_return_maybe:n
5287   \exp_after:wN\exp_after:wN\exp_after:wN
5288   {\cs:w l_stex_notation_}\l_stex_current_symbol_str _#1_cs \cs_end: {}}
5289 }
5290 }{
5291   \stex_do_default_notation:
5292   \tl_if_empty:NTF \l_stex_current_return_tl {
5293     \l_stex_default_notation{\group_end:\_stex_eat_exclamation_point:}
5294   }{
5295     \exp_after:wN
5296     \_\_stex_expr_invoke_return_maybe:n
5297     \exp_after:wN
5298     {\l_stex_default_notation {}}
5299   }
5300 }
5301 }
5302
5303 \cs_new_protected:Npn \_\_stex_expr_invoke_op_notation:w [#1] {
5304   \stex_debug:nn{expressions}{op~notation~for~\l_stex_current_symbol_str}
5305   \cs_if_exist:cTF{l_stex_notation_}\l_stex_current_symbol_str _op_#1_cs){
5306     \_\_stex_maybe_brackets:nn{\neginfpref}{%
5307       \_\_stex_term_oms_or_omv:nnn{#1}{}%
5308       {\use:c{l_stex_notation_}\l_stex_current_symbol_str _op_#1_cs}}
5309     }
5310     \group_end:
5311   }{
5312     \int_compare:nNnTF \l_stex_current_arity_str = 0 {
5313       \tl_clear:N \l_stex_current_return_tl
5314       \_\_stex_invoke_notation:w [#1]
5315     }{
5316       \stex_do_default_notation_op:
5317       \_\_stex_maybe_brackets:nn{\neginfpref}{%
5318         \_\_stex_term_oms_or_omv:nnn{#1}{}%
5319         {\l_stex_default_notation}}
5320       }
5321       \group_end:
5322     }
5323   }
5324 }

```

Return:

```

5325 \cs_new_protected:Nn \_\_stex_expr_invoke_return_maybe:n {
5326   \tl_clear:N \l_\_stex_expr_return_args_tl
5327   \tl_set:Nn \l_\_stex_expr_return_this_tl {#1}
5328   \exp_args:Nnx \use:n {
5329     \cs_generate_from_arg_count:NNnn \_\_stex_expr_ret_cs
5330     \cs_set:Npn \l_stex_current_arity_str } {
5331       \int_step_function:nN \l_stex_current_arity_str \_\_stex_expr_return_arg:n

```

```

5332     \__stex_expr_invoke_return_next:
5333 }
5334 \__stex_expr_ret_cs
5335 }
5336
5337 \cs_new:Nn \__stex_expr_return_arg:n {
5338     \tl_put_right:Nn \exp_not:N \l__stex_expr_return_args_tl {{#### #1}}
5339 }
5340
5341 \cs_new_protected:Nn \__stex_expr_invoke_return_next: {
5342     \peekCharCode_remove:NTF ! {
5343         \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl \group_end:
5344     }\__stex_expr_invoke_return:
5345 }
5346
5347 \cs_new_protected:Nn \__stex_expr_invoke_return: {
5348     \tl_set:Nx \l__stex_expr_return_this_tl {
5349         \__stex_expr_return_notation:n {
5350             \exp_after:wN \exp_after:wN \exp_after:wN
5351             \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN \exp_after:wN {
5352                 \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl
5353             }
5354         }
5355     }
5356 \stex_debug:nn{return}{Notation:~\meaning\l__stex_expr_return_this_tl}
5357 \tl_put_left:Nx \l__stex_expr_return_this_tl {
5358     \group_begin:\exp_args:No \exp_not:n \l_stex_current_redo_tl
5359 }
5360 \exp_args:Nnx \use:n {
5361 \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs
5362     \cs_set:Npn \l_stex_current_arity_str } {
5363     \exp_args:No \exp_not:n \l_stex_current_return_tl
5364 }
5365 \stex_debug:nn{return}{
5366     \meaning\__stex_expr_ret_cs^~J
5367     \meaning\l__stex_expr_return_this_tl^~J
5368     \exp_args:No \exp_not:n \l__stex_expr_return_args_tl^~J
5369 }
5370 \exp_args:Nnx \use:nn {
5371     \exp_after:wN \group_end: \__stex_expr_ret_cs
5372 }{
5373     \exp_args:No \exp_not:n \l__stex_expr_return_args_tl
5374     {
5375         \exp_args:No \exp_not:n \l__stex_expr_return_this_tl
5376         \group_end:
5377     }
5378 }
5379 }
5380
5381
5382 \cs_new_protected:Nn \__stex_expr_return_notation:n {
5383     \tl_if_empty:NTF \l_stex_return_notation_tl { #1 }{
5384         \l_stex_return_notation_tl
5385     }

```

```

5386 }
5387
      Custom Notations:
5388 \cs_new_protected:Nn \__stex_expr_invoke_op_custom:n {
5389   \stex_debug:nn{expressions}{custom~op}
5390   \bool_set_true:N \l_stex_allow_semantic_bool
5391   \stex_term_oms_or_omv:nnn{}{}{\maincomp{#1}}
5392   \group_end:
5393 }
5394
5395 \int_new:N \l_stex_expr_arg_counter_int
5396 \cs_new_protected:Nn \__stex_expr_invoke_custom:n {
5397   \stex_debug:nn{custom}{custom~notation~for~\l_stex_current_symbol_str}
5398   \stex_pseudogroup:nn{
5399     \bool_set_true:N \l_stex_allow_semantic_bool
5400     \prop_gclear:N \l_stex_expr_customs_prop
5401     \seq_gclear:N \l_stex_expr_customs_seq
5402     \int_gzero:N \l_stex_expr_arg_counter_int
5403     \tl_if_empty:NF \l_stex_current_args_tl {
5404       \exp_after:wn \__stex_expr_add_prop_arg:nnw \l_stex_current_args_tl \stex_args_end:
5405       \cs_set_eq:NN \arg \__stex_expr_arg:n
5406     }
5407     \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
5408     \cs_set_eq:NN \__stex_expr_do_ab_next:nnn \stex_term_oma:nnn
5409     \stex_map_args:N \__stex_expr_check_b:nn
5410     \__stex_expr_do_ab_next:nnn{}{}{#1}
5411   }{
5412     \prop_if_exist:NT \l_stex_expr_customs_prop {
5413       \prop_gset_from_keyval:Nn \exp_not:N \l_stex_expr_customs_prop {
5414         \prop_to_keyval:N \l_stex_expr_customs_prop
5415       }
5416     }
5417     \int_gset:Nn \l_stex_expr_arg_counter_int { \int_use:N \l_stex_expr_arg_counter_int}
5418     \seq_if_exist:NT \l_stex_expr_customs_seq {
5419       \seq_gset_split:Nnn \exp_not:N \l_stex_expr_customs_seq , {
5420         \seq_use:Nn \l_stex_expr_customs_seq ,
5421       }
5422     }
5423   }
5424   % TODO check that all arguments are present
5425   \group_end:
5426 }
5427
5428 \cs_new_protected:Npn \__stex_expr_add_prop_arg:nnw #1 #2 #3\stex_args_end: {
5429   \prop_gput:Nnn \l_stex_expr_customs_prop {#1} {}
5430   \seq_gput_right:Nn \l_stex_expr_customs_seq {#2}
5431   \tl_if_empty:nF{#3}{\__stex_expr_add_prop_arg:nnw #3 \stex_args_end:}
5432 }
5433
5434 \cs_new:Nn \__stex_expr_check_b:nn {
5435   \str_case:nn #2 {
5436     b {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \stex_term_omb:nnn}
5437     B {\cs_set_eq:NN \__stex_expr_do_ab_next:nnn \stex_term_omb:nnn}

```

```

5438     }
5439 }
5440
5441 \NewDocumentCommand \__stex_expr_arg:n {s O{} m} {
5442   \IfBooleanTF #1 {
5443     \stex_annotation_invisible:n{
5444       \__stex_expr_inner:nn{#2}{#3}
5445     }
5446   }{
5447     \__stex_expr_inner:nn{#2}{#3}
5448   }
5449 }
5450
5451 \cs_new_protected:Nn \__stex_expr_inner:nn {
5452   \tl_if_empty:nTF{#1}{%
5453     \int_gincr:N \l__stex_expr_arg_counter_int
5454     \exp_args:Ne \__stex_expr_check:nTF{ \int_use:N \l__stex_expr_arg_counter_int }{
5455       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl
5456     }{
5457       \__stex_expr_inner:nn{}
5458     }{ #2 }
5459   }{
5460     \__stex_expr_check:nTF {#1}{%
5461       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5462     }{
5463       \exp_args:No \str_case:nnTF \l_tmpb_tl {
5464         {a} {
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         {B} {
5471           \exp_args:NNnx \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{%
5472             \l_tmpa_tl X
5473           }
5474           \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5475         }
5476       }{
5477         \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5478       }{
5479         \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5480       }
5481     }
5482   }
5483 }
5484
5485 \prg_new_conditional:Nnn \__stex_expr_check:n {TF} {
5486   \exp_args:NNe \prop_get:NnNTF \l__stex_expr_customs_prop {#1} \l_tmpa_tl {
5487     \tl_set:Nx \l_tmpb_tl {\seq_item:Nn \l__stex_expr_customs_seq {#1} }
5488     \tl_if_empty:NTF \l_tmpa_tl {
5489       \exp_args:NNe \prop_gput:Nnn \l__stex_expr_customs_prop
5490         { #1 }{X}
5491       \exp_args:No \str_case:nnF \l_tmpb_tl {

```

```

5492 {a}{
5493     \tl_set:Nx \l_tmpa_tl{ #1 1 }
5494 }
5495 {B}){
5496     \tl_set:Nx \l_tmpa_tl{ #1 1 }
5497 }
5498 }{
5499     \tl_set:Nx \l_tmpa_tl{ #1 1 }
5500 }
5501 \prg_return_true:
5502 }{
5503     \prg_return_false:
5504 }
5505 }{
5506     \msg_error:nxxx{\stex}{error/invalidarg}{#1}{\l_stex_current_symbol_str}
5507     \prg_return_false:
5508 }
5509 }
5510
5511 % #1 argnum #2 argmode #3 code
5512 \cs_new_protected:Nn \__stex_expr_arg_do:nnn {
5513     \stex_debug:nn{custom}{Doing~argument~#1~of~mode~#2:~\tl_to_str:n{#3}}
5514     \group_begin:
5515         \bool_set_true:N \l_stex_allow_semantic_bool
5516         \stex_term_arg:nnn {#2}{#1}{#3}
5517     \group_end:
5518 }
5519 \cs_generate_variant:Nn \__stex_expr_arg_do:nnn {oon}

```

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

Argument Handling and Annotating

```

\stex_term_arg:nnnnn
\stex_term_arg:nnn
5520 % 1: argnum 2: argmode 3: precedence 4: argname 5: code
5521 \cs_new_protected:Nn \stex_term_arg:nnnnn {
5522     \group_begin:
5523         \str_clear:N \l_stex_current_symbol_str
5524         \tl_clear:N \l_stex_current_term_tl
5525         \int_set:Nn \l_stex_notation_downprec { #3 }
5526         \bool_set_true:N \l_stex_allow_semantic_bool
5527         \stex_term_arg:nnn {#2}{#1}{#
5528             \tl_if_empty:nTF{#4}{#
5529                 #5
5530             }{
5531                 \stex_annotation:nn{mml:arg={#4}}{#5}
5532             }
5533         }
5534     \group_end:
5535 }
5536
5537 \cs_new_protected:Nn \stex_term_arg:nnn {
5538     \stex_annotation:nn{ shtml:arg={#2}, shtml:argmode={#1}}{#
5539         \stex_annotation_force_break:n{ #3 }

```

```
5540     }
5541 }
```

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

`_stex_term_arg_aB:nnnnn`

```
5542 \tl_set:Nn \_stex_expr_do_aB_clist: {
5543   \seq_use:Nn \l_stex_aB_args_seq {
5544     \mathpunct{\comp{,}{}}}
5545   }
5546 }
5547 \tl_set_eq:NN \_stex_term_do_aB_clist: \_stex_expr_do_aB_clist:
5548
5549 \int_new:N \l__stex_expr_count_int
5550 \cs_new_protected:Nn \_stex_term_arg_aB:nnnnn {
5551   \tl_if_empty:nTF{#5} {
5552     \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{}
5553   }{
5554     \seq_clear:N \l_stex_aB_args_seq
5555     \int_zero:N \l__stex_expr_count_int
5556     \clist_map_inline:nn{#5} {
5557       \_stex_expr_aB_arg:nnnnn{##1}{#1}{#2}{#3}{#4}
5558     }
5559     \_stex_term_do_aB_clist:
5560   }
5561 }
5562
5563 % 1: code 2: argnum 3: argmode 4: precedence 5: argname
5564 \cs_new_protected:Npn \_stex_expr_aB_arg:nnnnn #1 {
5565   \int_incr:N \l__stex_expr_count_int
5566   \_stex_expr_is_varseq:nTF{#1} {
5567     \exp_after:wN \exp_after:wN \exp_after:wN
5568     \_stex_expr_assoc_seq:nnnnnnn
5569     \exp_after:wN
5570     \_stex_expr_gobble:nnnnnnnn #1 \_stex_expr_end:
5571   }{
5572     \_stex_expr_is_seqmap:nTF{#1} {
5573       \exp_args:NNe \use:nn \_stex_expr_do_seqmap:nnnnn {\tl_tail:n{#1}}
5574     }{
5575       \_stex_expr_aB_simple_arg:nnnnn{#1}
5576     }
5577   }
5578 }
5579
5580 \cs_new:Npn \_stex_expr_gobble:nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8 #9 \_stex_expr_end: {
5581   {#2} #3 {#6}
5582 }
5583
5584 \cs_new_protected:Nn \_stex_expr_aB_simple_arg:nnnnn{
5585   \seq_put_right:Nx \l_stex_aB_args_seq {
5586     \_stex_term_arg:nnnnn{#2}\int_use:N\l__stex_expr_count_int}{#3}{#4}{#5}{}
5587     \exp_not:n{
5588       \tl_set_eq:NN \_stex_term_do_aB_clist: \_stex_expr_do_aB_clist:
```

```

5589         #1
5590     }
5591   }
5592 }
5593 }
5594
5595 \cs_new_protected:Nn \_stex_is_sequentialized:n {
5596   \group_begin: #1 \group_end:
5597 }

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

5598 \prg_new_conditional:Nnn \_stex_expr_is_varseq:n {TF} {
5599   \int_compare:nNnTF {\tl_count:n{#1}} = 1 {
5600     \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_head:n{#1}}
5601     \_stex_invoke_variable:nnnnNN {
5602       \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No\tl_item:nn{#1}{8}}
5603         \stex_invoke_sequence:
5604           \prg_return_true:\prg_return_false:
5605         } \prg_return_false:
5606     } \prg_return_false:
5607   }

5608
5609 \prg_new_conditional:Nnn \_stex_expr_is_seqmap:n {TF} {
5610   \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
5611     \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\seqmap}
5612       \prg_return_true:\prg_return_false:
5613     } \prg_return_false:
5614   }

Sequence variable:

5615 % 1: name 2: arity 3: clist 4: argnum 5: argmode 6: precedence 7: argname
5616 \cs_new_protected:Nn \_stex_expr_assoc_seq:nnnnnnn {
5617   \group_begin:
5618     \seq_clear:N \l_stex_aB_args_seq
5619     \_stex_expr_assoc_make_seq:nnn{#1}{#3}{#2}
5620   \exp_args:NNe \use:nn \group_end: {
5621     \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5622       \_stex_is_sequentialized:n{
5623         \_stex_term_arg:nnnn{\#4\int_use:N\l_stex_expr_count_int}{\#5}{\#6}{\#7}{\#8}
5624           \bool_set_true:N \l_stex_allow_semantic_bool
5625           \str_set:Nn \exp_not:N \l_stex_current_symbol_str
5626             {\l_stex_current_symbol_str}
5627           \tl_if_empty:NF \l_stex_current_term_tl {
5628             \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5629               \exp_args:No \exp_not:n \l_stex_current_term_tl
5630             }
5631           }
5632           \stex_annotation:nnf{
5633             shtml:term=OMV,
5634             shtml:head={#1},
5635             shtml:notationid={}
5636           }{
5637             \_stex_annotation_force_break:n{
5638               \_stex_term_do_aB_clist:

```

```

5639         }
5640     }
5641   }
5642 }
5643 }
5644 }
5645 }
5646
5647 % #1: name, #2: clist, #3:arity
5648 \cs_new_protected:Nn \__stex_expr_assoc_make_seq:nnn {
5649   \cs_if_exist:cTF{l_stex_notation_#1__cs} {
5650     \cs_set_eq:Nc \l_stex_expr_cs {l_stex_notation_#1__cs}
5651   }{
5652     \stex_do_default_notation:
5653     \cs_set_eq:NN \l_stex_expr_cs \l_stex_default_notation
5654   }
5655 \clist_map_inline:nn{#2}{%
5656   \tl_if_eq:nnTF{##1}{\ellipses}{%
5657     \seq_put_right:Nn \l_stex_aB_args_seq { ##1 }
5658   }{
5659     \int_compare:nNnTF {#3} = 1 {
5660       \tl_set:Nn \l_stex_expr_iarg_tl { ##1 }
5661     }{
5662       \tl_set:Nn \l_stex_expr_iarg_tl { ##1 }
5663     }
5664     \seq_put_right:Nx \l_stex_aB_args_seq {
5665       \group_begin:
5666       \exp_not:n {
5667         \tl_set_eq:NN \stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5668         \def\comp{\varcomp}
5669         \str_set:Nn \l_stex_current_symbol_str{#1}
5670       }
5671       \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
5672       \exp_after:wN \exp_after:wN \exp_after:wN {
5673         \exp_after:wN \l_stex_expr_cs \exp_after:wN \group_end: \l_stex_expr_iarg_tl
5674       }
5675     }
5676   }
5677 }
5678 }

\seqmap:

5679 % 1: fun 2: clist 3: argnum 4: argmode 5: precedence 6: argname
5680 \cs_new_protected:Nn \__stex_expr_do_seqmap:nnnnnn {
5681   \group_begin:
5682   \cs_set:Npn \l_tmpa_cs ##1 {##1}
5683   \seq_clear:N \l_stex_aB_args_seq
5684   \clist_map_inline:nn{#2}{%
5685     \__stex_expr_is_varseq:nTF{##1}{%
5686       \exp_after:wN
5687       \__stex_expr_varseq_in_map:nnnnnnnn ##1
5688     }{
5689       \seq_put_right:Nn \l_stex_aB_args_seq {##1}
5690     }

```

```

5691 }
5692 \seq_clear:N \l__stex_expr_old_seq
5693 \seq_map_inline:Nn \l_stex_aB_args_seq {
5694   \tl_if_eq:nnTF{##1}{\ellipses}{
5695     \seq_put_right:Nn \l__stex_expr_old_seq {##1}
5696   }
5697   % TODO \_stex_is_sequentialized:n
5698   {
5699     \exp_args:NNo \seq_put_right:Nn \l__stex_expr_old_seq {
5700       \l_tmpa_cs {##1}
5701     }
5702   }
5703 }
5704 \seq_set_eq:NN \l_stex_aB_args_seq \l__stex_expr_old_seq
5705 \exp_args:NNe \use:nn \group_end: {
5706   \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5707     \_stex_is_sequentialized:n{
5708       \_stex_term_arg:nnnnn{#4}\int_use:N\l__stex_expr_count_int}{#4}{#5}{#6}{#7}
5709       \bool_set_true:N \l_stex_allow_semantic_bool
5710       \str_set:Nn \exp_not:N \l_stex_current_symbol_str
5711         {\l_stex_current_symbol_str}
5712       \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5713         \symuse{Metatheory?sequence-map}
5714         {\exp_args:NNo \exp_not:n \l_tmpa_tl}
5715         {\_stex_term_do_aB_clist:}
5716       }
5717       \__stex_expr_do_headterm:nn{}{
5718         \_stex_term_do_aB_clist:
5719       }
5720     }
5721   }
5722 }
5723 }
5724 }
5725
5726 \cs_new_protected:Nn \__stex_expr_varseq_in_map:nnnnnnnn {
5727   \__stex_expr_assoc_make_seq:nnn{#2}{#6}{#3}
5728 }

```

(End of definition for _stex_term_arg_a

```
\_stex_term_oms_or_omv:nnn
  \_stex_term_oms:nnn
  \_stex_term_omv:nnn
    5729  \cs_new_protected:Nn \_stex_eat_exclamation_point: {
    5730    \peek_charcode_remove:NT ! {
    5731      \_stex_eat_exclamation_point:
    5732    }
    5733  }
    5734
    5735  \bool_new:N \stex_in_invisible_html_bool
    5736  \bool_set_false:N \stex_in_invisible_html_bool
    5737  \stex_if_html_backend:TF {
    5738    % 1: variant 2: intent 3: code
```

```

5739 \cs_new_protected:Nn \_stex_term_oms:nnn {
5740   \tl_if_empty:NTF \l_stex_current_term_tl {
5741     \stex_annotation:nn{
5742       shtml:term=OMID,
5743       shtml:head={\l_stex_current_symbol_str},
5744       shtml:notationid={#1},
5745     }{
5746       \_stex_annotation_force_break:n{#3}
5747     }
5748   }{
5749     \__stex_expr_do_headterm:nn{#1}{#3}
5750   }
5751 }
5752 \cs_new_protected:Nn \_stex_term_omv:nnn {
5753   \tl_if_empty:NTF \l_stex_current_term_tl {
5754     \stex_annotation:nn{
5755       shtml:term=OMV,
5756       shtml:head={\l_stex_current_symbol_str},
5757       shtml:notationid={#1}
5758     }{
5759       \_stex_annotation_force_break:n{#3}
5760     }
5761   }{
5762     \__stex_expr_do_headterm:nn{#1}{#3}
5763   }
5764 }
5765 \cs_new_protected:Nn \__stex_expr_do_headterm:nn {
5766   \bool_if:NTF \stex_in_invisible_html_bool {
5767     {\bool_set_true:N \l_stex_allow_semantic_bool
5768      \ensuremath{\l_stex_current_term_tl}}
5769   }
5770 }{
5771   \stex_annotation:nn{
5772     shtml:term=complex,
5773     shtml:head={\l_stex_current_symbol_str},
5774     shtml:notationid={#1}
5775   }{
5776     \_stex_annotation_force_break:n{
5777       \stex_annotation_invisible:nn{shtml:headterm={}}{
5778         {\bool_set_true:N \l_stex_allow_semantic_bool
5779          \ensuremath{\l_stex_current_term_tl}}
5780        }
5781      }
5782    }
5783    #2
5784  }
5785 }
5786 }
5787 }{
5788   \cs_new_protected:Nn \_stex_term_oms:nnn {#3}
5789   \cs_new_protected:Nn \_stex_term_omv:nnn {#3}
5790   \cs_new_protected:Nn \__stex_expr_do_headterm:nn { #2 }
5791 }
5792 \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
5793 \stex_if_html_backend:TF {
5794   \cs_new_protected:Nn \_stex_term_oma:nnn {
5795     \tl_if_empty:NTF \l_stex_current_term_tl {
5796       \stex_annotation:nn{
5797         shtml:term=OMA,
5798         shtml:head={\l_stex_current_symbol_str},
5799         shtml:notationid={#1}
5800       }{
5801         \_stex_annotation_force_break:n{#3}
5802       }
5803     }{
5804       \stex_annotation:nn{
5805         shtml:term=OMA,
5806         shtml:head={\l_stex_current_symbol_str},
5807         shtml:notationid={#1}
5808       }{
5809         \_stex_annotation_force_break:n{
5810           \stex_annotation_invisible:nn{shtml:headterm={}}{
5811             \bool_set_true:N \l_stex_allow_semantic_bool
5812             \l_stex_current_term_tl
5813           }
5814         #3
5815       }
5816     }
5817   }
5818 }
5819 }{
5820   \cs_new_protected:Nn \_stex_term_oma:nnn {#3}
5821 }
```

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

```
\_stex_term_omb:nnn
5822 \stex_if_html_backend:TF {
5823   \cs_new_protected:Nn \_stex_term_omb:nnn {
5824     \tl_if_empty:NTF \l_stex_current_term_tl {
5825       \stex_annotation:nn{
5826         shtml:term=OMBIND,
5827         shtml:head={\l_stex_current_symbol_str},
5828         shtml:notationid={#1}
5829       }{
5830         \_stex_annotation_force_break:n{#3}
5831       }
5832     }{
5833       \stex_annotation:nn{
5834         shtml:term=OMBIND,
5835         shtml:head={\l_stex_current_symbol_str},
5836         shtml:notationid={#1}
5837       }{
5838         \_stex_annotation_force_break:n{
```

```

5839     \stex_annotation_invisible:nn{shml:headterm={}}{{
5840         \bool_set_true:N \l_stex_allow_semantic_bool
5841         \l_stex_current_term_tl
5842     }}
5843     #3
5844 }
5845 }
5846 }
5847 }
5848 }{
5849     \cs_new_protected:Nn \_stex_term_omb:nnn { #3 }
5850 }

```

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

Automated Bracketing

```

\infprec
\neginfprec
5851 \tl_const:Nx \infprec {\int_use:N \c_max_int}
5852 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}

(End of definition for \infprec and \neginfprec. These functions are documented on page 88.)

```

```

\dobrackets
5853 \int_new:N \l_stex_notation_downprec
5854 \int_set:Nn \l_stex_notation_downprec \infprec
5855 \tl_set:Nn \l_stex_expr_left_bracket_str (
5856 \tl_set:Nn \l_stex_expr_right_bracket_str )
5857 \bool_new:N \l_stex_brackets_dones_bool
5858
5859 \cs_new_protected:Nn \_stex_maybe_brackets:nn {
5860     \bool_if:NTF \l_stex_brackets_dones_bool {
5861         \bool_set_false:N \l_stex_brackets_dones_bool
5862         #2
5863     }{
5864         \stex_debug:n{brackets}{#1}\int_eval:n \l_stex_notation_downprec?
5865         \int_compare:nNnTF { #1 } > \l_stex_notation_downprec {
5866             \%bool_if:NTF \l_stex_inpararray_bool { #2 }{
5867                 \dobrackets {
5868                     #2
5869                 }
5870             %}
5871         }{
5872             #2
5873         }
5874     }
5875 }
5876 \%RequirePackage{scalerel}
5877 \cs_new_protected:Npn \dobrackets #1 {
5878     \%ThisStyle{\if D\m@switch
5879     \% \exp_args:Nnx \use:nn
5880     \% { \exp_after:wN \left\lvert \l_stex_expr_left_bracket_str #1 }
5881     \% { \exp_not:N\right\rvert \l_stex_expr_right_bracket_str }
```

```

5883 % \else
5884 \group_begin:
5885 \%stex_pseudogroup_with:nn{\l_stex_brackets_dones_bool\l_stex_notation_downprec}%
5886   \bool_set_true:N \l_stex_brackets_dones_bool
5887   \%int_set:Nn \l_stex_notation_downprec \infprec
5888   \mathopen{\cs_if_exist:NT\l_stex_current_symbol_str\comp
5889     \l__stex_expr_left_bracket_str
5890   }
5891   #1
5892 \group_end:%
5893 \mathclose{\cs_if_exist:NT\l_stex_current_symbol_str\comp
5894   \l__stex_expr_right_bracket_str
5895 }
5896 \%fi}
5897 }

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

\withbrackets

```

5898 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
5899   \%stex_pseudogroup_with:nn{\l__stex_expr_left_bracket_str\l__stex_expr_right_bracket_str}%
5900   \tl_set:Nn \l__stex_expr_left_bracket_str { #1 }
5901   \tl_set:Nn \l__stex_expr_right_bracket_str { #2 }
5902   #3
5903 }
5904 }
```

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

\dowithbrackets

```

5905 \cs_new_protected:Npn \dowithbrackets #1 #2 #3 {
5906   \withbrackets{#1}{#2}{\dobrackets{#3}}
5907 }
```

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

Symname and Variants

```

\symname
  \sn 5908 \def\maincomp{\comp}
  \sns 5909
\Symname 5910 \stex_keys_define:nnnn{symname}%
  \Sn 5911 \tl_clear:N \l_stex_key_pre_tl
  \Sns 5912 \tl_clear:N \l_stex_key_post_tl
\symref 5913 \%tl_clear:N \l_stex_key_proot_tl
  \sr 5914 }%
\varref 5915 pre .tl_set:N = \l_stex_key_pre_tl ,
\varname 5916 post .tl_set:N = \l_stex_key_post_tl ,
\Varname 5917 root .code:n = {}%.tl_set:N = \l_stex_key_root_tl
  5918 }{}}

  5919 \NewDocumentCommand \symref { O{} m m } {
  5920   \group_begin:
  5921   \stex_keys_set:nn{symname}{#1}
  5922   \stex_get_symbol:n{#2}
```

```

5924     \_\_stex\_expr\_do\_ref:nNn{\#3}\symrefemph@uri\_\_stex\_term\_oms:nnn
5925   }
5926 \let\sr\symref
5927
5928 \NewDocumentCommand \symname { O{} m} {
5929   \group_begin:
5930   \stex_keys_set:nn{symname}{#1}
5931   \stex_get_symbol:n{#2}
5932   \_\_stex\_expr\_do\_ref:nNn{
5933     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
5934   }\symrefemph@uri\_\_stex\_term\_oms:nnn
5935 }
5936 \let\sn\symname
5937 \protected\def\sns{\symname[post=s]}
5938
5939 \NewDocumentCommand \Symname { O{} m} {
5940   \group_begin:
5941   \stex_keys_set:nn{symname}{#1}
5942   \stex_get_symbol:n{#2}
5943   \_\_stex\_expr\_do\_ref:nNn{
5944     \l_stex_key_pre_tl\exp_after:wN\stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
5945   }\symrefemph@uri\_\_stex\_term\_oms:nnn
5946 }
5947 \cs_new_protected:Nn \stex_capitalize:n {
5948   \uppercase{#1}
5949 }
5950 \let\Sn\Symname
5951 \protected\def\Sns{\Symname[post=s]}
5952
5953 \cs_new:Npn \stex_split_slash: #1/#2/#3\stex_args_end: {
5954   \tl_if_empty:nTF{#3}{
5955     #2
5956   }{
5957     \stex_split_slash: #2 / #3 \stex_args_end:
5958   }
5959 }
5960
5961 \NewDocumentCommand \varref { O{} m m} {
5962   \group_begin:
5963   \stex_keys_set:nn{symname}{#1}
5964   \stex_get_var:n{#2}
5965   \_\_stex\_expr\_do\_ref:nNn{\#3}\varempth@uri{
5966     \str_set_eq:NN \l_stex_current_symbol_str\l_stex_get_symbol_name_str
5967     \def\comp{\_varcomp}
5968     \stex_term_omv:nnn
5969   }
5970 }
5971
5972 \NewDocumentCommand \varname { O{} m} {
5973   \group_begin:
5974   \stex_keys_set:nn{symname}{#1}
5975   \stex_get_var:n{#2}
5976   \_\_stex\_expr\_do\_ref:nNn{
5977     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl

```

```

5978 } \varemph@uri{
5979   \str_set_eq:NN \l_stex_current_symbol_str \l_stex_get_symbol_name_str
5980   \def\comp{\_varcomp}
5981   \stex_term_omv:nnn
5982 }
5983 }
5984
5985 \NewDocumentCommand \Varname { O{} m} {
5986   \group_begin:
5987   \stex_keys_set:nn{symname}{#1}
5988   \stex_get_var:n{#2}
5989   \stex_expr_do_ref:nNn{
5990     \l_stex_key_pre_tl\exp_after:wN\stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
5991   } \varemph@uri{
5992     \str_set_eq:NN \l_stex_current_symbol_str \l_stex_get_symbol_name_str
5993     \def\comp{\_varcomp}
5994     \stex_term_omv:nnn
5995   }
5996 }
5997
5998
5999 \cs_new_protected:Nn \stex_expr_do_ref:nNn {
6000   \stex_if_html_backend:T{\ifvmode\indent\fi}
6001   \bool_if:NTF \l_stex_allow_semantic_bool{
6002     \str_set:Nx \l_stex_current_symbol_str
6003       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6004     \str_if_in:NnT \l_stex_get_symbol_name_str / {
6005       \str_set:Nx \l_stex_get_symbol_name_str {
6006         \exp_after:wN \stex_split_slash: \l_stex_get_symbol_name_str
6007         /\stex_args_end:
6008       }
6009     }
6010     \tl_clear:N \l_stex_current_term_tl
6011     \def\comp{\_comp}
6012     \let\compemph@uri#2
6013     #3{}{}\comp{#1}
6014   }{
6015     \msg_error:nnnx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
6016   }
6017   \group_end:
6018 }

```

(End of definition for `\symname` and others. These functions are documented on page 86.)

Highlighting

```

6019 <@=stex_notationcomps>
6020
6021   \comp
6022   \compemph@uri
6023   \compemph
6024   \defemph
6025   \defemph@uri
6026   \symrefemph
6027   \symrefemph@uri
6028   \varemph
6029   \varemph@uri

```

```

6026     \stex_if_html_backend:TF {
6027         \stex_annotate:nn { shtml:#1 = \l_stex_current_symbol_str}{ #3 }
6028     }{
6029         \exp_args:Nno #2 { #3 } \l_stex_current_symbol_str
6030     }
6031 }
6032 }
6033 }
6034
6035 \cs_new_protected:Npn \_comp {
6036     \do_comp:nNn {comp}\compemph@uri
6037 }
6038
6039 \cs_new_protected:Npn \_varcomp {
6040     \do_comp:nNn {varcomp}\varempth@uri
6041 }
6042
6043 \cs_new_protected:Npn \_defcomp {
6044     \do_comp:nNn {definiendum}\defemph@uri
6045 }
6046
6047 \cs_set_protected:Npn \comp {}
6048
6049 \cs_new_protected:Npn \compemph@uri #1 #2 {
6050     \compemph{ #1 }
6051 }
6052
6053 \cs_new_protected:Npn \compemph #1 {
6054     #1
6055 }
6056
6057 \cs_new_protected:Npn \defemph@uri #1 #2 {
6058     \defemph{#1}
6059 }
6060
6061 \cs_new_protected:Npn \defemph #1 {
6062     \ifmmode\else\expandafter\textbf\fi{#1}
6063 }
6064
6065 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
6066     \symrefemph{#1}
6067 }
6068
6069 \cs_new_protected:Npn \symrefemph #1 {
6070     \emph{#1}
6071 }
6072
6073 \cs_new_protected:Npn \varempth@uri #1 #2 {
6074     \varempth{#1}
6075 }
6076
6077 \cs_new_protected:Npn \varempth #1 {
6078     #1
6079 }

```

(End of definition for `\comp` and others. These functions are documented on page 87.)

13.9 Mathematical Structures

```
6080 <@=stex_structures>
```

```
\this
```

```
6081 \cs_new_protected:Npn \stex_current_this: {
6082   { \bool_set_true:N \l_stex_allow_semantic_bool
6083     \tl_if_empty:NTF \l_stex_current_this_tl {}{
6084       \str_set:Nx \l_stex_current_symbol_str {
6085         \l_stex_current_module_str ? \l__stex_structures_name_str
6086       }
6087       \maincomp{\l_stex_current_this_tl}
6088     }
6089   }
6090 }
6091 \let \this \stex_current_this:
```

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

```
mathstructure (env.)
```

```
6092 \stex_new_stylable_env:nnnnnnn {mathstructure}{m 0{} }{
6093   \__stex_structures_begin:nn{#1}{#2}
6094   \stex_smsmode_do:
6095 }{
6096   \stex_structural_feature_module_end:
6097   \__stex_structures_do_externals:
6098 }{}{}{}
6099
6100 \stex_keys_define:nnnn{mathstructure}){
6101   \tl_clear:N \l_stex_current_this_tl
6102   \str_clear:N \l__stex_structures_name_str
6103 }{
6104   this .tl_set:N = \l_stex_current_this_tl ,
6105   unknown .code:n = {
6106     \str_if_empty:NTF \l_keys_key_str {
6107       \str_set:Nx \l__stex_structures_name_str {\l_keys_key_tl}
6108     }{
6109       \str_set_eq:NN \l__stex_structures_name_str \l_keys_key_str
6110     }
6111   }
6112 }{}}
6113
6114 \cs_new_protected:Nn \__stex_structures_begin:nn {
6115   \stex_keys_set:nn {mathstructure}{#2}
6116   \str_if_empty:NT \l__stex_structures_name_str {
6117     \str_set:Nn \l__stex_structures_name_str {#1}
6118   }
6119   \def\comp{\_comp}
6120
6121 \exp_args:Nne \use:nn { \stex_module_add_symbol:nnnnnnN }
6122   { {#1}{\l__stex_structures_name_str}{0}{}{defed}{}
6123     \l_stex_current_module_str / \l__stex_structures_name_str-module
```

```

6124     }}
6125     {} \stex_invoke_structure:
6126 \str_set:Nx \l_stex_mroname_str {#1}
6127 \stex_execute_in_module:x{
6128     \seq_clear:c{\l_stex_structures_macros_\l_stex_current_module_str / \l_stex_structures_name_str}
6129     \seq_put_right:cn{\l_stex_structures_macros_\l_stex_current_module_str / \l_stex_structures_name_str}{#1}
6130 }
6131 \exp_args:No \stex_structural_feature_module:nn
6132     {\l_stex_structures_name_str}{structure}
6133 }
6134
6135 \stex_sms_allow_env:n{mathstructure}
6136 \stex_deactivate_macro:Nn \mathstructure {module-environments}
6137 \stex_every_module:n {\stex_reactivate_macro:N \mathstructure}
6138
6139 \cs_new_protected:Nn \__stex_structures_do_externals: {
6140     \tl_set:Nn \l_stex_structures_replace_this_tl {####1}
6141     \exp_args:No \stex_iterate_symbols:nnf{\g_stex_last_feature_str} {
6142         \__stex_structures_external_decl:nnnn{##5}{##4}{##3}{##8}
6143     }
6144 }
6145
6146 \cs_new_protected:Nn \__stex_structures_external_decl:nnnn {
6147     \%stex_debug:nn{structure}%
6148     % Generating~external~declaration~\l_stex_structures_name_str/#3-in-
6149     % \l_stex_current_module_str^{^J}
6150     % \tl_to_str:n{#1}^{^J}\tl_to_str:n{#2}^{^J}\tl_to_str:n{#4}
6151     %
6152     \%tl_set:Nn \l_stex_get_symbol_args_tl {#1}
6153     \%exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnN} {
6154         {}{\l_stex_structures_name_str/#3}{\int_eval:n{#2 + 1}}
6155         % {ii}\tl_if_empty:nF{#1}{\stex_map_args:N \__stex_structures_shift_argls:nn}
6156         % {defed}{typed}
6157         \%{#4}\stex_invoke_outer_field:
6158     }
6159
6160 \cs_new:Nn \__stex_structures_shift_argls:nn {
6161     \int_eval:n{#1+1}#2
6162 }

```

\stex_get_mathstructure:n

```

6163 \cs_new_protected:Nn \stex_get_mathstructure:n {
6164     \stex_get_mathstructure:n{#1}
6165     \str_if_empty:NT \l_stex_get_structure_module_str {
6166         \msg_error:nnn{\stex}{error/unknownstructure}{#1}
6167     }
6168 }
6169 \cs_new_protected:Nn \stex_get_mathstructure:n {
6170     \str_clear:N \l_stex_get_structure_module_str
6171     \stex_get_symbol:n{#1}
6172     \str_if_empty:NF \l_stex_get_symbol_name_str {
6173         \exp_args:No \tl_if_eq:NNT \l_stex_get_symbol_invoke_cs \stex_invoke_structure: {
6174             \str_set_eq:NN \l_stex_get_structure_module_str \l_stex_get_symbol_type_tl
6175         }

```

```

6176     }
6177 }

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

extstructure (env.)
6178 \stex_new_stylable_env:nnnnnnn {extstructure}{m 0{} m}{
6179   \seq_clear:N \l__stex_structures_imports_seq
6180   \clist_map_inline:nn{#3}{
6181     \stex_get_mathstructure:n{##1}
6182     \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6183       \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{####1}}{
6184         \seq_put_right:Nn \l__stex_structures_imports_seq{####1}
6185       }
6186     }
6187     \stex_execute_in_module:x{
6188       \seq_put_right:cn{\l_stex_structure_macros_ \l_stex_get_structure_module_str _seq}{#1}
6189     }
6190   }
6191   \__stex_structures_begin:nn{#1}{#2}
6192   \seq_map_inline:Nn\l__stex_structures_imports_seq{
6193     \stex_if_do_html:T {
6194       \hbox{\stex_annotation_invisible:nn
6195         {shtml:import={##1}} {}}
6196     }
6197     \stex_module_add_morphism:nonn
6198       {}{##1}{import} {}
6199     \stex_execute_in_module:x{
6200       \stex_activate_module:n{##1}
6201     }
6202   }
6203   \stex_smsmode_do:
6204 }{
6205   \stex_structural_feature_module_end:
6206   \__stex_structures_do_exernals:
6207 }{}{}{}

6208 \stex_sms_allow_env:n{extstructure}
6209 \stex_deactivate_macro:Nn \extstructure {module-environments}
6210 \stex_every_module:n {
6211   \stex_reactivate_macro:N \extstructure
6212 }
6213 }

6214 \cs_new:Nn \__stex_structures_extend_structure_i:NnnnnnnN {
6215   \exp_not:n{#1{#2}{#3}{#4}{#5}{defed}}{\l__stex_structures_extmod_str,#7}\exp_not:n{#8}{#9}
6216 }
6217 }

6218 \cs_new_protected:Nn \__stex_structures_extend_structure:nn {
6219   \stex_debug:nn{ext}{Extending~#1~by~#2}
6220   \str_set:Nn \l__stex_structures_extmod_str{#2}
6221   \tl_set:cx{#1}{
6222     \exp_after:wN \exp_after:wN \exp_after:wN
6223     \__stex_structures_extend_structure_i:NnnnnnnN \cs:w #1 \cs_end:
6224   }
6225 }

```

```

6226 \stex_new_stylable_env:nnnnnnn {extstructure*}{m}{
6227   \__stex_structures_new_extstruct_name:
6228   \seq_clear:N \l__stex_structures_imports_seq
6229   \stex_get_mathstructure:n{#1}
6230
6231   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6232     \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6233       \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6234     }
6235   }
6236 }
6237
6238 \stex_execute_in_module:x{
6239   \seq_map_inline:cn{l_stex_structure_macros_\l_stex_get_structure_module_str _seq} {
6240     \exp_not:N \tl_if_exist:cT{####1} {
6241       \__stex_structures_extend_structure:nn{####1}{\l_stex_current_module_str/\l__stex_st
6242     }
6243   }
6244 }
6245
6246 \exp_args:No \__stex_structures_begin:nn\l__stex_structures_exstruct_name_str{}
6247
6248 \seq_map_inline:Nn \l__stex_structures_imports_seq {
6249   \stex_if_do_html:T {
6250     \stex_annotation_invisible:nn
6251       {shtml:import= {##1}} {}
6252   }
6253   \stex_module_add_morphism:nonn
6254     {}{##1}{import}{}}
6255   \stex_execute_in_module:x{
6256     \stex_activate_module:n{##1}
6257   }
6258 }
6259
6260 \stex_smsmode_do:
6261 }{
6262   \prop_map_inline:cn{
6263     c_stex_module_ \l_stex_current_module_str _symbols_prop
6264   }{
6265     \__stex_structures_check_def:nnnnnnn ##2
6266   }
6267   \stex_structural_feature_module_end:
6268   \__stex_structures_do_externals:
6269 }{}{}{}
6270
6271 \stex_sms_allow_env:n{extstructure*}
6272 \exp_after:wN \stex_deactivate_macro:Nn
6273   \cs:w extstructure*\cs_end: {module-environments}
6274 \stex_every_module:n {
6275   \exp_after:wN \stex_reactivate_macro:N \cs:w extstructure*\cs_end:
6276 }
6277
6278 \cs_new_protected:Nn \__stex_structures_check_def:nnnnnnn {
6279   \tl_if_empty:nT{#5}{
```

```

6280     \msg_error:nnnn{stex}{error/needsdefiniens}{#2}{extstructure*}
6281 }
6282 }
6283
6284 \stex_every_module:n{ \str_set:Nn \l__stex_structures_extname_count 0}
6285
6286 \cs_new_protected:Nn \__stex_structures_new_extstruct_name: {
6287     \stex_do_up_to_module:n {
6288         \str_set:Nx \l__stex_structures_extname_count {\int_eval:n{\l__stex_structures_extname_c
6289     }
6290     \str_set:Nx \l__stex_structures_exstruct_name_str {EXTSTRUCT_\l__stex_structures_extname_c
6291 }
6292
6293
6294 Invoking structures:
6295
6296 \cs_new_protected:Nn \stex_invoke_structure: {
6297     \tl_set:Nn \l__stex_structures_set_comp_tl {\__stex_structures_set_thiscomp:}
6298     \__stex_structures_invoke_top:n {}
6299 }
6300
6301 \cs_new_protected:Nn \__stex_structures_invoke_top:n {
6302     \stex_debug:nn{structure}{
6303         invoking~structure~{\l_stex_current_type_tl}<\tl_to_str:n{#1}>
6304     }
6305     \peekCharCode:NTF [ {
6306         \__stex_structures_merge:nw{#1}
6307     }{
6308         \__stex_structures_invocation_type:n {#1}
6309         \tl_set:Nn \l__stex_structures_this_tl {}
6310         \peekCharCode_remove:NTF ! {
6311             \peekCharCode:NTF [ {
6312                 \__stex_structures_maybe_notation:w
6313             }{
6314                 \__stex_structures_maybe_notation:w []
6315             }
6316         }{
6317             \__stex_structures_invoke_this:n
6318         }
6319     }
6320
6321 \cs_new_protected:Npn \__stex_structures_merge:nw #1 [ #2 ] {
6322     \exp_args:Ne \stex_if_starts_with:nnTF {\tl_to_str:n{#2}}{comp=}{%
6323         \__stex_structures_set_customcomp: #2 \__stex_structures_end:
6324         \__stex_structures_invoke_top:n{#1}
6325     }{
6326         \exp_args:Ne \stex_if_starts_with:nnTF {\tl_to_str:n{#2}}{this=}{%
6327             \__stex_structures_set_thisnotation: #2 \__stex_structures_end:
6328             \__stex_structures_invoke_top:n{#1}
6329         }{
6330             \tl_if_empty:nTF{#1}{%
6331                 \__stex_structures_invoke_top:n{#2}
6332             }{
6333                 \tl_if_empty:nTF{#2}{%

```

```

6332     \__stex_structures_invoke_top:n{#1}
6333   }{
6334     \__stex_structures_invoke_top:n{#1,#2}
6335   }
6336 }
6337 }
6338 }
6339 }
6340
6341 \cs_new_protected:Npn \__stex_structures_set_thisnotation: this= #1 \__stex_structures_end:
6342   \tl_set:Nn \l_stex_return_notation_tl { \comp{#1} }
6343   \tl_set:Nn \l__stex_structures_set_comp_tl {}
6344 }
6345
6346 \cs_new_protected:Npn \__stex_structures_set_customcomp: comp= #1 \__stex_structures_end: {
6347   \tl_set:Nn \l__stex_structures_set_comp_tl {
6348     \__stex_structures_set_custom_comp:n{#1}
6349   }
6350   \tl_set:Nn \l_stex_return_notation_tl { \comp{} }
6351 }

```

The structure type:

```

6352 \cs_new_protected:Nn \__stex_structures_invokation_type:n {
6353   \__stex_structures_do_assign_list:n{#1}
6354   \clist_if_empty:NTF \l__stex_structures_fields_clist {
6355     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6356     = 1 {
6357       \tl_set:Nx \l__stex_structures_current_type_tl {
6358         \exp_args:No \exp_not:n \l_stex_current_redo_tl
6359         \stex_term_oms_or_omv:nnn{}{}{}
6360       }
6361     }{
6362       \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6363     }
6364   }{
6365     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
6366     = 1 {
6367       \__stex_structures_make_type:n {}
6368     }{
6369       \exp_args:No \__stex_structures_make_type:n \l_stex_current_type_tl
6370     }
6371   }
6372 }
6373
6374 \cs_new_protected:Nn \__stex_structures_do_assign_list:n {
6375   \clist_clear:N \l__stex_structures_fields_clist
6376   \tl_if_empty:nF {#1} {
6377     \keyval_parse:NNn\TODO\__stex_structures_do_assign:nn{#1}
6378   }
6379 }
6380
6381 \cs_new_protected:Nn \__stex_structures_do_assign:nn {
6382   \clist_put_right:Nn \l__stex_structures_fields_clist {{#1}{#2}}
6383 }
6384

```

```

6385 \cs_new_protected:Nn \__stex_structures_make_type:n {
6386   \tl_if_empty:nTF{#1} {
6387     \seq_clear:N \l_tmpa_seq
6388   }{
6389     \seq_set_split:Nnn \l_tmpa_seq ,{#1}
6390     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
6391     \seq_reverse:N \l_tmpa_seq
6392   }
6393   \tl_set:Nx \l__stex_structures_current_type_tl {
6394     \symuse{Metatheory?module-type~merge}{}
6395     {
6396       \exp_args:No \exp_not:n \l_stex_current_redo_tl
6397       \stex_term_oms_or_omv:nnn{}{}{}
6398     }
6399     \seq_map_function:NN \l_tmpa_seq \__stex_structures_make_mod:n
6400     \clist_if_empty:NF \l__stex_structures_fields_clist {
6401       ,\symuse{Metatheory?anonymous-record}{}
6402       \exp_args:Ne \tl_tail:n{
6403         \clist_map_function:NN \l__stex_structures_fields_clist \__stex_structures_make_
6404         }
6405       }
6406     }
6407   }
6408 }
6409 }
6410
6411 \cs_new:Nn \__stex_structures_make_mod:n {
6412   ,\symuse{Metatheory?module-type}{}
6413   \stex_annotation:nn{shtml:term=OMMOD,shtml:head={#1}}{}
6414 }
6415 }
6416
6417 \cs_new:Nn \__stex_structures_make_oml:n {
6418   \__stex_structures_make_oml:nn #1
6419 }
6420 \cs_new:Nn \__stex_structures_make_oml:nn {
6421   ,\stex_annotation:nn{
6422     shtml:term=OML,
6423     shtml:head={#1}
6424   }{
6425     \__stex_annotation_force_break:n{
6426       \stex_annotation:nn{shtml:definiens={}}{\exp_not:n{#2!}}
6427     }
6428   }
6429 }

```

Insert the structure type as a term:

```

6430 \cs_new:Nn \__stex_structures_current_type: {
6431   \%exp_args:No \exp_not:n \l_stex_current_redo_tl
6432   \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6433     \exp_args:No\exp_not:n\l__stex_structures_current_type_tl
6434   }
6435   \stex_term_oms_or_omv:nnn{}{}{}
6436 }

```

The structure type itself:

```
6437 \cs_new_protected:Npn \__stex_structures_maybe_notation:w [ #1 ] {
6438   \tl_set_eq:NN \l_stex_current_term_tl \l_stex_structures_current_type_tl
6439   \cs_if_exist:cTF{\l_stex_notation_\l_stex_current_symbol_str _#1_cs} {
6440     \use:c{\l_stex_notation_\l_stex_current_symbol_str _#1_cs}\group_end:
6441   }{
6442     \__stex_structures_make_prop:
6443     \__stex_structures_make_prop_assign:
6444     \__stex_structures_present_i:w [#1]
6445   }
6446 }
6447
6448 \cs_new_protected:Nn \__stex_structures_present: {
6449   \peek_charcode:NTF [ {
6450     \__stex_structures_present_i:w
6451   }{
6452     \__stex_structures_present:nn{}{}
6453   }
6454 }
6455
6456 \cs_new_protected:Npn \__stex_structures_present_i:w [#1] {
6457   \int_compare:nNnTF{\clist_count:n{#1}} = 1 {
6458     \__stex_structures_present:nn{}{#1}
6459   }{
6460     \peek_charcode:NTF [ {
6461       \__stex_structures_present_ii:nw{#1}
6462     }{
6463       \__stex_structures_present:nn{#1}{}
6464     }
6465   }
6466 }
6467
6468 %First: clist, second:notation-id
6469 \cs_new_protected:Npn \__stex_structures_present_ii:nw #1 [#2] {
6470   \__stex_structures_present:nn{#1}{#2}
6471 }
6472
6473 \cs_new_protected:Nn \__stex_structures_present:nn {
6474   \clist_clear:N \l_stex_structures_clist
6475   \tl_if_empty:nTF{#1} {
6476     \cs_set:Npn \l_stex_structures_cs ##1 ##2 ##3 {
6477       \tl_if_empty:nF{##2} {
6478         \__stex_structures_present_entry:nn {##1}{##3}
6479       }
6480     }
6481   }{
6482     \cs_set:Npn \l_stex_structures_cs ##1 ##2 ##3 {
6483       \exp_args:Ne \clist_if_in:nnT{\tl_to_str:n{#1}}{##1} {
6484         \__stex_structures_present_entry:nn {##1}{##3}
6485       }
6486     }
6487   }
6488 \prop_map_inline:Nn \l_stex_structures_prop {
6489   \l_stex_structures_cs {##1} ##2
```

```

6490 }
6491 \_stex_term_oms_or_omv:nnn{}{}{
6492   \exp_args:Nno \use:n{
6493     \bool_set_true:N \l_stex_allow_semantic_bool
6494     \symuse{Metatheory?mathematical-structure}[\#2]
6495   }{\l__stex_structures_clist}
6496 }\group_end:
6497 }
6498
6499 \cs_new_protected:Nn \__stex_structures_present_entry:nn {
6500   \seq_if_in:NnTF \l_stex_structures_assigned_seq {\#1} {
6501     \clist_put_right:Nn \l_stex_structures_clist {\#2!}
6502   }{
6503     \exp_args:NNe \clist_put_right:Nn \l_stex_structures_clist {
6504       \stex_next_symbol:n {
6505         \exp_args:No \exp_not:n \l_stex_structures_set_comp_tl
6506         \tl_set:Nn \exp_not:N \l_stex_structures_this_tl {
6507           \exp_args:No \exp_not:n \l_stex_structures_this_tl
6508         }
6509         \exp_not:n {
6510           \tl_set_eq:NN \this \l_stex_structures_this_tl
6511         }
6512         \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6513           \exp_args:No \exp_not:n \l_stex_return_notation_tl
6514         }
6515       }
6516       \exp_not:n{\#2!}
6517     }
6518   }
6519 }
6520
6521 \cs_new_protected:Npn \_thiscomp #1 #2 {
6522   {\tl_set:cn{\this}{\{\}}\#1\#2}\c_math_subscript_token{
6523     \group_begin:
6524       \bool_set_true:N \l_stex_allow_semantic_bool
6525       \l_stex_structures_this_tl
6526     \group_end:
6527   }
6528 }
6529 }
6530
6531 \cs_new_protected:Nn \__stex_structures_set_thiscomp: {
6532   \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_thiscomp {
6533     \edef\maincomp {\_thiscomp\comp}
6534   }
6535 }
6536
6537 \cs_new_protected:Nn \__stex_structures_set_custom_comp:n {
6538   \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
6539     \cs_set_protected:Npx \customthiscomp ##1 {
6540       \group_begin:
6541         \bool_set_true:N \l_stex_allow_semantic_bool
6542         \exp_not:n{
6543           \cs_set:Npn \l_stex_structures_comp_cs ##1 {

```

```

6544          #1
6545      }
6546      \def\maincomp
6547      {\comp}
6548      \exp_not:N\l_stex_structures_comp_{\comp{##1}}
6549      \group_end:
6550  }
6551  \def\maincomp {\_customthiscomp}
6552 }
6553 }
6554

this (of type structure):

6555 \cs_new_protected:Nn \__stex_structures_invoke_this:n {
6556   \peekCharCode_remove:NTF ! {
6557     \exp_args:Nne\use:nn{
6558       \group_end:\symuse{Metatheory?of~type}[invisible]{#1}
6559     }{
6560       {\__stex_structures_current_type:}
6561     }
6562   }{
6563     \__stex_structures_invoke_maybe_field:nn{#1}
6564   }
6565 }
6566 }

6567 \cs_new_protected:Nn \__stex_structures_invoke_maybe_field:nn {
6568   \__stex_structures_make_prop:
6569   \__stex_structures_set_this:n{#1}
6570   \tl_if_empty:nTF{#2} {
6571     \__stex_structures_make_prop_assign:
6572     \__stex_structures_present:
6573   }{
6574     \__stex_structures_invoke_field:n{#2}
6575   }
6576 }
6577 }

6578 \cs_new_protected:Nn \__stex_structures_set_this:n {
6579   \tl_if_empty:nTF{#1} {
6580     \%tl_put_right:Nn \l_stex_current_redo_tl {
6581       \%tl_clear:N \l_stex_structures_this_tl
6582     }
6583   }{
6584     \tl_set:Nx \l_stex_structures_this_tl {{%
6585       \bool_set_true:N \l_stex_allow_semantic_bool
6586       \tl_set:Nn \exp_not:N \this {
6587         \exp_args:No \exp_not:n \this
6588       }
6589       \exp_not:n{#1}
6590     }%
6591     \tl_set_eq:NN \this \l_stex_structures_this_tl
6592     \%l_stex_return_notation_tl
6593   }
6594 }
6595 }

6596

```

```

6597 \cs_new_protected:Nn \__stex_structures_get_field_name:n {
6598   \str_set:Nx \l__stex_structures_field_name_str {
6599     \exp_args:Nne \use:n {\exp_after:wN \use_i:nn \use:n}
6600     {\prop_item:Nn \l__stex_structures_prop {#1}}
6601   }
6602   \str_if_empty:NT \l__stex_structures_field_name_str {
6603     \str_set:Nn \l__stex_structures_field_name_str {#1}
6604   }
6605 }
6606
6607 \cs_new_protected:Nn \__stex_structures_invoke_field:n {
6608   \prop_if_in:NnTF \l__stex_structures_prop {#1} {
6609     \__stex_structures_get_field_name:n{#1}
6610     \tl_clear:N \l__stex_structures_more_nextsymbol_tl
6611     \%exp_args:NNe \seq_if_in:NnF \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}{
6612       \tl_set:Nx \l__stex_structures_more_nextsymbol_tl {
6613         \tl_set:Nn \exp_not:N \l__stex_structures_this_tl {
6614           \exp_args:No \exp_not:n \l__stex_structures_this_tl
6615         }
6616         \exp_not:n {
6617           \tl_set_eq:NN \this \l__stex_structures_this_tl
6618         }
6619       \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
6620         \exp_args:No \exp_not:n \l_stex_return_notation_tl
6621       }
6622       \exp_args:No \exp_not:n \l__stex_structures_set_comp_tl
6623     }
6624   }
6625 \%}
6626 \exp_args:NNx \use:nn \group_end: {
6627   \stex_next_symbol:n {
6628     \exp_args:No \exp_not:n \l__stex_structures_redo_tl
6629     \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
6630       \symuse{Metatheory?record~field}{%
6631         \symuse{Metatheory?of~type}{%
6632           \exp_args:No \exp_not:n \l__stex_structures_this_tl
6633           }{ \__stex_structures_current_type: }
6634         }{%
6635           \stex_annotation:nn{shtml:term=OML,shtml:head={\l__stex_structures_field_name_str}}
6636         }
6637       \exp_args:No \exp_not:n \l__stex_structures_more_nextsymbol_tl
6638     }
6639     \exp_not:N \use_ii:nn
6640     \prop_item:Nn \l__stex_structures_prop {#1}
6641   }
6642 }{
6643   \msg_error:nnn{\stex}{error/unknownfield}{#1}
6644 }
6645 }
6646
6647 \cs_new_protected:Nn \__stex_structures_make_prop: {
6648   \prop_clear:N \l__stex_structures_prop
6649   \seq_clear:N \l__stex_structures_seq
6650   \seq_clear:N \l__stex_structures_assigned_seq

```

```

6651   \tl_clear:N \l__stex_structures_redo_tl
6652   \__stex_structures_prop_do_decls:
6653   \__stex_structures_prop_do_notations:
6654 }
6655
6656 \cs_new_protected:Nn \__stex_structures_make_prop_assign: {
6657   \clist_if_empty:NF \l__stex_structures_fields_clist {
6658     \clist_map_inline:Nn \l__stex_structures_fields_clist {
6659       \__stex_structures_make_prop_assign:nn ##1
6660     }
6661   }
6662 }
6663
6664 \cs_new_protected:Nn \__stex_structures_make_prop_assign:nn {
6665   \prop_if_in:NnTF \l__stex_structures_prop {#1} {
6666     \exp_args:NNe \seq_put_right:Nn \l__stex_structures_assigned_seq {\tl_to_str:n{#1}}
6667     \exp_args:Nne \use:nn {\__stex_structures_make_prop_assign_replace:nnnn {#1}{#2}}
6668     {\prop_item:Nn \l__stex_structures_prop {#1}}
6669   }{
6670     \msg_error:nnn{stex}{error/unknownfieldass}{#1}
6671   }
6672 }
6673 \cs_new_protected:Nn \__stex_structures_make_prop_assign_replace:nnnn {
6674   \prop_put:Nnn \l__stex_structures_prop {#1}{##3}{#2}
6675   \tl_if_empty:nF{#3} {
6676     \tl_set:cn{#1}{ #2 }
6677     \tl_put_right:Nn \l__stex_structures_redo_tl {
6678       \tl_set:cn{#1}{ #2 }
6679     }
6680   }
6681 }
6682
6683 \cs_new_protected:Nn \__stex_structures_prop_do_decls: {
6684   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_type_tl {
6685     \tl_if_empty:nTF{##2} {
6686       \__stex_structures_do_decl_nomacro:nnnnnnnn{##3}
6687     }{
6688       \__stex_structures_do_decl:nnnnnnnn{##2}
6689     }
6690     {##1}{##3}{##4}{##5}{##6}{##7}{##8}{##9}
6691   }
6692 }
6693
6694 \cs_new_protected:Nn \__stex_structures_do_decl_nomacro:nnnnnnnn {
6695   \prop_if_in:NnF \l__stex_structures_prop {#1} {
6696     \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2?#3}}
6697     \prop_put:Nnn \l__stex_structures_prop {#1} {
6698       {}{
6699         \stex_invoke_symbol:nnnnnnN
6700         {#2}
6701         {#3}
6702         {#4}{#5}{#6}{#7}{#8}{#9}
6703       }
6704     }

```

```

6705     }
6706 }
6707
6708 \cs_new_protected:Nn \__stex_structures_do_decl:nnnnnnnn {
6709   \prop_if_in:NnF \l__stex_structures_prop {#1} {
6710     \seq_put_left:Nx \l__stex_structures_seq {\tl_to_str:n{#2##3}}
6711     \prop_put:Nnn \l__stex_structures_prop {#1}{#3} {
6712       \stex_invoke_symbol:nnnnnnnN
6713       {#2}
6714       {#3}
6715       {#4}{#5}{#6}{#7}{#8}#9
6716     }
6717   }
6718 }
6719 }
6720 \%tl_set:cn{#1}{%
6721   \stex_invoke_symbol:nnnnnnnN
6722   {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
6723 }
6724 \%tl_put_right:Nn \l__stex_structures_redo_tl {
6725   \tl_set:cn{#1}{%
6726     \stex_invoke_symbol:nnnnnnnN
6727     {#2}{#3}{#4}{#5}{#6}{#7}{#8}#9
6728   }
6729 }
6730 }
6731
6732 \cs_new_protected:Nn \__stex_structures_prop_do_notations: {
6733   \exp_args:No \stex_iterate_notations:nn\l_stex_current_type_tl{
6734     \exp_args:NNe \seq_if_in:NnT \l__stex_structures_seq {\tl_to_str:n{##1}}{
6735       \tl_put_right:Nn \l__stex_structures_redo_tl {
6736         \cs_if_exist:cF{\l_stex_notation_##1 _##2_cs} {
6737           \tl_set:cn{\l_stex_notation_##1 _##2_cs}{##4}
6738         }
6739         \cs_if_exist:cF{\l_stex_notation_##1 __cs} {
6740           \tl_set:cn{\l_stex_notation_##1 __cs}{##4}
6741         }
6742       }
6743       \cs_if_exist:cF{\l_stex_notation_##1 _##2_cs} {
6744         \tl_set:cn{\l_stex_notation_##1 _##2_cs}{##4}
6745       }
6746       \cs_if_exist:cF{\l_stex_notation_##1 __cs} {
6747         \tl_set:cn{\l_stex_notation_##1 __cs}{##4}
6748       }
6749       \tl_if_empty:nF{##5} {
6750         \tl_put_right:Nn \l__stex_structures_redo_tl {
6751           \cs_if_exist:cF{\l_stex_notation_##1 _op_##2_cs} {
6752             \tl_set:cn{\l_stex_notation_##1 _op_##2_cs}{##5}
6753           }
6754           \cs_if_exist:cF{\l_stex_notation_##1 _op__cs} {
6755             \tl_set:cn{\l_stex_notation_##1 _op__cs}{##5}
6756           }
6757         }
6758       \cs_if_exist:cF{\l_stex_notation_##1 _op_##2_cs} {

```

```

6759         \tl_set:cn{l_stex_notation_##1 _op_##2_cs}{##5}
6760     }
6761     \cs_if_exist:cF{l_stex_notation_##1 _op__cs}{
6762         \tl_set:cn{l_stex_notation_##1 _op__cs}{##5}
6763     }
6764 }
6765 }
6766 }
6767 }

\usestructure
6768 \cs_new_protected:Npn \usestructure #1 {
6769     \stex_get_mathstructure:n{#1}
6770     \seq_clear:N \l_stex_structures_imports_seq
6771     \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6772         \exp_args:Ne \stex_if_module_exists:nT{\tl_to_str:n{##1}}{
6773             \seq_put_right:Nn \l_stex_structures_imports_seq{##1}
6774         }
6775     }
6776     \seq_map_inline:Nn \l_stex_structures_imports_seq {
6777         \stex_if_do_html:T {
6778             \hbox{\stex_annotation_invisible:nn
6779                 {shtml:usemodule=##1} {}}
6780         }
6781         \stex_activate_module:n {##1}
6782     }
6783 }

```

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

13.10 Statements

```

6784 <@=stex_statements>
6785
6786 \stex_keys_define:nnnn{statement}{
6787     \str_clear:N \l_stex_key_name_str
6788     \str_clear:N \l_stex_key_mroname_str
6789     \clist_clear:N \l_stex_key_for_clist
6790     \str_clear:N \l_stex_key_args_str
6791     \tl_clear:N \l_stex_key_type_tl
6792     \tl_clear:N \l_stex_key_def_tl
6793     \tl_clear:N \l_stex_key_return_tl
6794     \clist_clear:N \l_stex_key_argtypes_clist
6795 }{
6796     name .str_set:N = \l_stex_key_name_str ,
6797     for .clist_set:N = \l_stex_key_for_clist ,
6798     macro .str_set:N = \l_stex_key_mroname_str ,
6799     % start .str_set:N = \l_stex_key_title_str , % TODO remove
6800     type .tl_set:N = \l_stex_key_type_tl ,
6801     judgment .code:n = {},
6802     from .code:n= {}, % TODO remove
6803     to .code:n={} % TODO remove
6804 }{id,title,style,symargs}

```

```

\stex_new_statement:nn
6805 \cs_new_protected:Npn \stex_do_for_list: {
6806   \seq_clear:N \l_stex_fors_seq
6807   \clist_map_inline:Nn \l_stex_key_for_clist {
6808     \exp_args:N\stex_get_symbol:n{\tl_to_str:n{##1}}
6809     \seq_put_right:Nx \l_stex_fors_seq
6810       {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6811   }
6812 }
6813
6814 \cs_new_protected:Nn \__stex_statements_setup:nn {
6815   \str_if_empty:NF \l_stex_key_macroname_str {
6816     \str_if_empty:NT \l_stex_key_name_str {
6817       \str_set_eq:NN \l_stex_key_name_str \l_stex_key_macroname_str
6818     }
6819   }
6820   \stex_do_for_list:
6821   \str_if_empty:NF \l_stex_key_name_str {
6822     \__stex_statements_force_id:
6823     \seq_put_right:Nx \l_stex_fors_seq {
6824       \l_stex_current_module_str ? \l_stex_key_name_str
6825     }
6826     \str_set_eq:NN \l_stex_macroname_str \l_stex_key_macroname_str
6827     \str_set:Nn \l_stex_key_role_str {#2}
6828     \stex_symdecl_do:
6829     \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnN} {
6830       {\l_stex_key_macroname_str}{\l_stex_key_name_str}
6831       {\int_use:N \l_stex_get_symbol_arity_int}
6832       {\l_stex_get_symbol_args_tl}
6833       {#1}{}{}\stex_invoke_symbol:
6834     }
6835     \stex_if_do_html:T \__stex_symdecl_html:
6836   }
6837   \str_clear:N \l__stex_statements_uri_str
6838   \str_if_empty:NTF \l_stex_key_name_str {
6839     \stex_debug:nn{statement}{no~name}
6840     \int_compare:nNnTF {\seq_count:N \l_stex_fors_seq} = 1 {
6841       \str_set:Nx \l__stex_statements_uri_str {\seq_item:Nn \l_stex_fors_seq 1}
6842       \stex_debug:nn{statement}{for:~\l__stex_statements_uri_str}
6843     }{
6844       \stex_debug:nn{statement}{no~for}
6845     }
6846   }{
6847     \str_set:Nx \l__stex_statements_uri_str {\l_stex_current_module_str ? \l_stex_key_name_s
6848     \stex_debug:nn{statement}{name:~\l__stex_statements_uri_str}
6849   }
6850 }
6851
6852 \cs_new:Nn \__stex_statements_html_keyvals:nn {
6853   shtml:#1={},
6854   shtml:inline={#2},
6855   \seq_if_empty:NF \l_stex_fors_seq {
6856     shtml:fors={\seq_use:Nn \l_stex_fors_seq ,}
6857   }

```

```

6858 \str_if_empty:NF \l_stex_key_id_str {
6859     shtml:id={\stex_uri_use:N \l_stex_current_doc_uri ? \l_stex_key_id_str}
6860 }
6861 \clist_if_empty:NF \l_stex_key_style_clist {
6862     shtml:styles={\l_stex_key_style_clist}
6863 }
6864 }
6865
6866 \cs_new_protected:Nn \stex_new_statement:nnn {
6867     \stex_new_stylable_env:nnnnnnn {\#1}{0{} }{
6868         \stex_keys_set:nn{statement}{##1}
6869         #3
6870
6871         \stex_if_smsmode:F {
6872             \exp_args:Nne \begin{stex_annotation_env} {
6873                 \__stex_statements_html_keyvals:nn{\#1}{false}
6874             }
6875             \tl_set_eq:NN \thistitle \l_stex_key_title_tl
6876             \str_set_eq:NN \thisname \l_stex_key_name_str
6877             \clist_set_eq:NN \thisfor \l_stex_key_for_str
6878             \stex_if_html_backend:TF {
6879                 \noindent
6880                 \stex_annotation:nn{shtml:statementtitle={}}{\stex_annotation_force_break:n\l_stex_key_}
6881             }
6882             \stex_style_apply:
6883         }
6884         \stex_do_id:
6885         \stex_smsmode_do:
6886     }{
6887         \stex_if_smsmode:F {
6888             \stex_if_html_backend:F \stex_style_apply:
6889             \end{stex_annotation_env}
6890         }
6891     }{}{}{s}
6892     \stex_sms_allow_env:n{s#1}
6893
6894     \tl_if_empty:nF{#2} {
6895         \exp_after:wN \NewDocumentCommand \cs:w inline#2\cs_end: { 0{} m} {
6896             \group_begin:
6897                 \stex_keys_set:nn{statement}{##1}
6898                 #3
6899                 \stex_do_id:
6900                 \stex_if_smsmode:F{
6901                     \exp_args:Ne \stex_annotation:nn{\__stex_statements_html_keyvals:nn{\#1}{true}}{
6902                         \stex_annotation_force_break:n{##2}
6903                     }
6904                 }
6905                 \group_end:
6906                 \stex_smsmode_do:
6907             }
6908             \exp_after:wN \stex_sms_allow_escape:N\cs:w inline#2\cs_end:
6909         }
6910     }
6911

```

```

6912 \cs_new_protected:Nn \__stex_statements_setup_def: {
6913   \stex_if_smsmode:F{
6914     \seq_map_inline:Nn \l_stex_fors_seq {
6915       \stex_ref_new_sym_target:n{##1}
6916     }
6917   }
6918   \stex_reactivate_macro:N \definiendum
6919   \stex_reactivate_macro:N \defnotation
6920   \stex_reactivate_macro:N \defname
6921   \stex_reactivate_macro:N \Defname
6922   \stex_reactivate_macro:N \varbind
6923 }
6924
6925 \cs_new_protected:Nn \__stex_statements_force_id: {
6926   \str_if_empty:NT \l_stex_key_id_str {
6927     \stex_ref_new_id:n{}
6928     \str_set_eq:NN \l_stex_key_id_str \l__stex_refs_str
6929   }
6930 }
6931
6932 \stex_new_statement:nnn{definition}{def} {
6933   \__stex_statements_force_id:
6934   \__stex_statements_setup:nn{}{}
6935   \__stex_statements_setup_def:
6936   \stex_reactivate_macro:N \definiens
6937 }
6938 \stex_new_statement:nnn{assertion}{ass} {
6939   \__stex_statements_setup:nn{}{assertion}
6940   \stex_if_smsmode:F{
6941     \seq_map_inline:Nn \l_stex_fors_seq {
6942       \stex_ref_new_sym_target:n{##1}
6943     }
6944   }
6945   \stex_reactivate_macro:N \varbind
6946   \stex_reactivate_macro:N \conclusion
6947   \stex_reactivate_macro:N \premise
6948   \stex_reactivate_macro:N \definiendum
6949   \stex_reactivate_macro:N \defnotation
6950   \stex_reactivate_macro:N \defname
6951   \stex_reactivate_macro:N \Defname
6952 }
6953 \stex_new_statement:nnn{example}{ex} {\__stex_statements_setup:nn{}{example}}
6954 \stex_new_statement:nnn{paragraph}{} {
6955   \clist_if_in:NnTF \l_stex_key_style_clist {symdoc} {
6956     \__stex_statements_force_id:
6957     \__stex_statements_setup:nn{}{}
6958     \__stex_statements_setup_def:
6959   }{
6960     \__stex_statements_setup:nn{}{}
6961   }
6962 }

```

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

definiendum

```
6963 \cs_new_protected:Nn \__stex_statements_do_deref:nn {
6964   \stex_if_html_backend:T{\ifvmode\indent\fi}
6965   \group_begin:
6966   \stex_get_symbol:n{#1}
6967   \bool_if:NTF \l_stex_allow_semantic_bool{
6968     \str_set:Nx \l_stex_current_symbol_str
6969     {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
6970     \str_if_in:NnT \l_stex_get_symbol_name_str / {
6971       \str_set:Nx \l_stex_get_symbol_name_str {
6972         \exp_after:wN \_stex_split_slash: \l_stex_get_symbol_name_str
6973         /\_stex_args_end:
6974       }
6975     }
6976     \exp_args:No \stex_ref_new_sym_target:n \l_stex_current_symbol_str
6977     \def\comp{\_defcomp}
6978     \stex_annotate:nn{shtml:definiendum=\l_stex_current_symbol_str}{\comp{#2}}
6979   }{
6980     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
6981   }
6982   \group_end:
6983 }
6984
6985 \NewDocumentCommand \defnotation{ m } {
6986   \l_stex_next_symbol:n { \def\comp{\_defcomp}}#1
6987 }
6988 \stex_deactivate_macro:Nn \defnotation {definition~environments}
6989
6990 \NewDocumentCommand \definiendum { O{} m m} {
6991   \stex_keys_set:nn{symname}{ #1 }
6992   \__stex_statements_do_deref:nn{#2}{#3}
6993 }
6994 \stex_deactivate_macro:Nn \definiendum {definition~environments}
6995
6996 \NewDocumentCommand \defname { O{} m } {
6997   \stex_keys_set:nn{symname}{#1}
6998   \__stex_statements_do_deref:nn{#2}(
6999     \l_stex_key_pre_tl\l_stex_get_symbol_name_str\l_stex_key_post_tl
7000   )
7001 }
7002 \stex_deactivate_macro:Nn \defname {definition~environments}
7003
7004 \NewDocumentCommand \Defname { O{} m } {
7005   \stex_keys_set:nn{symname}{#1}
7006   \__stex_statements_do_deref:nn{#2}(
7007     \l_stex_key_pre_tl\exp_after:wN\_stex_capitalize:n\l_stex_get_symbol_name_str\l_stex_key
7008   )
7009 }
7010 \stex_deactivate_macro:Nn \Defname {definition~environments}
7011
7012
7013 \NewDocumentCommand \defniens { O{} m }{
7014   \group_begin:
7015   \str_clear:N \l_stex_get_symbol_name_str
```

```

7016 \tl_if_empty:nF {#1} {
7017   \stex_get_symbol:n { #1 }
7018   \str_set:Nx \l__stex_statements_uri_str
7019     {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7020 }
7021 \str_if_empty:NT \l__stex_statements_uri_str {
7022   \msg_error:nn{stex}{error/definiensfor}
7023 }
7024 \stex_debug:nn{definiens}{Checking-\l__stex_statements_uri_str}
7025
7026 \exp_args:No \_stex_add_definiens:nn \l__stex_statements_uri_str{#2}
7027
7028 \group_end:
7029 \stex_smsmode_do:
7030 }
7031 \stex_deactivate_macro:Nn \definiens {definition-environments}
7032 \stex_sms_allow_escape:N \definiens
7033
7034 \cs_new_protected:Nn \_stex_add_definiens:nn {
7035   \exp_args:Nno \stex_str_if_starts_with:nnT{#1} \l_stex_current_module_str {
7036     \prop_map_inline:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop} {
7037       \stex_debug:nn{definiens}{#1 == \l_stex_current_module_str?##1}
7038       \str_if_eq:not {#1} {\l_stex_current_module_str?##1} {
7039         \prop_map_break:n{\_stex_add_definiens_inner:nnnnnnnn ##2}
7040       }
7041     }
7042   }
7043 \stex_if_smsmode:F{
7044   \stex_annotation:nn{ shtml:definiens={#1}}{
7045     #2 \%_stex_annotation_force_break:n{ #2 }
7046   }
7047 }
7048 }
7049
7050 \cs_new_protected:Nn \_stex_add_definiens_inner:nnnnnnnn {
7051   \stex_debug:nn{definiens}{Adding-definiens-to-\l_stex_current_module_str?#2}
7052   \prop_gput:cnn{c_stex_module_\l_stex_current_module_str _symbols_prop}
7053     {#2}{##1}{#2}{#3}{#4}{defed}{#6}{#7}{#8}}
7054 }
7055
7056 \NewDocumentCommand \varbind {m} {
7057   \clist_map_inline:nn {#1} {
7058     \stex_get_var:n {##1}
7059     \stex_if_do_html:T {
7060       \stex_annotation_invisible:nn {shtml:bind=\l_stex_get_symbol_name_str}{}
7061     }
7062   }
7063 }
7064 \stex_deactivate_macro:Nn \varbind {definition-or-assertion-environments}
7065
7066 \NewDocumentCommand \conclusion { O{} m} {
7067   \group_begin:
7068   \str_clear:N \l_stex_get_symbol_name_str
7069   \tl_if_empty:nF {#1} {

```

```

7070     \stex_get_symbol:n { #1 }
7071     \str_set:Nx \l__stex_statements_uri_str
7072         {\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7073     }
7074     \str_if_empty:NT \l__stex_statements_uri_str {
7075         \msg_error:nn{stex}{error/conclusionfor}
7076     }
7077     \stex_annotate:nn{ shtml:conclusion=\l__stex_statements_uri_str}{
7078         #2 \%_stex_annotate_force_break:n{ #2 }
7079     }
7080     \group_end:
7081 }
7082 \stex_deactivate_macro:Nn \conclusion {assertion-environments}
7083
7084 \NewDocumentCommand \premise {O{} m} {
7085     \tl_if_empty:nF {#1} {
7086         \stex_debug:nn{Here:}{Variable~#1}
7087         \exp_args:Nne\use:nn{\vardef}{{v#1}[name=#1]{#1}}
7088     }
7089     \stex_annotate:nn{shtml:premise={#1}}{#2}
7090 }
7091 \stex_deactivate_macro:Nn \premise {assertion-environments}

```

(End of definition for `definiendum`. This function is documented on page 97.)

13.11 Proofs

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

```

7092 <@=stex_proof>
7093 \stex_keys_define:nnnn{ spf }{
7094 \tl_clear:N \l_stex_key_for_clist
7095 \tl_clear:N \l_stex_key_from_tl
7096 \tl_set_eq:NN \l_stex_key_proofend_tl \__stex_proof_proof_box_tl
7097 \tl_clear:N \l_stex_key_continues_tl
7098 \tl_clear:N \l_stex_key_term_tl
7099 \tl_clear:N \l_stex_key_functions_tl
7100 \tl_clear:N \l_stex_key_method_tl
7101     \bool_set_false:N \l_stex_key_hide_bool
7102 }{
7103     for      .clist_set:N = \l_stex_key_for_clist ,
7104     from     .tl_set:N   = \l_stex_key_from_tl ,
7105     proofend .tl_set:N   = \l_stex_key_proofend_tl,
7106     continues .tl_set:N = \l_stex_key_continues_tl,
7107     functions .tl_set:N = \l_stex_key_functions_tl,
7108     term     .tl_set:N   = \l_stex_key_term_tl,
7109     method    .tl_set:N = \l_stex_key_method_tl,
7110     hide      .bool_set:N = \l_stex_key_hide_bool
7111 }{id,style,title}
7112
7113 \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.

```

7114 \intarray_new:Nn\l__stex_proof_counter_intarray{50}
7115 \cs_new_protected:Npn \__stex_proof_insert_number: {
7116   \int_set:Nn \l_tmpa_int {1}
7117   \bool_while_do:nn {
7118     \int_compare_p:nNn {
7119       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7120     } > 0
7121   }{
7122     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .
7123     \int_incr:N \l_tmpa_int
7124   }
7125 }
7126 \cs_new_protected:Nn \__stex_proof_number_as_string:N {
7127   \str_clear:N #1
7128   \int_set:Nn \l_tmpa_int {1}
7129   \bool_while_do:nn {
7130     \int_compare_p:nNn {
7131       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7132     } > 0
7133   }{
7134     \str_put_right:Nx #1 {\intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .}
7135     \int_incr:N \l_tmpa_int
7136   }
7137 }
7138
7139 \cs_new_protected:Npn \__stex_proof_inc_counter: {
7140   \int_set:Nn \l_tmpa_int {1}
7141   \bool_while_do:nn {
7142     \int_compare_p:nNn {
7143       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7144     } > 0
7145   }{
7146     \int_incr:N \l_tmpa_int
7147   }
7148   \int_compare:nNnf \l_tmpa_int = 1 {
7149     \int_decr:N \l_tmpa_int
7150   }
7151   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int {
7152     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int + 1
7153   }
7154 }
7155
7156 \cs_new_protected:Npn \__stex_proof_add_counter: {
7157   \int_set:Nn \l_tmpa_int {1}
7158   \bool_while_do:nn {
7159     \int_compare_p:nNn {
7160       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7161     } > 0

```

```

7162   }{
7163     \int_incr:N \l_tmpa_int
7164   }
7165   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 1 }
7166 }
7167
7168 \cs_new_protected:Npn \__stex_proof_remove_counter: {
7169   \int_set:Nn \l_tmpa_int {1}
7170   \bool_while_do:nn {
7171     \int_compare_p:nNn {
7172       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7173     } > 0
7174   }{
7175     \int_incr:N \l_tmpa_int
7176   }
7177   \int_decr:N \l_tmpa_int
7178   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 0 }
7179 }

```

`spfsketch`

```

7180 \newenvironment{spfsketchenv}{}{}
7181 \stex_new_stylable_cmd:nnnn{spfsketch}{0{}} m}{\par
7182   \begin{spfsketchenv}
7183     \stex_keys_set:nn{spf}{#1}
7184     \stex_do_for_list:
7185     \stex_do_id:
7186     \exp_args:Ne \stex_annotate:nn{
7187       shtml:proofsketch={
7188         \seq_if_empty:NF \l_stex_fors_seq {
7189           \seq_use:Nn \l_stex_fors_seq ,
7190         }
7191       }
7192     }{
7193       \stex_style_apply:
7194       #2
7195     }
7196   \end{spfsketchenv}
7197 }{
7198   \noindent\emph{\spfsketchenv\autorefname :}-
7199 }

```

(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

```

7200 \tl_set:Nn \__stex_proof_proof_box_tl {
7201   \ltx@ifpackageloaded{amssymb}{$\square$}{
7202     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
7203   }
7204 }
7205
7206 \tl_set:Nn \sproofend {
7207   \tl_if_empty:NF \l_stex_key_proofend_tl {
7208     \hfil\null\nobreak\hfill\l_stex_key_proofend_tl\par\smallskip

```

```

7209     }
7210 }

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

\stexcommentfont
7211 \cs_new_protected:Npn \stexcommentfont {
7212   \small\itshape
7213 }

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

sproof (env.)
7214 \cs_new_protected:Nn \__stex_proof_start_list:n {
7215   \begin{list}{}{%
7216     \setlength\topsep{0pt}
7217     \setlength\parsep{0pt}
7218     \setlength\rightmargin{0pt}
7219   }\item[#1]
7220 }
7221 \cs_new_protected:Nn \__stex_proof_end_list: {
7222   \end{list}
7223 }

7224 \cs_new_protected:Nn \__stex_proof_html: {
7225   \stex_annotation_invisible:n{\hbox{
7226     \tl_if_empty:NF \l_stex_key_term_tl {
7227       $\stex_annotation:nn{shtml:proofterm={}}{\l_stex_key_term_tl}$
7228     }
7229     \tl_if_empty:NF \l_stex_key_method_tl {
7230       \stex_annotation:nn{shtml:proofmethod={}}{\l_stex_key_method_tl}
7231     }
7232   }{}}
7233 }
7234 }

7235 \cs_new_protected:Nn \__stex_proof_html_env:n {
7236   \exp_args:Nne \begin{stex_annotation_env}{%
7237     shtml:#1={%
7238       \seq_if_empty:NF \l_stex_fors_seq {
7239         \seq_use:Nn \l_stex_fors_seq ,
7240       }
7241     }
7242     \bool_if:NT \l_stex_key_hide_bool {,
7243       shtml:proofhide=true
7244     }
7245   }{%
7246     \__stex_proof_html:
7247   }
7248 }

7249 \bool_set_false:N \l__stex_proof_in_spfblock_bool
7250 \cs_new_protected:Nn \__stex_proof_begin_proof:nn {\par
7251   \intarray_gzero:N \l__stex_proof_counter_intarray
7252   \intarray_gset:Nnn \l__stex_proof_counter_intarray 1 1
7253   \stex_keys_set:nn{spfsteps}{#1}
7254   \stex_do_for_list:

```

```

7256   \stex_if_do_html:T {
7257     \__stex_proof_html_env:n{proof}
7258   }
7259   \seq_map_inline:Nn \l_stex_fors_seq {
7260     \stex_debug:nn{definiens}{Adding-definiens-to-##1}
7261     \_stex_add_definiens:nn {##1}{\STEXinvisibl{proven}}
7262   }
7263   \stex_style_apply:
7264   \_stex_do_id:
7265   \stex_reactivate_macro:N \subproof
7266   \stex_reactivate_macro:N \spfstep
7267   \stex_reactivate_macro:N \conclude
7268   \stex_reactivate_macro:N \assumption
7269   \stex_reactivate_macro:N \eqstep
7270   \stex_reactivate_macro:N \yield
7271   \stex_reactivate_macro:N \spfblock
7272   \stex_reactivate_macro:N \spfjust
7273   \stex_annotation:nn{html:prooftitle={}}{#2}
7274   \stex_if_do_html:T{
7275     \begin{stex_annotation_env}{shtml:proofbody={}}
7276   }
7277 }
7278 \stex_new_stylable_env:nnnnnnn{proof}{0{} m}{
7279   \__stex_proof_begin_proof:nn{#1}{#2}
7280   \bool_set_true:N\l_stex_proof_in_spfblock_bool\__stex_proof_start_list:n{}
7281   \group_begin:\stexcommentfont
7282 }{
7283   \stex_style_apply:
7284   \stex_if_do_html:T{\end{stex_annotation_env}\end{stex_annotation_env}}
7285 }{
7286   \emph{\sproofautorefname :}~
7287 }{
7288   \sproofend
7289 }{s}
7290 \AddToHook{env/sproof/end}{
7291   \bool_if:NT\l_stex_proof_in_spfblock_bool {
7292     \group_end:\__stex_proof_end_list:
7293   }
7294 }
7295 \stex_new_stylable_env:nnnnnnn{proof*}{0{} }{
7296   \__stex_proof_begin_proof:nn{#1}{}
7297   \bool_set_false:N\l_stex_proof_in_spfblock_bool
7298 }{
7299   \stex_style_apply:
7300   \stex_if_do_html:T{\end{stex_annotation_env}\end{stex_annotation_env}}
7301 }{
7302   \emph{Proof:}~
7303 }{
7304   \sproofend
7305 }{s}

subproof (env.)
7307 \str_set_eq:NN \subproofautorefname \spfstepautorefname

```

```

7308 \stex_new_stylable_env:nnnnnnn{subproof}{s 0{} m}{\par
7309   \stex_keys_set:nn{spf}{#2}
7310   \stex_do_for_list:
7311   \stex_if_do_html:T {
7312     \__stex_proof_html_env:n{subproof}
7313   }
7314   \seq_map_inline:Nn \l_stex_fors_seq {
7315     \stex_debug:nn{definiens}{Adding-definiens-to-##1}
7316     \stex_add_definiens:nn {##1}{\STEXinvisibl{proven}}
7317   }
7318
7319   \IfBooleanTF #1 {
7320     \stex_style_apply:
7321     \str_if_empty:NF \l_stex_key_id_str {
7322       \__stex_proof_number_as_string:N \@currentlabel
7323       \str_set:Nx \@currentHref{subproof.\@currentlabel}
7324       \stex_do_id:
7325     }
7326     \bool_set_false:N \l__stex_proof_in_spfblock_bool
7327     \stex_annotation:nn{shtml:prooftitle={}}{#3}
7328   }{
7329     \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7330       \str_if_empty:NF \l_stex_key_id_str {
7331         \__stex_proof_number_as_string:N \@currentlabel
7332         \str_set:Nx \@currentHref{subproof.\@currentlabel}
7333         \stex_do_id:
7334       }
7335       \__stex_proof_start_list:n\__stex_proof_insert_number:
7336         \stex_annotation:nn{shtml:prooftitle={}}{#3}
7337         \__stex_proof_add_counter:
7338         \stex_style_apply:
7339       }{
7340         \stex_annotation:nn{shtml:prooftitle={}}{#3}
7341         \stex_style_apply:
7342         \stex_do_id:
7343       }
7344     }
7345     \stex_if_do_html:T{
7346       \begin{stex_annotation_env}{shtml:proofbody={}}
7347     }
7348     \bool_if:NT \l__stex_proof_in_spfblock_bool {\group_begin:\stexcommentfont}
7349   }{
7350     \stex_style_apply:
7351     \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_inc_counter:
7352     \stex_if_do_html:T{\end{stex_annotation_env}}
7353     \bool_if:NT\l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
7354     \stex_if_do_html:T{\end{stex_annotation_env}}
7355     \aftergroup \__stex_proof_inblock_restore:
7356   }{}{}{}
7357 \AddToHook{env/subproof/before}{
7358   \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7359 }
7360 \AddToHook{env/subproof/end}{
7361   \bool_if:NT\l__stex_proof_in_spfblock_bool {

```

```

7362     \group_end:\__stex_proof_remove_counter:
7363     \%__stex_proof_end_list:
7364   }
7365 }
7366 \stex_deactivate_macro:Nn \subproof {sproof~environments}
7367
7368 \cs_new_protected:Nn \__stex_proof_inblock_restore: {
7369   \bool_if:NT\l__stex_proof_in_spfblock_bool {
7370     \group_begin:\stexcommentfont
7371   }
7372 }

\spfstep
\conclude
\assumption
\have
\eqstep
7373 \stex_keys_define:nnnn { spfsteps } {
7374   \clist_clear:N \l__stex_key_for_clist
7375   \str_clear:N \l__stex_key_name_str
7376   \tl_clear:N \l__stex_key_method_tl
7377   \tl_clear:N \l__stex_key_term_tl
7378   \str_set_x:N = \l__stex_key_name_str
7379 }{
7380   for .clist_set:N = \l__stex_key_for_clist ,
7381   method .tl_set:N = \l__stex_key_method_tl,
7382   term .tl_set:N = \l__stex_key_term_tl,
7383   name .str_set_x:N = \l__stex_key_name_str
7384   % todo: style=inline
7385 }{id,style,title}
7386
7387 \newenvironment{spfstepenv}{
7388   \str_set_eq:NN \spfstepenvautorefname \spfstepautorefname
7389 }{}
7390
7391 \cs_new_protected:Nn \__stex_proof_step_html:nn {
7392   \stex_if_do_html:TF{
7393     \exp_args:Ne \stex_annotation:nn{
7394       shtml:spf#1={
7395         \seq_if_empty:NF \l__stex_fors_seq {
7396           \seq_use:Nn \l__stex_fors_seq ,
7397         }
7398       }
7399       \str_if_empty:NF \l__stex_key_name_str {
7400         shtml:stepname={\l__stex_key_name_str}
7401       }
7402     }{
7403       \__stex_proof_html:
7404       #2
7405     }
7406   }{ #2 }
7407 }
7408
7409 \cs_new_protected:Nn \__stex_proof_make_step_macro:Nnnnn {
7410   \NewDocumentCommand #1 {s O{} +m} {
7411     \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7412     \stex_keys_set:nn{spfsteps}{##2}
7413     \str_if_empty:NF \l__stex_key_name_str {

```

```

7414     \stex_debug:nn{Here:}{Variable-\l_stex_key_name_str}
7415     \exp_args:Nne\use:nn{\vardef}{{v\l_stex_key_name_str}[name=\l_stex_key_name_str]{\l_stex_key_name_str}}
7416 }
7417
7418 \begin{spfstepenv}
7419   \str_if_empty:NF \l_stex_key_id_str {
7420     \l__stex_proof_number_as_string:N \@currentlabel
7421     \str_set:Nx \@currentHref{spfstep.\@currentlabel}
7422     \l_stex_do_id:
7423   }
7424
7425   \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7426     \IfBooleanTF ##1 {
7427       \l__stex_proof_step_html:nn{#2}{##3}
7428     }{
7429       \l__stex_proof_step_html:nn{#2}{\l__stex_proof_start_list:n{#3} ##3 \l__stex_proof_end_#5}
7430     }
7431     \end{spfstepenv}
7432     \group_begin:\stexcommentfont
7433   }{
7434     \l__stex_proof_step_html:nn{#2}{##3}
7435   \end{spfstepenv}
7436   }
7437 }
7438
7439 \stex_deactivate_macro:Nn #1 {sproof~environments}
7440 }
7441
7442 \l__stex_proof_make_step_macro:Nnnnn \assumption {assumption} \l__stex_proof_insert_number: {}
7443 \l__stex_proof_make_step_macro:Nnnnn \conclude {conclusion} {${\Rightarrow}{}$} {} {}
7444 \l__stex_proof_make_step_macro:Nnnnn \spfstep {step} \l__stex_proof_insert_number: {} \l__stex_
7445
7446 \NewDocumentCommand \eqstep {s m} {
7447   \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7448     \group_end:
7449     \IfBooleanTF #1 {
7450       \l__stex_proof_step_html:nn{eqstep}{$= #2$}
7451     }{
7452       \l__stex_proof_step_html:nn{eqstep}{\l__stex_proof_start_list:n{$=$} $#2$ \l__stex_proof_
7453     }
7454     \group_begin:\stexcommentfont
7455   }{
7456     \l__stex_proof_step_html:nn{eqstep}{$= #2$}
7457   }
7458 }
7459 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
7460
7461 \NewDocumentCommand \yield {+m} {
7462   \stex_annotation:nn{shtml:proofterm={}}{ #1 }
7463 }
7464 \stex_deactivate_macro:Nn \yield {sproof~environments}
7465
7466 \NewDocumentEnvironment{spfblock}{}{
7467   \bool_set_false:N \l__stex_proof_in_spfblock_bool

```

```

7468 }{
7469   \aftergroup\__stex_proof_inblock_restore:
7470 }
7471 \stex_deactivate_macro:Nn \spfblock {sproof~environments}
7472 \AddToHook{env/spfblock/before} {
7473   \bool_if:NT \l__stex_proof_in_spfblock_bool \group_end:
7474 }
7475
7476
7477 \newcommand\spfjust[1]{
7478   \stex_annotate:nn{\spfjust={}}{ #1 }
7479 }
7480 \stex_deactivate_macro:Nn \spfjust {sproof~environments}

```

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

13.12 Metatheory

```

7481 <@=stex_meta>
7482 \group_begin:
7483   \cs_set:Npn \__stex_modules_persist_module: {}
7484   \cs_set:Npn \stex_check_term:n #1 {}
7485   \cs_set:Npn \__stex_sref_do_aux:n #1 { #1 }
7486   \bool_set_false:N \stex_html_do_output_bool
7487   \bool_set_false:N \c_stex_check_terms_bool
7488   \stex_uri_resolve:Nn \l_stex_current_ns_uri {http://mathhub.info/sTeX/meta}
7489   \stex_module_setup:n{Metatheory}
7490
7491 \symdef{of~type}[args=ii,invisible]{#1}
7492 \notation{of~type}[colon]{#1 \mathbin{\comp{:}}}{#2}
7493
7494 \symdef{apply}[args=ia,prec=0;\infprec x\infprec]{#1\mathopen{\comp{}}}{#2 \mathclose{\comp{}}}
7495 \notation{apply}[lambda]{#1\mathopen{\comp{}}; \argsep{#2}\mathclose{\comp{}}}
7496 \notation{apply}[infixop]{\argsep{#2}\mathbin{#1}}
7497 \notation{apply}[infixrel]{\argsep{#2}\mathrel{#1}}
7498
7499 % structures
7500 \symdef{module~type}[args=i,op=\mathtt{MOD}]{\mathopen{\comp{\mathtt{MOD}}}{#1}\mathclose{\comp{}}}
7501 \symdef{module~type~merge}[args=a,op=\oplus]{\argsep{#1}\mathbin{\oplus}}
7502 \symdef{anonymous~record}[args=a]{\mathopen{\comp{[]}}{#1}\mathclose{\comp{[]}}}
7503 \symdef{record~field}[args=2]{#1\comp{.}}{#2}
7504 \symdecl*{record~type}
7505
7506 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
7507 \notation{mathstruct}[angle,prec=nobrackets]{\mathopen{\comp{\langle}}}{#1 \mathclose{\comp{\rangle}}}
7508 \notation{mathstruct}[parens,prec=nobrackets]{\mathopen{\comp{()}}}{#1 \mathclose{\comp{()}}}
7509
7510 % sequences
7511 \symdef{ellipses}[ldots]{\ldots}

```

```

7517 \symdef{sequence~expression}[comma,args=a]{#1}
7518 \symdef{sequence~type}[args=1]{#1^{\comp\ast}}
7519 \symdef{sequence~map}[args=ia]{
7520   \comp{\mathrm{map}}{\mathopen{\comp{()#1\mathpunct{\comp{,}}}}
7521   #2\mathclose{\comp{})}}
7522 }
7523 \iffalse
7524 % binder (\forall, \Pi, \lambda etc.)
7525 \symdef{pibind}[name=dependent function type,prec=nobrackets,
7526   op=(\cdot; \to; \cdot, args=Bi,assoc=pre]
7527   {\argmap{#1}{%
7528     \mathopen{\comp{} ##1 \mathclose{\comp{}}}
7529     }{\mathbin{\comp{\to}}} \mathbin{\comp{\to}} #2}
7530 \notation{pibind}[\forall]{\comp\forall #1\mathpunct{\comp.} #2}
7531 \notation{pibind}[\Pi]{\mathop{\comp\prod}\c_math_subscript_token{#1}#2}
7532
7533 \symdef{mapbind}[name=lambda, mapsto, prec=nobrackets, op=\mapsto, args=Bi,assoc=pre]
7534   { #1 \mathrel{\comp\mapsto} #2}
7535 \notation{mapbind}[\lambda,prec=nobrackets,op=\lambda]
7536   {\comp\lambda #1 \mathpunct{\comp.} #2}
7537 \fi
7538 \symdecl{bind}[args=Bi,assoc=pre]
7539 \notation{bind}[deffun,prec=nobrackets,op=(\cdot; \to; \cdot
7540   {\mathopen{\comp{} #1 \mathclose{\comp{}}}\mathbin{\comp{\to}}} #2}
7541 \notation{bind}[\forall]{\comp\forall #1.\;#2}
7542 \notation{bind}[\Pi]{\mathop{\comp\prod}\c_math_subscript_token{#1}#2}
7543
7544 \symdef{implicit~bind}[args=Bi,assoc=pre]{\mathopen{\comp{} #1 \mathclose{\comp{\}}}\c_math_
7545
7546 \symdecl*{integer~literal}
7547 \notation{integer~literal}{\mathbb Z}
7548
7549 \symdecl*{ordinal}
7550 \notation{ordinal}{\mathit{Ord}}
7551
7552 % propositions
7553 \symdef{prop}[name=proposition]{\mathit{Prop}}
7554 \symdef{judgment~holds}[args=i,role=judgment]{\comp\vdash;#1}
7555
7556 % any object
7557 \symdef{object}{\mathit{Obj}}
7558
7559 % TODO DELETE
7560 \symdef{aseqdots}[args=a,prec=nobrackets]
7561   {#1\comp{,\ldots}}%{##1\comp,##2}
7562 \symdef{aseqfromto}[args=ai,prec=nobrackets]
7563   {#1\comp{,\ldots},#2}%{##1\comp,##2}
7564 \symdef{aseqfromtovia}[args=aai,prec=nobrackets]
7565   {#1\comp{,\ldots},#2\comp{,\ldots},#3}%{##1\comp,##2}
7566
7567
7568 \stex_close_module:
7569 \stex_uri_add_module:NNn \l_stex_metatheory_uri \l_stex_current_ns_uri {Metatheory}
7570 \global \let \l_stex_metatheory_uri \l_stex_metatheory_uri

```

```

7571   \global \let \c_stex_default_metatheory \l_stex_metatheory_uri
7572 \group_end:
```

13.13 MMT Interfaces

```

7573 <@=todo>
7574 \cs_new_protected:Npn \MSC #1 {}

\MMTinclude

7575 \stex_new_stylable_cmd:nnnn{MMTinclude}{m}{
7576   \stex_annotation_invisible:nn{shtml:import={#1}}{}}
7577 }{}
7578 \stex_deactivate_macro:Nn \MMTinclude {module~environments}
7579 \stex_every_module:n {\stex_reactivate_macro:N \MMTinclude}
```

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

```

\MMTrule

7580 \NewDocumentCommand \MMTrule {m m} {
7581   \tl_if_empty:nTF{#2}{\seq_clear:N \l_tmpa_seq}{%
7582     \seq_set_split:Nnn \l_tmpa_seq , {#2}}
7583   }
7584   \int_zero:N \l_tmpa_int
7585   \stex_annotation_invisible:n{
7586     $
7587     \stex_annotation:nn{shtml:rule={scala://#1}}{%
7588       \stex_annotation_force_break:n{%
7589         \seq_if_empty:NF \l_tmpa_seq {%
7590           \seq_map_inline:Nn \l_tmpa_seq {%
7591             \int_incr:N \l_tmpa_int
7592             \stex_annotation:nn{%
7593               shtml:argmode=i,
7594               shtml:arg={\int_use:N \l_tmpa_int}
7595             }{ ##1 }
7596           }
7597         }
7598       }
7599     }$}
7600   }
7601 }

7602 \stex_deactivate_macro:Nn \MMTrule {module~environments}
7603 \stex_every_module:n{\stex_reactivate_macro:N \MMTrule}
```

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

```

mmtinterface (env.)

7604 \NewDocumentEnvironment { mmtinterface } { O{} m m } {
7605   \stex_module_setup_top_nosig:n { #3 }
7606   \str_set_eq:NN \l__todo_mmt_module_str \l_stex_current_module_str
7607   \str_clear:N \l_stex_current_module_str
7608   \stex_keys_set:nn { smodule }{ #1 }
7609   \stex_module_setup:n{ #2 }
7610   \str_set_eq:NN \l__todo_stex_module_str \l_stex_current_module_str
7611   \stex_debug:nn{mmt}{Interface~\l__todo_stex_module_str^~Jfor~\l__todo_mmt_module_str}
7612 }
```

```

7613 \stex_if_do_html:T {
7614   \exp_args:Nne \begin{stex_annotation_env} {
7615     shtml:theory={\l_stex_current_module_str},
7616     shtml:language={ \l_stex_current_language_str},
7617     shtml:signature={}
7618     \tl_if_empty:NF \l_stex_metatheory_uri ,,
7619       shtml:metatheory={\stex_uri_use:N \l_stex_metatheory_uri}
7620   }
7621 }
7622 \stex_annotation_invisible:n{}
7623 \stex_annotation_invisible:nn
7624   {shtml:import=\l_todo_mmt_module_str} {}
7625 }
7626 \stex_module_add_code:x{
7627   \stex_activate_module:n{ \l_todo_mmt_module_str }
7628 }
7629 \stex_module_add_morphism:nonn
7630   {}{\l_todo_mmt_module_str}{import}{}
7631 \stex_reactivate_macro:N \mmtdef
7632 \stex_smsmode_do:
7633 }{
7634   \str_set_eq:NN \l_stex_current_module_str \l_todo_mmt_module_str
7635   \stex_close_module:
7636   \str_set_eq:NN \l_stex_current_module_str \l_todo_stex_module_str
7637   \stex_close_module:
7638   \stex_if_do_html:T { \end{stex_annotation_env} }
7639 }
7640 \stex_sms_allow_env:n{mmtinterface}

\mmtdef

7641 \NewDocumentCommand \mmtdef {m O{} m} {
7642   \stex_keys_set:nn{symdef}{#2}
7643   \str_set:Nx \l_stex_macroname_str { #1 }
7644   \str_if_empty:NT \l_stex_key_name_str {
7645     \str_set:Nx \l_stex_key_name_str { #1 }
7646   }
7647   \stex_symdecl_do:

7648
7649   \str_set_eq:NN \l_stex_current_module_str \l_todo_mmt_module_str
7650   \cs_set_eq:NN \l_todo_old_metagroup_cd \stex_metagroup_do_in:nn
7651   \cs_set_protected:Npn \stex_metagroup_do_in:nn ##1 ##2 {##2}
7652   \exp_args:Nnx \use:nn {\stex_module_add_symbol:nnnnnnnN} {
7653     {\l_stex_macroname_str}
7654     {\l_stex_key_name_str}
7655     {\int_use:N \l_stex_get_symbol_arity_int}
7656     {\l_stex_get_symbol_args_t1}
7657     {}
7658     {}
7659     {}
7660     \stex_invoke_symbol:
7661   }
7662   \cs_set_eq:NN \stex_metagroup_do_in:nn \l_todo_old_metagroup_cd
7663   \str_set_eq:NN \l_stex_current_module_str \l_todo_stex_module_str
7664

```

```

7665 \str_set_eq:NN \l_stex_get_symbol_mod_str \l__todo_mmt_module_str
7666 \str_set_eq:NN \l_stex_get_symbol_name_str \l_stex_key_name_str
7667 \stex_notation_parse:n{#3}
7668 \_stex_notation_check:
7669 \_stex_notation_add:
7670 \stex_if_do_html:T{
    \_stex_notation_do_html:n{\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str}
7672 }
7673 \stex_smsmode_do:
7674 }
7675 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
7676 \stex_sms_allow_escape:N \mmtdef

```

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

VoLL-KI Annotations

```

7677 \newcommand\precondition[2]{
7678     \str_clear:N \l_stex_get_symbol_name_str
7679     \stex_get_symbol:n{#2}
7680     \str_if_empty:NTF \l_stex_get_symbol_name_str{
        \errmessage{Unknown~symbol~#2}
    }{
7683 \str_case:nnTF {#1}{
        {remember}{}
        {understand}{}
        {analyze}{}
        {evaluate}{}
        {apply}{}
        {create}{}
    }{
7691 \stex_annotation_invisible:nn{
        \shtml:preconditionsymbol={\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str},
        \shtml:preconditiondimension={#1}
    }{}
7695 }{\errmessage{Unknown~cognitive~dimension~#1}}
    }
7696 }
7697 }
7698 \newcommand\objective[2]{
7699     \str_clear:N \l_stex_get_symbol_name_str
7700     \stex_get_symbol:n{#2}
7701     \str_if_empty:NTF \l_stex_get_symbol_name_str{
        \errmessage{Unknown~symbol~#2}
    }{
7704 \str_case:nnTF {#1}{
        {remember}{}
        {understand}{}
        {analyze}{}
        {evaluate}{}
        {apply}{}
        {create}{}
    }{
7711 \stex_annotation_invisible:nn{
        \shtml:objectivesymbol={\l_stex_get_symbol_mod_str ? \l_stex_get_symbol_name_str},
        \shtml:objectivedimension={#1}
    }{}
7715 }

```

```

7716     }{\errmessage{Unknown~cognitive~dimension~#1}}
7717     }
7718 }
7719 \seq_if_empty:NT \g_stex_current_file {
7720   \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file
7721 }
7722 \_stex_persist_read_now:
7723 \_stex_every_file:
7724 \cs_new_protected:Nn \__todo_newlabel:n {
7725   \exp_args:Ne \__todo_old_newlabel:{\tl_to_str:n{#1}}
7726 }
7727 \AtBeginDocument{
7728   \iow_now:Nn \@auxout {
7729     \ExplSyntaxOn
7730     \let\__todo_old_newlabel:\newlabel
7731     \let\newlabel\__todo_newlabel:n
7732     \ExplSyntaxOff
7733   }
7734 }
7735 
```

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.

```
7736 <*cls>
7737 <@=notesslides>
7738 \ProvidesExplClass{notesslides}{2023/03/19}{3.3.0}{notesslides Class}
7739 \RequirePackage{13keys2e}
7740
7741 \str_const:Nn \c__notesslides_class_str {article}
7742
7743 \keys_define:nn{notesslides / cls}{
7744     class .str_set_x:N = \c__notesslides_class_str,
7745     notes .bool_set:N = \c__notesslides_notes_bool ,
7746     slides .code:n    = { \bool_set_false:N \c__notesslides_notes_bool },
7747     %docopt .str_set_x:N = \c__notesslides_docopt_str,
7748     unknown .code:n   = {
7749         \PassOptionsToClass{\CurrentOption}{beamer}
7750         \PassOptionsToClass{\CurrentOption}{\c__notesslides_class_str}
7751         \PassOptionsToPackage{\CurrentOption}{notesslides}
7752         \PassOptionsToPackage{\CurrentOption}{stex}
7753     }
7754 }
7755 \ProcessKeysOptions{ notesslides / cls }
7756
7757 \RequirePackage{stex}
7758 \stex_if_html_backend:T {
7759     \bool_set_true:N \c__notesslides_notes_bool
7760 }
7761
7762 \bool_if:NTF \c__notesslides_notes_bool {
7763     \PassOptionsToPackage{notes=true}{notesslides}
7764     \message{notesslides.cls:-Formatting-document-in-notes-mode}
```

```

7765 }{
7766   \PassOptionsToPackage{notes=false}{notesslides}
7767   \message{notesslides.cls:~Formatting~document~in~slides~mode}
7768 }
7769
7770 \bool_if:NTF \c_notesslides_notes_bool {
7771   \LoadClass{\c__notesslides_class_str}
7772 }{
7773   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7774   \%newcounter{Item}
7775   \%newcounter{paragraph}
7776   \%newcounter{ subparagraph}
7777   \%newcounter{Hfootnote}
7778 }
7779 \RequirePackage{notesslides}
7800 
```

now we do the same for the `notesslides` package.

```

7781 <*package>
7782 \ProvidesExplPackage{notesslides}{2023/03/19}{3.3.0}{notesslides Package}
7783 \RequirePackage{l3keys2e}
7784
7785 \keys_define:nn{notesslides / pkg}{
7786   notes          .bool_set:N  = \c_notesslides_notes_bool ,
7787   slides         .code:n    = { \bool_set_false:N \c_notesslides_notes_bool },
7788   sectocframes  .bool_set:N  = \c_notesslides_sectocframes_bool ,
7789   topsect        .str_set_x:N = \c_notesslides_topsect_str,
7790   unknown        .code:n    = {
7791     \PassOptionsToPackage{\CurrentOption}{stex}
7792     \PassOptionsToPackage{\CurrentOption}{tikzinput}
7793   }
7794 }
7795 \ProcessKeysOptions{ notesslides / pkg }
7796
7797 \RequirePackage{stex}
7798 \stex_if_html_backend:T {
7799   \bool_set_true:N\c_notesslides_notes_bool
7800 }
7801
7802 \cs_set:Npn \sectiontitleemph #1 {
7803   \textbf{\Large #1}
7804 }
7805
7806 \newif\ifnotes
7807 \bool_if:NTF \c_notesslides_notes_bool {
7808   \notestrue
7809   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7810   \RequirePackage[noamsthm,hyperref]{beamerarticle}
7811   \RequirePackage{mdframed}
7812   \str_if_empty:NTF \c_notesslides_topsect_str{
7813     \%setsectionlevel{section}
7814   }{
7815     \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7816   }

```

```

7817 }{
7818   \notesfalse
7819
7820   \cs_new_protected:Nn \__notesslides_do_sectocframes: {
7821     \cs_set_protected:Nn \__notesslides_do_label:n {
7822       \str_case:nnF{##1} {
7823         {part} {
7824           \tl_set:Nx\l__notesslides_num{\the part}
7825           \tl_set:cx{@ @ label}{%
7826             \cs_if_exist:NTF\parttitlename{\exp_not:N\parttitlename}{\exp_not:N\partname}{-}}
7827         }
7828         {chapter} {
7829           \tl_set:Nx\l__notesslides_num{\the chapter}
7830           \tl_set:cx{@ @ label}{%
7831             \cs_if_exist:NTF\chapertitlename{\exp_not:N\chapertitlename}{\exp_not:N\chaptername}{-}}
7832         }
7833         {section} {
7834           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesection.}
7835           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7836         }
7837         {subsection} {
7838           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsection.}
7839           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7840         }
7841         {subsubsection} {
7842           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsubsection.}
7843           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7844         }
7845         {paragraph} {
7846           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubparagraph.}
7847           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7848         }
7849       }{
7850         \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\the chapter.}\thesubsubsubsection.}
7851         \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7852       }
7853     }
7854     \cs_set_protected:Nn \_sfragment_do_level:nn {
7855       \tl_if_exist:cT{c@##1}{\stepcounter{##1}}
7856       \addcontentsline{toc}{##1}{\protect\numberline{\use:c{the##1}}##2}
7857       \__notesslides_do_label:n{##1}
7858       \pdfbookmark[\int_use:N \l_stex_docheader_sect]{\l__notesslides_num##2}{##1.\l__note}
7859       \begin{frame}[noframenumbering]
7860         \vfill\centering
7861         \sectiontitleemph{%
7862           \use:c{@ @ label} ##2
7863         }
7864         \end{frame}
7865         \int_incr:N \l_stex_docheader_sect
7866         \tl_set:Nn \stex_current_section_level{##1}
7867     }
7868   }
7869
7870 \AtBeginDocument{

```

```

7871 \str_if_empty:NTF \c_notesslides_topsect_str {
7872   \setsectionlevel{section}
7873 } {
7874   \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7875   \exp_args:No \str_if_eq:nnTF \c_notesslides_topsect_str {chapter} {
7876     \__notesslides_define_chapter:
7877   }{
7878     \exp_args:No \str_if_eq:nnT \c_notesslides_topsect_str {part} {
7879       \__notesslides_define_chapter:
7880       \__notesslides_define_part:
7881     }
7882   }
7883 }
7884 }
7885
7886 \bool_if:NT \c_notesslides_sectocframes_bool {
7887   \__notesslides_do_sectocframes:
7888 }
7889 }
7890
7891 \cs_new_protected:Nn \__notesslides_define_chapter: {
7892   \cs_if_exist:NF \chaptername {
7893     \cs_set_protected:Npn \chaptername {Chapter}
7894   }
7895   \cs_if_exist:NF \chapter {
7896     \cs_set_protected:Npn \chapter {INVALID}
7897   }
7898   \cs_if_exist:NF \c@chapter {
7899     \newcounter{chapter}\counterwithin*{section}{chapter}
7900   }
7901 }
7902
7903 \cs_new_protected:Nn \__notesslides_define_part: {
7904   \cs_if_exist:NF \partname {
7905     \cs_set_protected:Npn \partname {Part}
7906   }
7907   \cs_if_exist:NF \part {
7908     \cs_set_protected:Npn \part {INVALID}
7909   }
7910   \cs_if_exist:NF \c@part {
7911     \newcounter{part}\counterwithin*{chapter}{part}
7912   }
7913 }

```

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

```

7914 \def \c_notesslides_document_str{document}
7915 \newcommand\afterprematurestop{}
7916 \def\prematurestop@endsfragment{
7917   \unless\ifx\@currenvir\c_notesslides_document_str
7918     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7919     \expandafter\prematurestop@endsfragment
7920   \fi
7921 }

```

```

7922 \providecommand\prematurestop{
7923   \stex_if_html_backend:F{
7924     \message{Stopping-sTeX-processing-prematurely}
7925     \prematurestop@endsfragment
7926   \afterprematurestop
7927   \end{document}
7928 }
7929 }
```

(End of definition for `\prematurestop`. This function is documented on page 105.)

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.

```

7930 \bool_if:NT \c_notesslides_notes_bool {
7931   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamertheme#2}}
7932 }
7933 \NewDocumentCommand \libusetheme {O{} m} {
7934   \libusepackage[#1]{beamertheme#2}
7935 }
```

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

```

7936 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7937 \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.

```

7938 \ifnotes
7939
7940 \newlength{\slideframewidth}
7941 \setlength{\slideframewidth}{1.5pt}
```

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

```

7942 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7943   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7944     \bool_set_true:N #1
7945   }{
7946     \bool_set_false:N #1
7947   }
7948 }
7949
7950 \stex_keys_define:nnnn{notesslides / frame}{
7951   \str_clear:N \l__notesslides_frame_label_str
7952   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7953   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7954   \bool_set_true:N \l__notesslides_frame_fragile_bool
7955   \bool_set_true:N \l__notesslides_frame_shrink_bool
7956   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7957   \bool_set_true:N \l__notesslides_frame_t_bool
7958 }
7959   label .str_set_x:N = \l__notesslides_frame_label_str,
7960   allowframebreaks .code:n = {
```

```

7961     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7962 },
7963 allowdisplaybreaks .code:n      = {
7964     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7965 },
7966 fragile          .code:n      = {
7967     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7968 },
7969 shrink           .code:n      = {
7970     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7971 },
7972 squeeze          .code:n      = {
7973     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7974 },
7975 t                .code:n      = {
7976     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7977 },
7978 unknown         .code:n      = {}
7979 }{}
```

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

```

7980 \cs_new_protected:Nn \__notesslides_setup_itemize: {
7981     \def\itemize@level{outer}
7982     \def\itemize@outer{outer}
7983     \def\itemize@inner{inner}
7984     \%newcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
7985     \renewenvironment{itemize} {
7986         \ifx\itemize@level\itemize@outer
7987             \def\itemize@label{$\rhd$}
7988         \fi
7989         \ifx\itemize@level\itemize@inner
7990             \def\itemize@label{$\scriptstyle\rhd$}
7991         \fi
7992         \begin{list}
7993             {\itemize@label}
7994             {\setlength{\labelsep}{.3em}
7995              \setlength{\labelwidth}{.5em}
7996              \setlength{\leftmargin}{1.5em}
7997            }
7998             \edef\itemize@level{\itemize@inner}
7999         }{
8000             \end{list}
8001         }
8002     }
```

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

```

8003 \stex_if_html_backend:TF {
8004     \cs_new_protected:Nn \__notesslides_frame_box_begin: {
8005         \vbox\bgroup
8006         \begin{stex_annotation_env}[shtml:frame={}]
8007             \mdf@patchamsthm\notesslidesfont
8008         }
8009     \cs_new_protected:Nn \__notesslides_frame_box_end: {
8010         %^A \notesslides@slide_label
```

```

8011     \medskip\par\noindent\tiny\notesslidesfooter
8012     \end{stex_annotation_env}\egroup
8013   }
8014 }{
8015   \cs_new_protected:Nn \__notesslides_frame_box_begin: {
8016     \begin{mdframed}[
8017       linewidth=\slideframewidth,
8018       skipabove=1ex,
8019       skipbelow=1ex,
8020       userdefinedwidth=\slidewidth,
8021       align=center
8022     ]\notesslidesfont
8023   }
8024   \cs_new_protected:Nn \__notesslides_frame_box_end: {
8025     \medskip\par\noindent\tiny\notesslidesfooter%^^A\notesslides@slidelabel
8026     \end{mdframed}
8027   }
8028 }

```

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

```

8029 \renewenvironment{frame}[1][]{
8030   \stex_keys_set:nn{notesslides / frame}{#1}
8031   \stepcounter{framenumber}
8032   \renewcommand{\newpage}{\addtocounter{framenumber}{1}}
8033   \def\@currentlabel{\theframenumber}
8034   \str_if_empty:NF \l__notesslides_frame_label_str {
8035     \label{\l__notesslides_frame_label_str}
8036   }
8037   \__notesslides_setup_itemize:
8038   \__notesslides_frame_box_begin:
8039 }{
8040   \__notesslides_frame_box_end:
8041 }

```

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

```

\frametitle
8042 \renewcommand{\frametitle}[1]{
8043   \stexdoctitle { #1 }
8044   \notesslidestitlenameph{#1}\medskip
8045 }

```

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

`\pause`

```
8046 \newcommand{\pause}{}
```

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

We redefine the `columns` and `column` environments:

```

8047 \renewenvironment{columns}[1][]{
8048   \par\noindent
8049   \begin{minipage}
8050     \slidewidth\centering\leavevmode
8051   % \stex_if_html_backend:T{

```

```

8052 %     \cs_if_exist:NT \rustex_if:T {
8053 %         \rustex_if:T {\par
8054 %             \rustex_direct_HTML:n{<table><tr><td>}
8055 %         }
8056 %     }
8057 % }
8058 }{
8059 % \stex_if_html_backend:T{
8060 %     \cs_if_exist:NT \rustex_if:T {
8061 %         \rustex_if:T {\par
8062 %             \rustex_direct_HTML:n{</td></tr></table>}
8063 %         }
8064 %     }
8065 % }
8066 \end{minipage}\par\noindent
8067 }
8068 \newsavebox\columnbox
8069 \renewenvironment<>{column}[2] []{
8070     \begin{lrbox}{\columnbox}
8071 % \stex_if_html_backend:T{
8072 %     \cs_if_exist:NT \rustex_if:T {
8073 %         \rustex_if:T {\par
8074 %             \rustex_direct_HTML:n{</td><td>}
8075 %         }
8076 %     }
8077 % }
8078     \begin{minipage}{#2}
8079 }{
8080     \end{minipage}
8081 % \stex_if_html_backend:T{
8082 %     \cs_if_exist:NT \rustex_if:T {
8083 %         \rustex_if:T {\par
8084 %             \rustex_direct_HTML:n{</td><td>}
8085 %         }
8086 %     }
8087 % }
8088     \end{lrbox}\usebox\columnbox
8089 }
8090 \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.

```

8091 \bool_if:NTF \c_notesslides_notes_bool {
8092     \renewenvironment{note}{\ignorespaces}{}
8093 }{
8094     \renewenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup\egroup}
8095 }

```

For other environments we introduce variants prefixed with `n`, which are excluded in `slides` mode.

```

8096 \cs_new_protected:Nn \__notesslides_notes_env:n {
8097   \bool_if:NTF \c_notesslides_notes_bool {
8098     \newenvironment{#1}#2{#3}{#4}
8099   }{
8100     \newenvironment{#1}#2{
8101       \cs_set:Npn \__notesslides_eat: #####1 \end #####2 {
8102         \str_if_eq:nnTF{#1}{#####2} {
8103           \end{#1}
8104         }{
8105           \__notesslides_eat:
8106         }
8107       }
8108       \__notesslides_eat:
8109       \%setbox\l_tmpa_box\vbox\bgroup#3
8110     }{
8111       %#4\egroup
8112     }
8113   }
8114 }

8115 \__notesslides_notes_env:nnnn{nparagraph}{[1]}{\begin{sparagraph}{#1}}{\end{sparagraph}}
8116 \__notesslides_notes_env:nnnn{nfragment}{[2]}{\begin{sfragment}{#1}{#2}}{\end{sfragment}}
8117 \__notesslides_notes_env:nnnn{ndefinition}{[1]}{\begin{sdefinition}{#1}}{\end{sdefinition}}
8118 \__notesslides_notes_env:nnnn{nassertion}{[1]}{\begin{sassertion}{#1}}{\end{sassertion}}
8119 \__notesslides_notes_env:nnnn{nproof}{[2]}{\begin{sproof}{#1}{#2}}{\end{sproof}}
8120 \__notesslides_notes_env:nnnn{nexample}{[1]}{\begin{sexample}{#1}}{\end{sexample}}
8121

8122 \RequirePackage{graphicx}

8123 \NewDocumentCommand\frameimage{s O{} m}{

8124   \IfBooleanTF #1 {
8125     \begin{frame}[plain]
8126   }{
8127     \begin{frame}
8128   }
8129   \bool_if:NTF \c_notesslides_notes_bool {
8130     \slidewidth=\dimexpr\slidewidth-(2\slideframewidth)\relax
8131   }{
8132     \slidewidth=\textwidth\relax
8133   }
8134   \def\Gin@ewidth{}\setkeys{Gin}{#2}
8135   \tl_if_empty:NTF \Gin@ewidth {
8136     \mhgraphics[width=\slidewidth,#2]{#3}
8137   }{
8138     \mhgraphics[#2]{#3}
8139   }
8140   \end{frame}
8141 }

8142 }

8143 }

8144 hacking inputref:

8145 \inputref*

```

```

8144 \cs_set_eq:NN \__notesslides_inputref:\inputref
8145 \cs_set_protected:Npn \inputref{@ifstar\ninputref\__notesslides_inputref:}

```

```

8146 \bool_if:NTF \c_notesslides_notes_bool {
8147   \newcommand\nininputref[2][]{%
8148     \__notesslides_inputref:[#1]{#2}%
8149   }%
8150 }{%
8151   \newcommand\nininputref[2]{}%
8152 }

```

(End of definition for `\inputref*`. This function is documented on page 104.)

14.1.4 Styling Across Notes/Slides

```

8153 \def\notesslidestitleemph#1{%
8154   {\Large\bf\sf#1}%
8155   \vskip0.1\baselineskip
8156   \leaders\vrule width \textwidth
8157   \vskip0.4pt%
8158   \nointerlineskip
8159 }%
8160
8161 \def\notesslidesfooter{}%
8162
8163 \let\notesslidesfont\sffamily

```

14.1.5 Beamer Compatibility

All of this should be removed and made part of a template

```

8164
8165 \bool_if:NT \c_notesslides_notes_bool {%
8166   \def\author{\@dblarg\ns@author}%
8167   \long\def\ns@author[#1]#2{%
8168     \tl_if_empty:nTF{#1}{%
8169       \def\beamer@shortauthor{#2}%
8170     }{%
8171       \def\beamer@shortauthor{#1}%
8172     }%
8173   \def\@author{#2}%
8174 }%
8175 \def\title{\@dblarg\ns@title}%
8176 \long\def\ns@title[#1]#2{%
8177   \tl_if_empty:nTF{#1}{%
8178     \def\beamer@shorttitle{#2}%
8179   }{%
8180     \def\beamer@shorttitle{#1}%
8181   }%
8182   \def\@title{#2}%
8183   \stexdoctitle{#2}%
8184 }%
8185 \def\insertshortauthor{%
8186   \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shortauthor\beamer@shortauthor\egroup
8187 }%
8188 \def\insertshorttitle{%
8189   \hbox\bgroup\def\\{}\cs_if_exist:NT\beamer@shorttitle\beamer@shorttitle\egroup
8190 }%
8191 \stex_if_html_backend:TF{%

```

```

8192     \def\insertframenumber{\stex_annotate:nn{shml:framenumber={}}{}}
8193   }{
8194     \def\insertframenumber{\@arabic\c@framenumber}
8195   }
8196   \def\insertshortdate{\today}
8197 }

```

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.

```

8198 \gdef\printexcursions{}
8199 \newcommand\excursionref[2]{% label, text
8200   \bool_if:NT \c_notesslides_notes_bool {
8201     \begin{sparagraph}[title=Excursion]
8202       #2 \sref[fallback=the appendix]{#1}.
8203     \end{sparagraph}
8204   }
8205 }
8206 \newcommand\activate@excursion[2][]{%
8207   \tl_gput_right:Nn\printexcursions{\inputref[#1]{#2}}
8208 }
8209 \newcommand\excursion[4][]{% repos, label, path, text
8210   \bool_if:NT \c_notesslides_notes_bool {
8211     \activate@excursion[#1]{#3}
8212     \excursionref{#2}{#4}
8213   }
8214 }

```

(End of definition for `\excursion`. This function is documented on page 106.)

\excursiongroup

```

8215 \keys_define:nn{notesslides / excursiongroup }{
8216   id      .str_set_x:N = \l__notesslides_excursion_id_str,
8217   intro    .tl_set:N    = \l__notesslides_excursion_intro_tl,
8218   archive  .str_set_x:N = \l__notesslides_excursion_mrepos_str
8219 }
8220 \cs_new_protected:Nn \__notesslides_excursion_args:n {
8221   \tl_clear:N \l__notesslides_excursion_intro_tl
8222   \str_clear:N \l__notesslides_excursion_id_str
8223   \str_clear:N \l__notesslides_excursion_mrepos_str
8224   \keys_set:nn {notesslides / excursiongroup }{ #1 }
8225 }
8226 \newcommand\excursiongroup[1][]{%
8227   \__notesslides_excursion_args:n{ #1 }
8228   \tl_if_empty:NF\printexcursions
8229   {\IfInputref{}{\begin{note}
8230     \begin{sfragment}{Excursions}% TODO pass on id
8231     \ifdefempty\l__notesslides_excursion_intro_tl{}{
8232       \exp_args:NNe \use:nn \inputref{[\l__notesslides_excursion_mrepos_str]}{
8233         \l__notesslides_excursion_intro_tl
8234       }
8235     }
8236   }

```

```

8236     \printexcursions%
8237     \end{sfragment}
8238     \end{note}}}
8239 }
8240 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
(End of definition for \excursiongroup. This function is documented on page 106.)
8241 \prop_new:N \g__notesslides_variables_prop
8242 \cs_set_protected:Npn \setSGvar #1 #2 {
8243     \prop_gput:Nnn \g__notesslides_variables_prop {#1}{#2}
8244 }
8245 \cs_set_protected:Npn \useSGvar #1 {
8246     \prop_item:Nn \g__notesslides_variables_prop {#1}
8247 }
8248 \cs_set_protected:Npn \ifSGvar #1 #2 #3 {
8249     \prop_get:NnNF \g__notesslides_variables_prop {#1} \l__notesslides_tmp {
8250         \PackageError{document-structure}
8251         {The sTeX Global variable #1 is undefined}
8252         {set it with \protect\setSGvar}\TODO better error
8253     }
8254     \tl_if_eq:NnT \l__notesslides_tmp {#2}{ #3 }
8255 }
8256
8257
8258 </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.

```

8259 <*package>
8260 <@@=problems>
8261 \ProvidesExplPackage{problem}{2023/03/19}{3.3.0}{Semantic Markup for Problems}
8262 \RequirePackage{l3keys2e}
8263
8264 \keys_define:nn { problem / pkg }{
8265     notes      .default:n   = { true },
8266     notes      .bool_set:N = \c__problems_notes_bool,
8267     gnotes     .default:n   = { true },
8268     gnotes     .bool_set:N = \c__problems_gnotes_bool,
8269     hints      .default:n   = { true },
8270     hints      .bool_set:N = \c__problems_hints_bool,
8271     solutions   .default:n   = { true },
8272     solutions   .bool_set:N = \c__problems_solutions_bool,
8273     pts        .default:n   = { true },
8274     pts        .bool_set:N = \c__problems_pts_bool,
8275     min        .default:n   = { true },
8276     min        .bool_set:N = \c__problems_min_bool,
8277     %boxed     .default:n   = { true },
8278     %boxed     .bool_set:N = \c__problems_boxed_bool,

```

```

8279 test .default:n = { true },
8280 test .bool_set:N = \c__problems_test_bool,
8281 unknown .code:n = {
8282   \PassOptionsToPackage{\CurrentOption}{stex}
8283 }
8284 }
8285 \newif\ifsolutions
8286
8287 \ProcessKeysOptions{ problem / pkg }
8288 \bool_if:NTF \c__problems_solutions_bool {
8289   \solutionstrue
8290 }{
8291   \solutionsfalse
8292 }
8293 \newif\ifintest
8294 \bool_if:NTF \c__problems_test_bool {
8295   \intesttrue
8296 }{
8297   \intestfalse
8298 }
8299
8300 \RequirePackage{stex}

```

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

```

8301 \AddToHook{begindocument}{
8302   \ExplSyntaxOn\makeatletter
8303   \input{problem-english.ldf}
8304   \ltx@ifpackageloaded{babel}{
8305     \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
8306       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8307         \input{problem-ngerman.ldf}
8308       }
8309       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8310         \input{problem-finnish.ldf}
8311       }
8312       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8313         \input{problem-french.ldf}
8314       }
8315       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8316         \input{problem-russian.ldf}
8317       }
8318   }{}}
8319   \makeatother\ExplSyntaxOff
8320 }

```

(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.

```
8321 \bool_new:N \l_stex_key_autogradable_bool
```

```

8322 \stex_keys_define:nnnn{ problem }{
8323   \tl_set:Nn \l_stex_key_pts_tl 0
8324   \tl_set:Nn \l_stex_key_min_tl 0
8325   \str_clear:N \l_stex_key_name_str
8326   \str_clear:N \l_stex_key_mhrepos_str
8327   \bool_set_false:N \l_stex_key_autogradable_bool
8328 }{
8329   pts .tl_set:N = \l_stex_key_pts_tl,
8330   min .tl_set:N = \l_stex_key_min_tl,
8331   name .str_set:N = \l_stex_key_name_str,
8332   autogradable .bool_set:N = \l_stex_key_autogradable_bool,
8333   archive .code:n = {},
8334   %archive .str_set:N = \l_stex_key_mhrepos_str,
8335   creators .code:n = {}
8336   %imports .tl_set:N = \l__problems_prob_imports_tl,
8337   %refnum .int_set:N = \l__problems_prob_refnum_int,
8338 }{id,title,style,uses}

```

Then we set up a counter for problems.

```
\numberproblemsin
8339 \newcounter{sproblem}[section]
8340 \newcommand\numberproblemsin[1]{
8341   \@addtoreset{sproblem}{#1}
8342   \def\thesproblem{\arabic{#1}.\arabic{sproblem}}
8343 }
8344 \numberproblemsin{section}
8345 \%def\theplainsproblem{\arabic{sproblem}}
8346 \%def\thesproblem{\thesection.\theplainsproblem}
```

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

```
sproblem (env.)
8347 \cs_new:Nn \__problems_activate_macros: {
8348   \stex_reactivate_macro:N \solution
8349   \stex_reactivate_macro:N \mcb
8350   \stex_reactivate_macro:N \scb
8351   \stex_reactivate_macro:N \fillinsol
8352   \stex_reactivate_macro:N \hint
8353   \stex_reactivate_macro:N \exnote
8354   \stex_reactivate_macro:N \gnote
8355 }
8356
8357 \newcounter{pts}
8358 \newcounter{min}
8359 \bool_new:N \l__problems_in_problem_bool
8360 \bool_new:N \l__problems_has_pts_bool
8361 \bool_new:N \l__problems_has_min_bool
8362 \bool_set_false:N \l__problems_in_problem_bool
8363 \stex_new_stylable_env:nnnnnn {problem} {0{}>{
8364   \bool_if:NT \l__problems_in_problem_bool {
8365     \msg_error:nn{stex}{error/nestedproblem}
8366   }
8367   \cs_if_exist:NTF \l_problem_inputproblem_keys_tl {
8368     \tl_put_left:Nn \l_problem_inputproblem_keys_tl {#1,}

```

```

8369   \exp_args:Nno \stex_keys_set:nn{problem}{
8370     \l_problem_inputproblem_keys_tl
8371   }
8372 }{
8373   \stex_keys_set:nn{problem}{#1}
8374 }
8375 \refstepcounter{sproblem}
8376
8377 \stex_if_do_html:T {
8378   \str_if_empty:NT \l_stex_key_name_str {
8379     \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8380     \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8381   }
8382   \exp_args:Nne \begin{stex_annotation_env} {
8383     shtml:problem={\l_stex_key_name_str},
8384     shtml:language={ \l_stex_current_language_str},
8385     shtml:autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
8386   }
8387   \stex_annotation_invisible:n{}
8388   \tl_if_empty:NF \l_stex_key_title_tl {
8389     \exp_args:No \stexdoctitle \l_stex_key_title_tl
8390   }
8391 }
8392
8393 \tl_set_eq:NN \thistitle \l_stex_key_title_tl
8394
8395 \bool_set_true:N \l__problems_in_problem_bool
8396 \tl_set_eq:NN \l__problems_pts_tl \l_stex_key_pts_tl
8397 \tl_set_eq:NN \l__problems_min_tl \l_stex_key_min_tl
8398 \tl_if_eq:NnTF \l__problems_pts_tl {0}
8399   {\bool_set_false:N \l__problems_has_pts_bool}
8400   {\bool_set_true:N \l__problems_has_pts_bool}
8401 \tl_if_eq:NnTF \l__problems_min_tl {0}
8402   {\bool_set_false:N \l__problems_has_min_bool}
8403   {\bool_set_true:N \l__problems_has_min_bool}
8404 \int_gzero:N \g__problems_subproblem_int
8405
8406 \stex_style_apply:
8407 \stex_do_id:
8408 \__problems_activate_macros:
8409 }{
8410   \addtocounter{pts}{\l__problems_pts_t1}
8411   \addtocounter{min}{\l__problems_min_t1}
8412   \__problems_record_problem:
8413   \stex_style_apply:
8414   \stex_if_do_html:T{ \end{stex_annotation_env} }
8415 }{
8416   \par\noindent\problemheader
8417   \bool_if:NT \c__problems_pts_bool {
8418     \tl_if_eq:NnF \l__problems_pts_t1 {0} {
8419       \marginpar{\l__problems_pts_t1{}-\problem@kw@pts\smallskip}
8420     }
8421   }
8422   \bool_if:NT \c__problems_min_bool {

```

```

8423   \tl_if_eq:NnF \l__problems_min_tl {0} {
8424     \marginpar{\l__problems_min_tl{}-\problem@kw@minutes\smallskip}
8425   }
8426 }
8427 \par
8428 \stex_ignore_spaces_and_pars:
8429 }{
8430   \par\bigskip
8431 % \bool_if:NT \c__problems_test_bool \pagebreak
8432 }{s}
8433
8434 \tl_set:Nn \problemheader {
8435   \textbf{\sproblemautorefname{}\thesproblem}
8436   \tl_if_empty:NF \thistitle {
8437     {-}(\thistitle)
8438   }
8439 }
8440 }
8441
8442 \cs_new_protected:Nn \__problems_record_problem: {
8443   \exp_args:NNe \iow_now:Nn \auxout {
8444     \problem@restore {\thesproblem}{\l__problems_pts_tl}{\l__problems_min_tl}
8445   }
8446 }
8447
8448 \cs_new_protected:Npn \problem@restore #1 #2 #3 {}

subproblem (env.)
8449 \int_new:N \g__problems_subproblem_int
8450
8451 \stex_new_stylable_env:nnnnnnn {subproblem} {0}{){
8452   \stex_keys_set:nn{problem}{#1}
8453   \bool_if:NF \l__problems_in_problem_bool{
8454     \ifstexhtml\else
8455       \par{\bfseries WARNING-subproblem-to-be-used-in-some-problem}\par
8456     \fi
8457     \__problems_activate_macros:
8458     \bool_set_true:N \l__problems_in_problem_bool
8459     \tl_set:Nn \l__problems_pts_tl{0}
8460     \tl_set:Nn \l__problems_min_tl{0}
8461   }
8462   \str_if_empty:NT \l_stex_key_name_str {
8463     \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8464     \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8465   }
8466   \stex_if_do_html:T {
8467     \str_if_empty:NT \l_stex_key_name_str {
8468       \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8469       \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8470     }
8471     \exp_args:Nne \begin{stex_annotation_env} {
8472       shtml:subproblem={\l_stex_key_name_str},
8473       shtml:language={ \l_stex_current_language_str},
8474       shtml:autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}

```

```

8475      }
8476      \stex_annotation_invisible:n{ }
8477      \tl_if_empty:NF \l_stex_key_title_tl {
8478        \exp_args:No \stexdoctitle \l_stex_key_title_tl
8479      }
8480    }
8481    \int_gincr:N \g__problems_subproblem_int
8482    \stex_if_smsmode:F \stex_style_apply:
8483    \bool_if:NF \l__problems_has_pts_bool {
8484      \tl_gset:Nx \l__problems_pts_tl {\int_eval:n {\l__problems_pts_tl + \l_stex_key_pts_tl}}
8485    }
8486    \bool_if:NF \l__problems_has_min_bool {
8487      \tl_gset:Nx \l__problems_min_tl {\int_eval:n {\l__problems_min_tl + \l_stex_key_min_tl}}
8488    }
8489  }{
8490    \stex_if_smsmode:F \stex_style_apply:
8491    \stex_if_do_html:T{ \end{stex_annotation_env} }
8492  }{
8493    \begin{list}{}{
8494      \setlength\topsep{0pt}
8495      \setlength\parsep{0pt}
8496      \setlength\rightmargin{0pt}
8497    }\item[\int_use:N \g__problems_subproblem_int .]
8498  }{
8499    \bool_if:NT \c__problems_pts_bool {
8500      \bool_if:NF \l__problems_has_pts_bool {
8501        \marginpar{\l_stex_key_pts_tl{}~\problem@kw@pts\smallskip}
8502      }
8503    }
8504    \bool_if:NT \c__problems_min_bool {
8505      \bool_if:NF \l__problems_has_min_bool{
8506        \marginpar{\l_stex_key_min_tl{}~\problem@kw@minutes\smallskip}
8507      }
8508    }
8509    \end{list}
8510  }{}}

\includeproblem
8511 \stex_keys_define:nnnn{ includeproblem }{
8512   \str_clear:N \l_stex_key_mhrepos_str
8513 }{
8514   archive .str_set:N     = \l_stex_key_mhrepos_str,
8515   unknown .code:n = {}
8516 }{}

8517 \NewDocumentCommand\includeproblem{o{} m}{
8518   \group_begin:
8519   \tl_set:Nn \l_problem_inputproblem_keys_tl {#1}
8520   \stex_keys_set:nn{includeproblem}{#1}
8521   \exp_args:Nno \use:nn{\inputref[]}\l_stex_key_mhrepos_str]{#2}
8522   \group_end:
8523 }

8524 }
8525

```

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

```

solution (env.)
8526 \int_new:N \g_problem_id_counter
8527 \dim_new:N \l_stex_key_testspace_dim
8528 \stex_keys_define:nmmn{ solution }{
8529   \str_clear:N \l_stex_key_answerclass_str
8530   \dim_zero:N \l_stex_key_testspace_dim
8531 }{
8532   testspace .dim_set:N = \l_stex_key_testspace_dim,
8533   answerclass .str_set:N = \l_stex_key_answerclass_str
8534 }{id,title,style}
8535
8536 \cs_new_protected:Nn \__problems_solution_start:n {
8537   \stex_keys_set:nn{ solution }{#1}
8538   \str_if_empty:NT \l_stex_key_id_str {
8539     \int_gincr:N \g_problem_id_counter
8540     \str_set:Nx \l_stex_key_id_str {
8541       SOLUTION_\int_use:N \g_problem_id_counter
8542     }
8543   }
8544   \stex_if_do_html:T{
8545     \begin{stex_annotation_env}[
8546       shtml:solution=\l_stex_key_id_str,
8547       shtml:answerclass={\l_stex_key_answerclass_str}
8548     ]
8549   }
8550   \stex_style_apply:
8551 }
8552
8553 \stex_new_stylable_env:nnnnnn { solution }{ 0{} }{
8554   \stex_if_do_html:TF{
8555     \__problems_solution_start:n{#1}
8556   }{
8557     \ifsolutions
8558       \__problems_solution_start:n{#1}
8559     \else
8560       \stex_keys_set:nn{ solution }{#1}
8561       \testspace{\l_stex_key_testspace_dim}
8562       \setbox\l_tmpa_box\vbox\bgroup
8563     \fi
8564   }
8565 }{
8566   \stex_if_do_html:TF{
8567     \stex_style_apply:
8568     \end{stex_annotation_env}
8569   }{
8570     \ifsolutions
8571       \stex_style_apply:
8572       \stex_if_do_html:T{
8573         \end{stex_annotation_env}
8574       }
8575     \else
8576       \egroup
8577     \fi
8578 }

```

```

8579 }{
860  \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
861  \noindent\emph{\problem@kw@solution\tl_if_empty:NF \l_stex_key_title_tl{%
862      \l_stex_key_title_tl \str_if_empty:NF \l_stex_key_answerclass_str {%
863          (Answer Class: \l_stex_key_answerclass_str)
864      }
865  } :~}
866 }{
867  \par\rule[.3em]{\linewidth}{0.4pt}\newline
868 }{}%
869
870 \stex_deactivate_macro:Nn \solution {sproblem~environments}

\startsolution
\stopsolution
8591 \cs_new_protected:Npn \startsolution{%
8592     \global\solutionstrue
8593 }
8594 \cs_new_protected:Npn \stopsolution{%
8595     \global\solutionsfalse
8596 }

(End of definition for \startsolution and \stopsolution. These functions are documented on page
108.)
```

hint (env.)

```

8597
8598 \stex_keys_define:nnnn{ problemenv }{}{}{id,title,style}
8599
8600 \cs_new_protected:Nn \__problems_hint_start:n {
8601     \stex_keys_set:nn{ problemenv }{\#1}
8602     \str_if_empty:NT \l_stex_key_id_str {
8603         \int_gincr:N \g_problem_id_counter
8604         \str_set:Nx \l_stex_key_id_str {
8605             HINT_ \int_use:N \g_problem_id_counter
8606         }
8607     }
8608     \stex_if_do_html:T{
8609         \begin{stex_annotation_env}%
8610             shtml:problemhint=\l_stex_key_id_str
8611         %
8612     }
8613     \stex_style_apply:
8614 }
8615
8616 \stex_new_stylable_env:nnnnnnn { hint }{ 0{} }{%
8617     \stex_if_do_html:TF{%
8618         \__problems_hint_start:n{\#1}
8619     }{%
8620         \bool_if:NTF \c__problems_hints_bool {
8621             \__problems_hint_start:n{\#1}
8622         }{%
8623             \setbox\l_tmpa_box\vbox\bgroup
8624         }
8625     }
8626 }
```

```

8626 }{
8627   \stex_if_do_html:TF{
8628     \stex_style_apply:
8629     \end{stex_annotation_env}
8630 }{
8631   \bool_if:NTF \c__problems_hints_bool {
8632     \stex_style_apply:
8633     \stex_if_do_html:T{
8634       \end{stex_annotation_env}
8635     }
8636   }{
8637     \egroup
8638   }
8639 }
8640 }{
8641   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8642   \noindent\emph{\problem@kw@hint\tl_if_empty:N \l_stex_key_title_tl{
8643     \{} \l_stex_key_title_tl
8644     \} : \}}
8645 }{
8646   \par\rule[.3em]{\linewidth}{0.4pt}\newline
8647 }{}
8648 \stex_deactivate_macro:Nn \hint {sproblem~environments}

exnote (env.)
8649 \cs_new_protected:Nn \__problems_exnote_start:n {
8650   \stex_keys_set:nn{ problemenv }{#1}
8651   \str_if_empty:NT \l_stex_key_id_str {
8652     \int_gincr:N \g_problem_id_counter
8653     \str_set:Nx \l_stex_key_id_str {
8654       EXNOTE_\int_use:N \g_problem_id_counter
8655     }
8656   }
8657   \stex_if_do_html:T{
8658     \begin{stex_annotation_env} {
8659       \shtml:problemnote=\l_stex_key_id_str
8660     }
8661   }
8662   \stex_style_apply:
8663 }
8664
8665 \stex_new_stylable_env:nnnnnnn { exnote }{ 0{} }{
8666   \stex_if_do_html:TF{
8667     \__problems_exnote_start:n{#1}
8668   }{
8669     \bool_if:NTF \c__problems_notes_bool {
8670       \__problems_exnote_start:n{#1}
8671     }{
8672       \setbox\l_tmpa_box\vbox\bgroup
8673     }
8674   }
8675 }{
8676   \stex_if_do_html:TF{
8677     \stex_style_apply:

```

```

8678     \end{stex_annotation_env}
8679 }
8680 \bool_if:NTF \c__problems_notes_bool {
8681     \stex_style_apply:
8682     \stex_if_do_html:T{
8683         \end{stex_annotation_env}
8684     }
8685 }
8686     \egroup
8687 }
8688 }
8689 }
8690 \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8691 \noindent\emph{\problem@kw@note\tl_if_empty:N \l_stex_key_title_tl{
8692     \~{}\l_stex_key_title_tl
8693 } :~}
8694 }
8695 \par\rule[.3em]{\linewidth}{0.4pt}\newline
8696 }{}}
8697 \stex_deactivate_macro:Nn \exnote {sproblem~environments}

gnote (env.)
8698 \int_new:N \l__problems_anscls_int
8699
8700 \cs_new_protected:Nn \__problems_gnote_start:n {
8701     \stex_keys_set:nn{ problemenv }{\#1}
8702     \str_if_empty:NT \l_stex_key_id_str {
8703         \int_gincr:N \g_problem_id_counter
8704         \str_set:Nx \l_stex_key_id_str {
8705             GNOTE_\int_use:N \g_problem_id_counter
8706         }
8707     }
8708
8709     \stex_if_do_html:T{
8710         \begin{stex_annotation_env} {
8711             shtml:problemgnote=\l_stex_key_id_str
8712         }
8713     }
8714     \stex_style_apply:
8715 }
8716
8717 \stex_new_stylable_env:nnnnnnn { gnote }{ 0{} }{
8718     \stex_if_do_html:TF{
8719         \__problems_gnote_start:n{\#1}
8720     }
8721     \bool_if:NTF \c__problems_gnotes_bool {
8722         \__problems_gnote_start:n{\#1}
8723     }
8724         \setbox\l_tmpa_box\vbox\bgroup
8725     }
8726 }
8727     \stex_reactivate_macro:N \anscls
8728 }
8729 \stex_if_do_html:TF{

```

```

8730     \stex_style_apply:
8731     \end{stex_annotation_env}
8732 }{
8733     \bool_if:NTF \c__problems_gnotes_bool {
8734         \stex_style_apply:
8735         \stex_if_do_html:T{
8736             \end{stex_annotation_env}
8737         }
8738     }{
8739         \egroup
8740     }
8741 }
8742 }{
8743     \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8744     \noindent\emph{\problem@kw@grading\str_if_empty:N \l_stex_key_title_tl{
8745         \~{}\l_stex_key_title_tl
8746     } :~}
8747 }{
8748     \par\rule[.3em]{\linewidth}{0.4pt}\newline
8749 }{}
8750 \stex_deactivate_macro:Nn \gnote {sproblem~environments}
8751
8752
8753 \stex_keys_define:nnnn{ anscls }{
8754     \str_clear:N \l_stex_key_pts_str
8755     \str_clear:N \l_stex_key_schema_str
8756 }{
8757     pts      .str_set:N = \l_stex_key_pts_str,
8758     schema   .str_set:N = \l_stex_key_schema_str,
8759     update   .code:n   = {}
8760 }{id}
8761 \newcommand \anscls [2][]{ {
8762     \stex_keys_set:nn{ anscls }{#1}
8763     \str_if_empty:NT \l_stex_key_id_str {
8764         \int_incr:N \l__problems_anscls_int
8765         \str_set:Nx \l_stex_key_id_str {
8766             AC\int_use:N \l__problems_anscls_int
8767         }
8768     }
8769     \begin{list}{}{
8770         \setlength\topsep{0pt}
8771         \setlength\parsep{0pt}
8772         \setlength\rightmargin{0pt}
8773     }\item[\l_stex_key_id_str]
8774     \stex_if_do_html:TF{
8775         \stex_annotation:nn{
8776             shtml:answerclass={\l_stex_key_id_str}
8777             \str_if_empty:N \l_stex_key_pts_str{
8778                 ,shtml:answerclass-pts={\l_stex_key_pts_str}
8779             }
8780             \str_if_empty:N \l_stex_key_schema_str{
8781                 ,shtml:answerclass-schema={\l_stex_key_schema_str}
8782             }
8783 }{

```

```

8784         #2
8785     }
8786   }{#2}
8787   \str_if_empty:NF \l_stex_key_schema_str {
8788     ~\emph{(\problem@kw@schema:\~\l_stex_key_schema_str)}
8789   }
8790   \str_if_empty:NF \l_stex_key_pts_str {
8791     ~\emph{(\problem@kw@points:\~\l_stex_key_pts_str)}
8792   }
8793   \end{list}
8794 }
8795 \stex_deactivate_macro:Nn \anscls {gnote~environments}

The margin pars are reader-visible, so we need to translate
8796 \def\pts#1{
8797   \bool_if:NT \c__problems_pts_bool {
8798     \stex_annotation:n{html:problempoints={}}{\marginpar{\#1~\problem@kw@pts}}
8799   }\hbox_unpack:N\c_empty_box
8800 }
8801 \def\min#1{
8802   \bool_if:NT \c__problems_min_bool {
8803     \stex_annotation:n{html:problemminutes={}}{\marginpar{\#1~\problem@kw@minutes}}
8804   }\hbox_unpack:N\c_empty_box
8805 }

mcb (env.)
8806 \stex_new_stylable_env:nnnnnnn{mcb}{0}{}{
8807   \stex_keys_set:nn{style}{#1}
8808   \stex_if_do_html:T{
8809     \tl_set:Nn\problem_mcc_box_tl{}
8810     \begin{stex_annotation_env}{shtml:multiple-choice-block={}}
8811   }
8812   \stex_deactivate_macro:Nn \mcb {sproblem~environments}
8813   \stex_deactivate_macro:Nn \scb {sproblem~environments}
8814   \stex_deactivate_macro:Nn \solution {sproblem~environments}
8815   \stex_deactivate_macro:Nn \hint {sproblem~environments}
8816   \stex_deactivate_macro:Nn \exnote {sproblem~environments}
8817   \stex_deactivate_macro:Nn \gnote {sproblem~environments}
8818   \stex_reactivate_macro:N \mcc
8819   \cs_set:Nn \__problems_mccline:n{
8820     \begin{list}{}{
8821       \setlength\topsep{0pt}
8822       \setlength\parsep{0pt}
8823       \setlength\rightmargin{0pt}
8824     }\item[\problem_mcc_box_tl] ##1 \end{list}
8825   }
8826   \stex_style_apply:
8827 }{
8828   \stex_style_apply:
8829   \stex_if_do_html:T{
8830     \end{stex_annotation_env}
8831   }
8832 }{\par}{}{
8833 \stexstylemcb[inline]{
```

```

8834   \cs_set:Nn \__problems_mccline:n{\problem_mcc_box_tl{~} #1}
8835 }{}
8836
8837 \stex_deactivate_macro:Nn \mcb {sproblem~environments}
    we define the keys for the mcc macro
8838 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8839   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8840     \bool_set_true:N #1
8841   }{
8842     \bool_set_false:N #1
8843   }
8844 }
8845 \stex_keys_define:nnnn{mcc}{
8846   \tl_clear:N \l_stex_key_feedback_tl
8847   \bool_set_false:N \l_stex_key_T_bool
8848   \tl_clear:N \l_stex_key_Ttext_tl
8849   \tl_clear:N \l_stex_key_Ftext_tl
8850 }{
8851   feedback .tl_set:N      = \l_stex_key_feedback_tl ,
8852   T       .code:n      = {\bool_set_true:N \l_stex_key_T_bool} ,
8853   F       .code:n      = {\bool_set_false:N \l_stex_key_T_bool} ,
8854   Ttext   .tl_set:N      = \l_stex_key_Ttext_tl ,
8855   Ftext   .tl_set:N      = \l_stex_key_Ftext_tl ,
8856 }{id}
8857
\mcc
8858 \tl_set:Nn \problem_mcc_box_tl {
8859   \ltx@ifpackageloaded{amssymb}{$\square$}{
8860     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
8861   }
8862 }
8863 \newcommand\mcc[2][]{%
8864   \stex_keys_set:nn{mcc}{#1}
8865   \tl_set:Nn \l_tmpa_tl{%
8866     \bool_if:NTF \l_stex_key_T_bool {
8867       \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
8868     }{
8869       \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
8870     }
8871     \tl_if_empty:NF \l_stex_key_feedback_tl {
8872       \\\emph{\l_stex_key_feedback_tl}
8873     }
8874   }
8875
8876 \__problems_mccline:n{
8877   \stex_if_do_html:TF{
8878     \stex_annotation:nn{shtml:mcc= {
8879       \bool_if:NTF \l_stex_key_T_bool {true}{false}
8880     }}{
8881       #2\stex_annotation:nn{shtml:mcc-solution={}{}{\l_tmpa_tl}}
8882     }
8883 }

```

```

8884         #2\ifsolutions\footnote{\l_tmpa_t1}\fi
8885     }
8886   }
8887 }
8888 \stex_deactivate_macro:Nn \mcc {mcb~environments}

(End of definition for \mcc. This function is documented on page 109.)
```

scb (env.)

```

8889
8890 \tl_set:Nn\problem_scc_box_tl{$\bigcirc$}
8891
8892 \stex_new_stylable_env:nnnnnn{scb}{0}{}{
8893   \stex_keys_set:nn{style}{#1}
8894   \stex_if_do_html:T{
8895     \begin{stex_annotation_env}{shtml:single-choice-block={}}
8896     \tl_set:Nn\problem_scc_box_tl{}
8897   }
8898   \stex_deactivate_macro:Nn \mcb {sproblem~environments}
8899   \stex_deactivate_macro:Nn \scb {sproblem~environments}
8900   \stex_deactivate_macro:Nn \solution {sproblem~environments}
8901   \stex_deactivate_macro:Nn \hint {sproblem~environments}
8902   \stex_deactivate_macro:Nn \exnote {sproblem~environments}
8903   \stex_deactivate_macro:Nn \gnote {sproblem~environments}
8904   \stex_reactivate_macro:N \scc
8905   \cs_set:Nn \__problems_sccline:n{
8906     \begin{list}{}{
8907       \setlength\topsep{0pt}
8908       \setlength\parsep{0pt}
8909       \setlength\rightmargin{0pt}
8910     }\item[\problem_scc_box_tl] ##1 \end{list}
8911   }
8912   \stex_style_apply:
8913 }{
8914   \stex_style_apply:
8915   \stex_if_do_html:T{
8916     \end{stex_annotation_env}
8917   }
8918 }{\par}{}}
8919 \stexstylescb[inline]{
8920   \cs_set:Nn \__problems_sccline:n{\problem_scc_box_tl{~} #1}
8921 }{}
8922
8923 \stex_deactivate_macro:Nn \scb {sproblem~environments}
```

\scc

```

8924
8925 \newcommand\scc[2][]{%
8926   \stex_keys_set:nn{mcc}{#1}
8927   \tl_set:Nn \l_tmpa_t1{
8928     \bool_if:NTF \l_stex_key_T_bool {
8929       \tl_if_empty:NTF \l_stex_key_Ttext_t1 \problem@kw@correct \l_stex_key_Ttext_t1
8930     }{
8931       \tl_if_empty:NTF \l_stex_key_Ftext_t1 \problem@kw@wrong \l_stex_key_Ftext_t1
```

```

8932     }
8933     \tl_if_empty:NF \l_stex_key_feedback_tl {
8934         \\\emph{\l_stex_key_feedback_tl}
8935     }
8936 }
8937
8938 \__problems_sccline:n{
8939     \stex_if_do_html:TF{
8940         \stex_annotate:nn{shtml:scc={
8941             \bool_if:NTF \l_stex_key_T_bool {true}{false}
8942         }){
8943             #2\stex_annotate:nn{shtml:scc-solution={}}{\l_tmpa_tl}
8944         }
8945     }{
8946         #2\ifsolutions\footnote{\l_tmpa_tl}\fi
8947     }
8948 }
8949 }
8950 \stex_deactivate_macro:Nn \scc {scb~environments}
8951
8952
8953 \newcommand\yesTnoF{
8954     \begin{scb}[style=inline]
8955         \scc[T]{yes}~\scc[F]{no}
8956     \end{scb}
8957 }
8958 \newcommand\yesFnoT{
8959     \begin{scb}[style=inline]
8960         \scc[F]{yes}~\scc[T]{no}
8961     \end{scb}
8962 }
8963 \newcommand\trueTfalseF{
8964     \begin{scb}[style=inline]
8965         \scc[T]{true}~\scc[F]{false}
8966     \end{scb}
8967 }
8968 \newcommand\trueFfalseT{
8969     \begin{scb}[style=inline]
8970         \scc[F]{true}~\scc[T]{false}
8971     \end{scb}
8972 }

```

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

\fillinsol

```

8973 \stex_keys_define:nnnn{fillinsol}{
8974     \tl_clear:N \l__problems_fillin_solution_tl
8975     \dim_zero:N \l_stex_key_testsphere_dim
8976 }{
8977     testspace .dim_set:N = \l_stex_key_testsphere_dim,
8978     exact .code:n = {\__problems_parse_fillin_arg:nnnn{exact}#1 },
8979     numrange .code:n = {\__problems_parse_fillin_arg:nnnn{numrange}#1 },
8980     regex .code:n = {\__problems_parse_fillin_arg:nnnn{regex}#1 }
8981 }{}

```

```

8982 \cs_new:Nn \__problems_parse_fillin_arg:nnnn {
8983   \stex_if_do_html:TF{
8984     \tl_set:Nn \l_tmpa_tl{\#3}
8985     \tl_set:Nn \l_tmpb_tl{T}
8986     \stex_annotate_invisible:nn{
8987       shtml:fillin-case={#1},
8988       shtml:fillin-case-value={#2},
8989       shtml:fillin-case-verdict={
8990         \tl_if_eq:NNTF\l_tmpa_tl\l_tmpb_tl{true}{false}
8991       },
8992     }{#4}
8993   }{
8994     \tl_put_right:Nn \l__problems_fillin_solution_tl {
8995       #1{~} & #2{~} & #3{~} & #4{~} \\
8996     }
8997   }
8998 }
8999 }

9000 \newcommand\fillinsol[2][]{

9001   \quad
9002   \mode_if_math:TF{
9003     \hbox{\__problems_fillinsol:nn{#1}{##2$}}
9004   }{
9005     \__problems_fillinsol:nn{#1}{#2}
9006   }
9007   \quad
9008 }

9009 }

9010 \cs_new_protected:Nn \__problems_fillinsol:nn {

9011   \stex_keys_set:nn{fillinsol}{#1}
9012   \stex_if_do_html:TF{
9013     \stex_annotate:nn{shtml:fillinsol={}}{ \stex_annotate_force_break:n{
9014       #2
9015       \l__problems_fillin_solution_tl
9016     } }
9017   }{
9018     \ifsolutions
9019       \textcolor{red}{\fbox{#2}}
9020       \tl_if_empty:NF \l__problems_fillin_solution_tl {
9021         \footnote{
9022           \halign{ ##\hfil & ##\hfil & ##\hfil&##\hfil \cr
9023             \textbf{type }&\textbf{case }&\textbf{verdict }&\textbf{feedback }\cr
9024             \l__problems_fillin_solution_tl
9025           }
9026         }
9027       }
9028     \else
9029       \fbox{\dim_compare:nNnTF\l_stex_key_testsphere_dim={0pt}{

9030         \phantom{\huge{#2}}
9031       }{
9032         \hspace{\l_stex_key_testsphere_dim}
9033       }}
9034     \fi
9035   }

```

```

9036 }
9037 \stex_deactivate_macro:Nn \fillinsol {sproblem-environments}

```

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

`\testemptypage`

```

9038 \newcommand\testemptypage[1] []{%
9039 \bool_if:NT \c__problems_test_bool {\vfill\begin{center}\hwexam@kw@testemptypage\end{center}}
9040 }

```

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

`\testspace`

```

9041 \newcommand\testspace[1]{\bool_if:NT \c__problems_test_bool {\vspace*{#1}}}
9042 \newcommand\testsmallspace{\testspace{1cm}}
9043 \newcommand\testmedspace{\testspace{2cm}}
9044 \newcommand\testbigspace{\testspace{3cm}}

```

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

`\testnewpage`

```

9045 \newcommand\testnewpage{\bool_if:NT \c__problems_test_bool {\newpage}}

```

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

```

9046 
```

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.

```

9047 <package>
9048 \ProvidesExplPackage{hwexam}{2023/03/19}{3.3.0}{homework assignments and exams}
9049 \RequirePackage{l3keys2e}
9050
9051 \keys_define:nn {hwexam / pkg}{
9052 multiple .default:n = { false },
9053 multiple .bool_set:N = \c_hwexam_multiple_bool,
9054 unknown .code:n = {
9055 \PassOptionsToPackage{\CurrentOption}{problem}
9056 }
9057 }
9058 \ProcessKeysOptions{ hwexam /pkg }
9059 \RequirePackage{problem}

```

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


```

9060 \AddToHook{begindocument}{
9061 \ExplSyntaxOn\makeatletter
9062 \input{hwexam-english.ldf}
9063 \ltx@ifpackageloaded{babel}{
```

```

9064 \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
9065 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
9066 \input{hwexam-ngerman.ldf}
9067 }
9068 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
9069 \input{hwexam-finnish.ldf}
9070 }
9071 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
9072 \input{hwexam-french.ldf}
9073 }
9074 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
9075 \input{hwexam-russian.ldf}
9076 }
9077 }{}
9078 \makeatother\ExplSyntaxOff
9079 }

```

(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.)
9080 \stex_keys_define:nnnn{ assignment }{
9081   \tl_clear:N \l_stex_key_number_tl
9082 \tl_clear:N \l_stex_key_given_tl
9083 \tl_clear:N \l_stex_key_due_tl
9084 }{
9085   number .tl_set:N     = \l_stex_key_number_tl,
9086   given  .tl_set:N     = \l_stex_key_given_tl,
9087   due    .tl_set:N     = \l_stex_key_due_tl,
9088 unknown .code:n = { }
9089 }{id,title,style}
9090
9091 \newcounter{assignment}
9092 \stex_new_stylable_env:nnnnnn {assignment}{0{}){
9093   \cs_if_exist:NTF \l_hwexam_includeassignment_keys_tl {
9094     \tl_put_left:Nn \l_hwexam_includeassignment_keys_tl {#1,}
9095     \exp_args:Nno \stex_keys_set:nn{assignment}{}{
9096       \l_hwexam_includeassignment_keys_tl
9097     }
9098   }{
9099     \stex_keys_set:nn{assignment}{#1}
9100   }
9101 \tl_if_empty:NF \l_stex_key_number_tl {
9102 \global\setcounter{assignment}{\int_eval:n{\l_stex_key_number_tl-1}}
9103 }
9104 \global\refstepcounter{assignment}
9105 \setcounter{sproblem}{0}
9106 \def\thesproblem{\theassignment.\arabic{sproblem}}
9107 \stex_style_apply:

```

```

9108 \_stex_do_id:
9109 }{
9110 \stex_style_apply:
9111 }{
9112 \par\begin{center}
9113 \textbf{\Large\assignmentautorefname~\theassignment}
9114 \tl_if_empty:NF \l_stex_key_title_tl {
9115 {\~}---\l_stex_key_title_tl
9116 }
9117 }\par\smallskip
9118 \textbf{
9119 \tl_if_empty:NF \l_stex_key_given_tl {
9120 \hwexam@kw@given :~\l_stex_key_given_tl\quad
9121 }
9122 \tl_if_empty:NF \l_stex_key_due_tl {
9123 \hwexam@kw@due :~\l_stex_key_due_tl\quad
9124 }
9125 }
9126 \end{center}
9127 \par\bigskip
9128 }{
9129 \par\pagebreak
9130 }{}}

\includeassignment
9131 \NewDocumentCommand\includeassignment{O{} m}{
9132 \group_begin:
9133 \tl_set:Nn \l_hwexam_includeassignment_keys_tl {#1}
9134 \stex_keys_set:nn{includeproblem}{#1}
9135 \exp_args:Nno \use:nn{\inputref[]}\l_stex_key_mhrepos_str]{#2}
9136 \group_end:
9137 }

```

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

Restoring information about problems:

```

9138 \prop_new:N \c_@@_problems_prop
9139 \tl_set:Nn \c_@@_total_mins_tl {0}
9140 \tl_set:Nn \c_@@_total_pts_tl {0}
9141 \int_new:N \c_@@_total_problems_int
9142 \cs_set_protected:Npn \problem@restore #1 #2 #3 {
9143 \int_gincr:N \c_@@_total_problems_int
9144 \prop_gput:Nnn \c_@@_problems_prop {#1}{#2}{#3}
9145 \tl_gset:Nx \c_@@_total_pts_tl { \int_eval:n { \c_@@_total_pts_tl + #2 } }
9146 \tl_gset:Nx \c_@@_total_mins_tl { \int_eval:n { \c_@@_total_mins_tl + #2 } }
9147 }

```

\correction@table This macro generates the correction table

```

9148 \newcommand\correction@table{
9149 \int_compare:nNnT \c_@@_total_problems_int = 0 {
9150 \int_incr:N \c_@@_total_problems_int
9151 \prop_put:Nnn \c_@@_problems_prop {\~}{\~}{\~}
9152 }
9153 \tl_clear:N \l_tmpa_tl
9154 \tl_clear:N \l_tmpb_tl

```

```

9155 \tl_clear:N \l_tmpc_tl
9156 \prop_map_inline:Nn \c_@@_problems_prop {
9157 \tl_put_right:Nn \l_tma_t1 { ##1 & }
9158 \tl_put_right:Nx \l_tmfp_t1 { \use_i:nn ##2 & }
9159 \tl_put_right:Nn \l_tmfp_t1 { & }
9160 }
9161 \resizebox{\textwidth}{!}{%
9162 \exp_args:Nne \begin{tabular}{|l|*\int_use:N \c_@@_total_problems_int}{c|}c||l|}\hline
9163 &\exp_args:Ne \multicolumn{\int_eval:n{ \c_@@_total_problems_int + 1}}{c||}
9164 {\footnotesize\hwexam@kw@forgrading} &\hline
9165 \hwexam@kw@probs & \l_tma_t1 \hwexam@kw@sum & \hwexam@kw@grade\\ \hline
9166 \hwexam@kw@pts & \l_tmfp_t1 \c_@@_total_pts_t1 & \\ \hline
9167 \hwexam@kw@reached & \l_tmfp_t1 & \hline
9168 \end{tabular}}}

```

(End of definition for \correction@table. This function is documented on page ??.)

\testheading

```

9169 \def\hwexamheader{\input{hwexam-default.header}}
9170
9171 \def\hwexamminutes{
9172 \tl_if_empty:NTF \hwexam@duration {
9173 {\hwexam@min}~\hwexam@minutes@kw
9174 }{
9175 \hwexam@duration
9176 }
9177 }
9178
9179 \stex_keys_define:nnnn{ hwexam / testheading }{
9180 \tl_clear:N \hwexam@min
9181 \tl_clear:N \hwexam@duration
9182 \tl_clear:N \hwexam@reqpts
9183 \tl_clear:N \hwexam@tools
9184 }{
9185 min .tl_set:N = \hwexam@min,
9186 duration .tl_set:N = \hwexam@duration,
9187 reqpts .tl_set:N = \hwexam@reqpts,
9188 tools .tl_set:N = \hwexam@tools
9189 }{}}
9190
9191 \newenvironment{testheading}[1][]{%
9192 \stex_keys_set:nn { hwexam / testheading}{#1}
9193
9194 \tl_set_eq:NN \hwexam@totalpts \c_@@_total_pts_t1
9195 \tl_set_eq:NN \hwexam@totalmin \c_@@_total_mins_t1
9196 \tl_set:Nx \hwexam@checktime {\int_eval:n { \hwexam@min - \hwexam@totalmin }}}
9197
9198 \newif\if@bonuspoints
9199 \tl_if_empty:NTF \hwexam@reqpts {
9200 \@bonuspointsfalse
9201 }{
9202 \tl_set:Nx \hwexam@bonuspts {
9203 \int_eval:n{\hwexam@totalpts - \hwexam@reqpts}
9204 }

```

```

9205 \@bonuspointstrue
9206 }
9207
9208 \makeatletter\hwexamheader\makeatother
9209 }{
9210 \newpage
9211 }

```

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

```

9212 </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

```

9213 <@@=tikzinput>
9214 <*package>
9215
9216 %%%%%%%%%% tikzinput.dtx %%%%%%
9217
9218 \ProvidesExplPackage{tikzinput}{2023/03/19}{3.3.0}{tikzinput package}
9219 \RequirePackage{l3keys2e}
9220
9221 \keys_define:nn { tikzinput } {
9222   image .bool_set:N    = \c_tikzinput_image_bool,
9223   image .default:n     = false ,
9224   unknown .code:n      = {}
9225 }
9226
9227 \ProcessKeysOptions { tikzinput }
9228
9229 \bool_if:NTF \c_tikzinput_image_bool {
9230   \RequirePackage{graphicx}
9231

```

```

9232   \providecommand\usetikzlibrary[]{}
9233   \newcommand\tikzininput[2][]{\includegraphics[#1]{#2}}
9234 }{
9235   \RequirePackage{tikz}
9236   \RequirePackage{standalone}
9237
9238   \newcommand\tikzininput[2][]{%
9239     \setkeys{Gin}{#1}
9240     \ifx\Gin@ewidth\Gin@exclamation
9241       \ifx\Gin@eheight\Gin@exclamation
9242         \input{#2}
9243       \else
9244         \resizebox{!}{\Gin@eheight}{\input{#2}}
9245       \fi
9246     \else
9247       \ifx\Gin@eheight\Gin@exclamation
9248         \resizebox{\Gin@ewidth}{!}{\input{#2}}
9249       \else
9250         \resizebox{\Gin@ewidth}{\Gin@eheight}{\input{#2}}
9251       \fi
9252     \else
9253       \ifx\Gin@eheight\Gin@exclamation
9254         \input{#2}
9255       \else
9256         \ifx\Gin@ewidth\Gin@eheight
9257           \input{#2}
9258         \else
9259           \begin{center}
9260             \tikzininput[#1]{#2}
9261           \end{center}
9262         \end{center}
9263       \else
9264         \tikzininput[#1]{#2}
9265       \end{center}
9266     \fi
9267   }

```

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	
\\$	2086, 2089, 2093
\;	7495, 7526, 7539, 7541, 7554
\@ commands:	
\c_@_problems_prop	9138, 9144, 9151, 9156
\c_@_total_mins_tl ..	9139, 9146, 9195
\c_@_total_problems_int	9141, 9143, 9149, 9150, 9162, 9163
\c_@_total_pts_tl	9140, 9145, 9166, 9194
\\	53, 77, 1055, 7826, 7831, 8186, 8189, 8872, 8934, 8996, 9164, 9165, 9166, 9167
\{	7544
\}	7544
_	44, 640, 7858, 9039
_comp	3949, 4155, 4496, 5192, 6011, 6035, 6119
_customthiscomp	6538, 6539, 6551
_defcomp	6043, 6977, 6986
_stex_html_do_output_bool	279, 280, 283, 288, 292, 323, 326, 2155, 7486
_thiscomp	6521, 6532, 6533
_varcomp	4477, 4776, 4785, 4942, 4980, 5226, 5668, 5967, 5980, 5993, 6039
\ 	2085, 2088, 2092
A	
\activateexcursion	106
\addbibresource	77, 1955
\addcontentsline	7856
\addmhbibresource	77, 1949
\addtocounter	8032, 8410, 8411
\AddToHook	1037, 1040, 7290, 7357, 7360, 7472, 8301, 9060
\aftergroup	140, 5180, 7355, 7469
\afterprematurestop	105, 7915, 7926
\anscls	8727, 8761, 8795
\apply	85
\arabic	8342, 8345, 9106
\arg	46, 90, 5405
\argarraymap	89, 4216, 4716
\argmap	89, 4215, 4357, 4692, 7527
\argsep	40, 89, 4214, 4352, 4676, 7495, 7496, 7497, 7503
\assign	64, 126, 2991, 3347, 3474
\assignment (env.)	71, 113, 9080
\assignmentautorefname	9113
\assignMorphism	126, 2992, 3476
\assumption	7268, 7373
\ast	7518
\AtBeginDocument	1007, 1260, 1446, 1449, 1497, 7727, 7870
\AtEndDocument	648, 1498
\AtEndOfPackageFile	2036, 2050, 2063
\author	8166
\autoref	81, 1642
B	
\backmatter	1461, 1462, 1463
\baselineskip	8155
\beameritemnestingprefix	8240
\begin	93, 132, 133, 138, 198, 1303, 1383, 2047, 2060, 2078, 2235, 2257, 2274, 2524, 2782, 2828, 4739, 6872, 7182, 7215, 7237, 7275, 7346, 7418, 7614, 7859, 7992, 8006, 8016, 8049, 8070, 8078, 8116, 8117, 8118, 8119, 8120, 8121, 8127, 8129, 8201, 8229, 8230, 8382, 8471, 8493, 8545, 8609, 8658, 8710, 8769, 8810, 8820, 8895, 8906, 8954, 8959, 8964, 8969, 9039, 9112, 9162, 9263
\begingroup	359, 1927
\bf	8154
\bfseries	8455
\bgroup	3792, 4822, 5034, 8005, 8094, 8109, 8186, 8189, 8562, 8623, 8672, 8724
\bigcirc	8890
\bigskip	8430, 9127
\blindfragment (env.)	79
bool commands:	
\bool_if:NTF	197, 619, 671, 672, 678, 1145, 1155, 1157, 1216, 1237, 1450, 2138, 2393, 2743, 2776, 2960, 3456, 3665, 4837, 4940, 5190, 5224, 5766, 5860, 5866, 6001, 6967, 7243, 7291, 7329, 7348, 7351, 7353, 7358, 7361, 7369, 7411, 7425, 7447, 7473, 7762, 7770, 7807, 7886, 7930, 8091, 8097,

```

8131, 8146, 8165, 8200, 8210, 8288,
8294, 8364, 8385, 8417, 8422, 8431,
8453, 8474, 8483, 8486, 8499, 8500,
8504, 8505, 8620, 8631, 8669, 8680,
8721, 8733, 8797, 8802, 8866, 8879,
8928, 8941, 9039, 9041, 9045, 9229
\bool_if:nTF ..... 283
\bool_if_exist:NTF ..... 256, 272
\bool_lazy_any:nTF ..... 4044, 4084
\bool_lazy_any_p:n ..... 774
\bool_new:N ..... 279, 2135, 2378, 2801, 4748, 5160,
5735, 5857, 8321, 8359, 8360, 8361
\bool_not_p:n ..... 773, 777
\bool_set_false:N .. 182, 288, 326,
1142, 1166, 2136, 2155, 2379, 2756,
2823, 3429, 4608, 4751, 5164, 5736,
5861, 7101, 7250, 7298, 7326, 7467,
7486, 7487, 7746, 7787, 7946, 8327,
8362, 8399, 8402, 8842, 8847, 8853
\bool_set_true:N ... 187, 258, 274,
280, 292, 323, 608, 614, 1139, 2154,
2740, 2820, 3441, 3451, 3662, 3793,
4842, 5082, 5161, 5390, 5399, 5515,
5526, 5624, 5709, 5767, 5778, 5811,
5840, 5886, 6082, 6493, 6524, 6541,
6586, 7113, 7280, 7759, 7799, 7944,
7952, 7953, 7954, 7955, 7956, 7957,
8395, 8400, 8403, 8458, 8840, 8852
\bool_while_do:Nn ..... 1140
\bool_while_do:nn ..... 772, 1987, 7117, 7129, 7141, 7158, 7170
\l_tmpa_bool .. 3429, 3441, 3451, 3456
box commands:
\box_clear:N ..... 2158
\c_empty_box .. 3974, 3981, 8799, 8804
\l_tmpa_box 2153, 2158, 3677, 4207,
8094, 8109, 8562, 8623, 8672, 8724
boxed ..... 107

C
\catcode ..... 44, 53, 359, 640, 1055,
1695, 2084, 2085, 2086, 2087, 2088,
2089, 2091, 2092, 2093, 2435, 2436
\cdot ..... 7526, 7539
\centering ..... 7860, 8050
\chapter ..... 27, 79, 7895, 7896
\chaptername ..... 7831, 7892, 7893
\chaptertitlename ..... 7831
\circ ..... 54
\clearpage ..... 1456, 1466, 1477, 1488
clist commands:
\clist_clear:N 94, 406, 3705, 4133,
5083, 6375, 6474, 6789, 6794, 7375
\clist_count:N ..... 6355, 6365
\clist_count:n ..... 4724, 6457
\clist_get>NN ..... 462, 513
\clist_if_empty:NTF .... 181, 458,
509, 3809, 4305, 6354, 6400, 6657, 6861
\clist_if_in:NnTF ..... 66, 69, 100, 6955, 8306, 8309,
8312, 8315, 9065, 9068, 9071, 9074
\clist_if_in:nnTF ..... 6483
\clist_item:Nn ..... 4314
\clist_item:nn ..... 4735
\clist_map_function>NN ... 5084, 6403
\clist_map_function:nN ..... 2693, 2718, 3596, 3616, 3637
\clist_map_inline:Nn ..... 95, 104, 184, 2845, 3811, 4860,
5053, 6182, 6232, 6658, 6771, 6807
\clist_map_inline:nn ..... 370, 419, 5556, 5655, 5684, 6180, 7057
\clist_pop>NN ..... 4316
\clist_put_right:Nn ..... 96, 5097, 5101, 6382, 6501, 6503
\clist_set:Nn 38, 92, 4961, 8305, 9064
\clist_set_eq>NN ..... 90, 6877
\l_tmpa_clist 8305, 8306, 8309, 8312,
8315, 9064, 9065, 9068, 9071, 9074
\clstinputmhlisting ..... 80, 2035
\cmhgraphics ..... 80, 2035
\cmhtikzinput ..... 115, 2035
\columnbox ..... 8068, 8070, 8088
\comp ..... 32, 33, 36, 46, 87, 90,
100, 128, 3949, 4155, 4328, 4477,
4496, 4538, 4544, 4546, 4776, 4785,
4980, 5209, 5210, 5544, 5668, 5888,
5893, 5908, 5967, 5980, 5993, 6011,
6013, 6020, 6119, 6342, 6350, 6533,
6547, 6548, 6977, 6978, 6986, 7492,
7494, 7501, 7503, 7505, 7506, 7511,
7513, 7518, 7520, 7521, 7528, 7529,
7530, 7531, 7534, 7536, 7540, 7541,
7542, 7544, 7554, 7561, 7563, 7565
\compemph ..... 100, 6020
\conclude ..... 7267, 7373
\conclusion 50–52, 96, 98, 6946, 7066, 7082
\copymod ..... 62, 63, 3606, 3608, 3610
\copymodule ..... 3493, 3495
\counterwithin ..... 7899, 7911
\cr ..... 9022, 9023
cs commands:
\cs:w .... 445, 448, 480, 641, 2168,
2397, 2398, 2399, 2406, 2410, 2416,
5288, 6223, 6273, 6275, 6895, 6908
\cs_argument_spec:N ..... 3998

```

```

\cs_end: . 445, 448, 480, 641, 2168,
2397, 2398, 2399, 2408, 2412, 2416,
5288, 6223, 6273, 6275, 6895, 6908
\cs_generate_from_arg_count:NNnn
..... 3802,
4604, 4851, 5044, 5128, 5329, 5361
\cs_generate_variant:Nn .....
.... 87, 157, 231, 244, 558, 582,
591, 603, 627, 687, 716, 871, 876,
880, 964, 1126, 1151, 1555, 1883,
2200, 2558, 2562, 2567, 2585, 2634,
2656, 2669, 3124, 5185, 5202, 5519
\cs_gset:Npn ..... 2431
\cs_if_eq:NNTF ..... 140,
886, 940, 1422, 2224, 2235, 2238,
2257, 2260, 4000, 4677, 5600, 5602
\cs_if_exist:NTF ..... 309,
1094, 1328, 1332, 1370, 1373, 1397,
1401, 1430, 1436, 1451, 1461, 1474,
1485, 1641, 1789, 1813, 1833, 1839,
1843, 2106, 2646, 2651, 3976, 3983,
3995, 4426, 4434, 4875, 4882, 5069,
5115, 5279, 5305, 5649, 5888, 5893,
6439, 6736, 6739, 6743, 6746, 6751,
6754, 6758, 6761, 7826, 7831, 7834,
7838, 7842, 7846, 7850, 7892, 7895,
7898, 7904, 7907, 7910, 8052, 8060,
8072, 8082, 8186, 8189, 8367, 9093
\cs_new:Nn ..... 209, 432, 546,
552, 559, 568, 584, 593, 765, 885,
2209, 2392, 2440, 2951, 3772, 4103,
4116, 4320, 4387, 4404, 4409, 4553,
5124, 5337, 5434, 6160, 6215, 6411,
6417, 6420, 6430, 6852, 8347, 8983
\cs_new:Npn 544, 621, 622, 630, 631,
881, 2892, 2923, 4109, 4122, 5580, 5953
\cs_new_nopar:Nn ..... 366, 381
\cs_new_protected:Nn 43, 52, 60, 65,
75, 81, 84, 136, 158, 216, 227, 237,
246, 264, 286, 329, 335, 349, 355,
437, 444, 457, 471, 479, 498, 508,
523, 545, 624, 634, 646, 651, 670,
682, 693, 709, 718, 733, 791, 798,
828, 863, 873, 877, 899, 916, 929,
935, 939, 966, 970, 986, 1002, 1008,
1021, 1086, 1093, 1116, 1127, 1136,
1153, 1163, 1172, 1199, 1246, 1254,
1300, 1308, 1312, 1322, 1368, 1381,
1391, 1428, 1508, 1518, 1544, 1584,
1618, 1630, 1640, 1648, 1688, 1746,
1757, 1788, 1794, 1798, 1812, 1819,
1861, 1886, 1902, 1926, 1937, 1949,
1976, 2006, 2080, 2096, 2119, 2123,
2127, 2142, 2146, 2152, 2163, 2171,
2201, 2223, 2229, 2247, 2270, 2279,
2295, 2302, 2315, 2328, 2343, 2351,
2360, 2381, 2403, 2424, 2446, 2450,
2458, 2470, 2555, 2559, 2563, 2572,
2580, 2586, 2604, 2611, 2623, 2637,
2644, 2657, 2671, 2688, 2697, 2708,
2713, 2722, 2733, 2738, 2762, 2780,
2794, 2802, 2843, 2852, 2864, 2868,
2880, 2888, 2896, 2911, 2924, 2970,
2976, 2997, 3040, 3073, 3140, 3159,
3179, 3185, 3190, 3217, 3248, 3273,
3299, 3349, 3393, 3460, 3543, 3571,
3581, 3676, 3684, 3745, 3757, 3776,
3818, 3826, 3880, 3889, 3971, 3980,
3985, 3992, 4013, 4030, 4043, 4054,
4056, 4069, 4080, 4172, 4188, 4239,
4251, 4278, 4299, 4304, 4325, 4336,
4345, 4455, 4486, 4528, 4557, 4564,
4585, 4599, 4635, 4640, 4645, 4676,
4749, 4793, 4807, 4820, 4870, 4888,
4901, 4907, 4917, 4925, 4988, 5002,
5015, 5016, 5018, 5062, 5095, 5114,
5126, 5138, 5167, 5189, 5204, 5223,
5248, 5253, 5258, 5325, 5341, 5347,
5382, 5388, 5396, 5451, 5512, 5521,
5537, 5550, 5584, 5595, 5616, 5648,
5680, 5726, 5729, 5739, 5752, 5765,
5788, 5789, 5790, 5794, 5820, 5823,
5849, 5859, 5947, 5999, 6020, 6114,
6139, 6146, 6163, 6169, 6218, 6278,
6286, 6293, 6298, 6352, 6374, 6381,
6385, 6448, 6473, 6499, 6531, 6537,
6556, 6568, 6579, 6597, 6607, 6647,
6656, 6664, 6673, 6683, 6694, 6708,
6732, 6814, 6866, 6912, 6925, 6963,
7034, 7050, 7126, 7214, 7221, 7225,
7236, 7251, 7368, 7391, 7409, 7724,
7820, 7891, 7903, 7942, 7980, 8004,
8009, 8015, 8024, 8096, 8220, 8442,
8536, 8600, 8649, 8700, 8838, 9010
\cs_new_protected:Npn ..... 206, 338, 341, 345,
346, 429, 689, 892, 895, 1242, 1326,
1395, 1411, 1499, 1501, 1832, 1838,
1842, 1857, 2215, 2298, 2374, 2857,
2917, 2955, 2983, 3014, 3028, 3111,
3126, 3260, 3589, 4424, 4570, 4577,
4627, 4683, 4692, 4717, 5068, 5081,
5146, 5152, 5156, 5186, 5236, 5277,
5303, 5428, 5564, 5878, 5898, 5905,
6035, 6039, 6043, 6049, 6053, 6057,
6061, 6065, 6069, 6073, 6077, 6081,
6319, 6341, 6346, 6437, 6456, 6469,
6521, 6768, 6805, 7115, 7139, 7156,
```

7168, 7211, 7574, 8448, 8591, 8594
`\cs_parameter_spec:N` 548, 562
`\cs_prefix_spec:N` 560
`\cs_set:Nn` 8819, 8834, 8905, 8920
`\cs_set:Npn` 449, 484, 489,
 637, 638, 820, 821, 822, 823, 944,
 951, 1095, 1098, 1111, 2177, 2426,
 2459, 2673, 2675, 2678, 2691, 2716,
 2741, 2755, 3803, 4016, 4209, 4210,
 4214, 4215, 4216, 4328, 4358, 4588,
 4604, 4698, 4723, 4852, 4890, 5045,
 5128, 5330, 5362, 5682, 6476, 6482,
 6543, 7483, 7484, 7485, 7802, 8101
`\cs_set:Npx` 2615
`\cs_set_eq:NN` 41, 218, 222,
 814, 815, 816, 1262, 1269, 1387,
 2652, 2994, 2995, 3996, 4602, 4655,
 4668, 4883, 5116, 5119, 5193, 5195,
 5227, 5229, 5405, 5408, 5436, 5437,
 5650, 5653, 5792, 7650, 7662, 8144
`\cs_set_protected:Nn`
 811, 818, 1568, 7821, 7854
`\cs_set_protected:Npn`
 1281, 1290, 3972, 3973, 4352, 4357,
 4363, 6047, 7651, 7893, 7896, 7905,
 7908, 8145, 8242, 8245, 8248, 9142
`\cs_set_protected:Npx` 6539
`\cs_undefine:N` ... 213, 223, 270, 1207
`\l_tmpa_cs` 4209, 4219, 5682, 5700
`\csname` 268,
 297, 356, 2082, 2083, 2084, 2085,
 2086, 2091, 2092, 2093, 2396, 2659
`\ctikzinput` 115, 9262
`\CurrentFile` 1024, 1026, 1028
`\CurrentFilePath` 1024, 1025, 1028
`\currentgrouplevel` . 228, 250, 251, 252,
 254, 256, 258, 266, 268, 270, 272, 274
`\CurrentOption` 10, 7749, 7750,
 7751, 7752, 7791, 7792, 8282, 9055
`\Currentsectionlevel` 79, 1290
`\currentsectionlevel` 79, 1281

D

`\DeclareOption` 10
`\def` 88, 138,
 303, 307, 310, 312, 358, 561, 1244,
 1248, 1265, 1270, 2052, 2055, 2066,
 2070, 3949, 4155, 4176, 4475, 4477,
 4494, 4496, 4776, 4783, 4785, 4978,
 4980, 5209, 5210, 5668, 5908, 5937,
 5951, 5967, 5980, 5993, 6011, 6022,
 6119, 6546, 6551, 6977, 6986, 7914,
 7916, 7981, 7982, 7983, 7987, 7990,
 8033, 8136, 8153, 8161, 8166, 8167,

8169, 8171, 8173, 8175, 8176, 8178,
 8180, 8182, 8185, 8186, 8188, 8189,
 8192, 8194, 8196, 8240, 8342, 8345,
 8346, 8796, 8801, 9106, 9169, 9171
`\defemph` 100, 6020
`\Definename` 97, 6921, 6951, 7004, 7010
`\define` 24, 36,
 45, 96, 97, 100, 6920, 6950, 6996, 7002
`\definiendum` 24, 36,
 45, 96, 97, 100, 6918, 6948, 6990, 6994
`\definiendum` 6963
`\definiens` 47,
 57, 64, 96–98, 6936, 7013, 7031, 7032
`\defnotation`
 ... 36, 45, 97, 6919, 6949, 6985, 6988
`\detokenize`
 ... 247, 352, 356, 1661, 8306, 8309,
 8312, 8315, 9065, 9068, 9071, 9074
dim commands:
 `\dim_compare:nNnTF` 9029
 `\dim_new:N` 8527
 `\dim_zero:N` 8530, 8975
`\dimexpr` 8132
do commands:
 `_do_comp:nNn`
 ... 4364, 6020, 6036, 6040, 6044
`\dobracket` 89
`\dobrackets` 89, 5853, 5906
`\dowithbrackets` 5905
`\due` 71, 113
`\duration` 71, 113

E

`\edef` ... 117, 2084, 2085, 2086, 6533, 7998
`\egroup`
 3816, 4867, 5060, 8012, 8094, 8111,
 8186, 8189, 8576, 8637, 8686, 8739
`\eject` 9039
`\ellipses` 92,
 93, 4702, 4703, 5096, 5098, 5656, 5694
`\else` 296, 308, 1890,
 1973, 2211, 5883, 6062, 8240, 8454,
 8559, 8575, 9028, 9243, 9248, 9253
`\emph` 17, 19, 99,
 6070, 7198, 7286, 7303, 8581, 8642,
 8691, 8744, 8788, 8791, 8872, 8934
`\end` 133, 138, 1309,
 1363, 2047, 2060, 2078, 2238, 2260,
 2283, 2543, 2798, 2876, 4741, 6889,
 7196, 7222, 7284, 7301, 7352, 7354,
 7432, 7436, 7638, 7864, 7918, 7927,
 8000, 8012, 8026, 8066, 8080, 8088,
 8101, 8103, 8116, 8117, 8118, 8119,
 8120, 8121, 8142, 8203, 8237, 8238,

8414, 8491, 8509, 8568, 8573, 8629,
 8634, 8678, 8683, 8731, 8736, 8793,
 8824, 8830, 8910, 8916, 8956, 8961,
 8966, 8971, 9039, 9126, 9168, 9265
`\endcsname` 268, 296, 297,
 356, 2082, 2083, 2084, 2085, 2086,
 2091, 2092, 2093, 2396, 2659, 8240
`\endgroup` 361, 363, 1934
`\endinput` 1769, 2285
`\ensuremath` 5768, 5779
 environments:
 `assignment` 71, 113, 9080
 `blindfragment` 79
 `exnote` 1, 8649
 `extstructure` 93, 6178
 `extstructure*` 93
 `frame` 1, 7942
 `gnote` 1, 8698
 `hint` 1, 8597
 `mathstructure` 93, 6092
 `mcb` 1, 8806
 `mmtinterface` 7604
 `nassertion` 1
 `ndefinition` 1
 `nexample` 1
 `note` 1
 `nparagraph` 1, 1
 `nsproof` 1
 `problem` 1
 `sassertion` 96
 `scb` 8889
 `sdefinition` 96
 `sexample` 96
 `sfragment` 79
 `smodule` 83, 2515
 `solution` 1, 8526
 `sparagraph` 96
 `sproblem` 8347
 `sproof` 98, 7214
 `stex_annotation_env` 138
 `subproblem` 8449
 `subproof` 7307
 `testheading` 71, 113
`\eq` 32, 42
`\eqstep` 7269, 7373
`\equal` 31
`\errmessage` 7681, 7695, 7702, 7716
`\escapechar` 53, 1055
`\everyeof` 2164
`\excursion` 106, 8198
`\excursiongroup` 106, 8215
`\excursionref` 106, 8199, 8212
`exnote (env.)` 1, 8649
`\exnote` 8353, 8697, 8816, 8902

exp commands:
`\exp_after:wN` 356, 445,
 448, 480, 547, 698, 887, 2166, 2210,
 2211, 2212, 2352, 2397, 2398, 2399,
 2406, 2410, 2414, 2415, 2937, 2958,
 3574, 3806, 4105, 4118, 4706, 4734,
 4855, 5048, 5285, 5287, 5295, 5297,
 5343, 5350, 5351, 5352, 5371, 5404,
 5567, 5569, 5671, 5672, 5673, 5686,
 5881, 5944, 5990, 6006, 6222, 6272,
 6275, 6599, 6895, 6908, 6972, 7007
`\exp_args:N` ... 535, 540, 554, 555,
 886, 940, 1028, 1101, 1128, 1652,
 1801, 1814, 1828, 1844, 1847, 1850,
 1852, 2346, 2840, 2846, 2936, 2939,
 3368, 3396, 3402, 3768, 3777, 3934,
 3998, 4000, 4677, 4822, 5019, 5454,
 5600, 5602, 5611, 6183, 6233, 6320,
 6324, 6402, 6483, 6532, 6538, 6772,
 6808, 6901, 7186, 7393, 7725, 9163
`\exp_args:NNe`
 ... 54, 198, 201, 595, 1073, 1078,
 1084, 2460, 2899, 3548, 4037, 4223,
 4321, 5486, 5489, 5573, 5620, 5705,
 6503, 6611, 6666, 6734, 8232, 8443
`\exp_args:Nne` 207, 547, 2028, 2099,
 2484, 2524, 2581, 2782, 2860, 4179,
 4330, 4369, 4477, 4496, 4785, 4980,
 6121, 6558, 6599, 6667, 6872, 7087,
 7237, 7415, 7614, 8382, 8471, 9162
`\exp_args:NNNo`
 832, 1176, 3558, 4558, 4560, 4794, 4989
`\exp_args:NNno` 267, 684, 832
`\exp_args:Nnno` 1728, 1736
`\exp_args:NNNx` 1176, 1717
`\exp_args:NNnx`
 ... 837, 3075, 4794, 4989, 5465, 5471
`\exp_args:NNo` 38,
 48, 66, 69, 96, 100, 165, 173, 696,
 805, 837, 841, 846, 851, 1016, 1179,
 1659, 1990, 2000, 3208, 3234, 3555,
 3561, 3565, 4271, 4284, 4616, 5699
`\exp_args:Nno` 252, 254, 267,
 379, 2427, 2962, 2965, 3801, 4037,
 4850, 5043, 5149, 5154, 5238, 6029,
 6492, 7035, 8369, 8522, 9095, 9135
`\exp_args:NNx`
 ... 55, 110, 122, 911, 998, 1899,
 1947, 2128, 2147, 6625, 8306, 8309,
 8312, 8315, 9065, 9068, 9071, 9074
`\exp_args:Nnx` ... 247, 1317, 1593,
 1661, 1899, 1947, 2815, 3016, 3021,
 3208, 3234, 3758, 3925, 5127, 5328,
 5360, 5370, 5880, 6153, 6829, 7652

\exp_args:N	142, 162, 163, 164, 211, 302, 440, 607, 613, 751, 1026, 1112, 1212, 1264, 1272, 1294, 1317, 1532, 1717, 1722, 1727, 1759, 1764, 1766, 2074, 2367, 2464, 2519, 2556, 2885, 2897, 2934, 3121, 3136, 3378, 3379, 3572, 3577, 3661, 3763, 3764, 3765, 4035, 4191, 4549, 4724, 4800, 4801, 4802, 4813, 4814, 4815, 4995, 4996, 4997, 5008, 5009, 5010, 5102, 5171, 5174, 5178, 5219, 5358, 5363, 5368, 5373, 5375, 5463, 5491, 5600, 5602, 5629, 5714, 6131, 6141, 6173, 6246, 6358, 6362, 6369, 6396, 6431, 6433, 6505, 6507, 6513, 6588, 6614, 6620, 6622, 6627, 6631, 6637, 6684, 6733, 6976, 7026, 7815, 7874, 7875, 7878, 8305, 8389, 8478, 9064	
\exp_args:Nx	.. 249, 697, 752, 760, 1536, 1642, 1644, 2216, 2521, 3000, 7943, 8839	
\exp_not:N	56, 211, 213, 561, 946, 953, 1111, 1222, 1225, 1952, 2009, 2164, 2210, 2420, 2816, 3371, 3405, 4190, 4566, 5130, 5133, 5169, 5170, 5173, 5177, 5181, 5338, 5413, 5419, 5621, 5625, 5628, 5706, 5710, 5712, 5882, 6240, 6432, 6506, 6512, 6548, 6587, 6613, 6619, 6628, 6639, 7826, 7831	
\exp_not:n	.. 211, 544, 1112, 1317, 1532, 2414, 2441, 2464, 2594, 2618, 3377, 3378, 3379, 3763, 3764, 3765, 3954, 4191, 4232, 4233, 4549, 4706, 4734, 4800, 4801, 4802, 4813, 4814, 4815, 4995, 4996, 4997, 5008, 5009, 5010, 5124, 5171, 5174, 5178, 5180, 5182, 5219, 5351, 5358, 5363, 5368, 5373, 5375, 5587, 5629, 5666, 5671, 5714, 6216, 6358, 6396, 6426, 6431, 6433, 6505, 6507, 6509, 6513, 6516, 6542, 6588, 6590, 6614, 6616, 6620, 6622, 6627, 6631, 6637	
\expandafter	.. 297, 361, 2082, 2083, 2084, 2085, 2086, 2396, 2659, 6062, 7918, 7919	
\ExplSyntaxOff	2133, 2573, 7732, 8319, 9078	
\ExplSyntaxOn	2132, 2569, 7729, 8302, 9061	
\extref	.. 82, 1556	
\extstructure (env.)	.. 93, 6178	
\extstructure*	(env.) .. 93	
	F	
\fbox	.. 9019, 9029	
	\fi	
	299, 314, 1894, 1909, 1917, 1973, 2212, 5191, 5225, 5896, 6000, 6062, 6964, 7537, 7920, 7988, 7991, 8090, 8240, 8456, 8563, 8577, 8884, 8946, 9034, 9247, 9257, 9258	
	\fiboxed	
	102	
	file commands:	
	\file_if_exist:nTF .. 635, 652, 1165	
	\filepath	
	\fillinsol	
	\first	
	\fn	
	\foo	
	15, 54, 85, 138, 140	
	\fooname	
	\footnote	
	8884, 8946, 9021	
	\footnotesize	
	9164	
	\foral	
	46, 51	
	\forallall	
	46, 7524, 7530, 7541	
	frame (env.)	
	1, 7942	
	\frameimage	
	105, 8125	
	\frameimages	
	102	
	\frametitle	
	8042	
	\frontmatter	
	1451, 1452, 1453	
	\fun	
	45	
	\funspace	
	37	
	G	
	\gdef	
	8198	
	\given	
	71, 113	
	\global .. 1244, 1248, 1265, 1270, 2083, 7570, 7571, 8592, 8595, 9102, 9104	
	\gnote (env.)	
	1, 8698	
	\gnote	
	8354, 8750, 8817, 8903	
	\gnotes	
	70, 107, 112	
	group commands:	
	\group_begin: .. 43, 52, 636, 1054, 1695, 2175, 2434, 2471, 2790, 2810, 2834, 3065, 3289, 3330, 3678, 3730, 3892, 3894, 3896, 3898, 3958, 4164, 4471, 4517, 4779, 4934, 4974, 5103, 5107, 5205, 5358, 5514, 5522, 5596, 5617, 5665, 5681, 5884, 5921, 5929, 5940, 5962, 5973, 5986, 6523, 6540, 6896, 6965, 7014, 7067, 7281, 7348, 7370, 7433, 7454, 7482, 8519, 9132	
	\group_end: .. 47, 55, 642, 1058, 1742, 2198, 2452, 2485, 2799, 2815, 2877, 3070, 3295, 3336, 3680, 3738, 3892, 3894, 3896, 3898, 3963, 3983, 4167, 4482, 4520, 4790, 4946, 4985, 5074, 5092, 5104, 5108, 5149, 5282, 5293, 5310, 5321, 5343, 5371, 5376, 5392, 5425, 5517, 5534, 5596, 5620, 5673, 5705, 5892, 6017, 6440,	

6496, 6526, 6549, 6559, 6625, 6905,
 6982, 7028, 7080, 7292, 7358, 7362,
 7411, 7448, 7473, 7572, 8523, 9136
`\group_insert_after:N` 257, 273

H

`\halign` 9022
`\have` 7373
`\hbox` 1256, 3061, 3364, 3792, 3946,
 3975, 4346, 4822, 5034, 6194, 6778,
 7202, 7226, 8186, 8189, 8860, 9004
`\hbox` commands:
 `\hbox_set:Nn` 3677, 4207
 `\hbox_unpack:N` 3974, 3981, 8799, 8804
`\HCode` 309
`\hfil` 7208, 9022
`\hfill` 7208
`hint (env.)` 1, 8597
`\hint` 8352, 8648, 8815, 8901
`hints` 70, 107, 112
`\hline` 9162, 9164, 9165, 9166, 9167
`\href` 1839
`\hrule` 7202, 8860
`\hspace` 9032
`\HTML` 15, 26
`\huge` 9030
`\hwexam` commands:
 `_hwexam_includeassignment_-`
 `keys_tl` 9093, 9094, 9096, 9133
 `\hwexam_kw_*` 9060
 `\c_hwexam_multiple_bool` 9053
`\hwexamheader` 9169, 9208
`\hwexamminutes` 9171
`\hyperlink` 1833, 1834
`\hypertarget` 1813, 1814

I

`\id` 120
`\if` 2210, 5879
`\IfBooleanTF` 3722, 4158,
 4468, 5442, 7319, 7426, 7449, 8126
`\ifcsname` 296, 8240
`\ifdefempty` 8231
`\iffalse` 7523
`\IfFileExists` 6, 20, 1722, 1908,
 1916, 1989, 1999, 3237, 3242, 3262
`\IfInputref` 28, 80, 1966, 8229
`\ifinputref` 28, 80, 1884, 1973
`\ifintest` 8293
`\ifmmode` 6062
`\ifnotes` 103, 7806, 7938
`\ifSGvar` 106, 8248
`\ifsolutions`
 108, 8285, 8557, 8570, 8884, 8946, 9018

`\ifstexhtml` 28, 78, 137, 296, 8454
`\ifvmode` 1909, 1917, 5191, 5225, 6000, 6964
`\ifx` 2082, 7917, 7986, 7989, 9240, 9241, 9249
`\ignorespaces` 4790, 4985, 8092
`\image` 114
`\imly` 51
`\importmodule` 38, 49, 55, 90–92,
 127, 130, 132, 133, 140, 209, 2988, 3267
`\includeassignment` 9131
`\includegraphics` 80, 2045, 9233
`\includeproblem` 70, 111, 8511
`\indent` 5191, 5225, 6000, 6964
`\infprec` 87, 88, 4271, 5851, 5854, 5887, 7494
`\inline*` 97
`\inlineass` 50, 55, 58, 97
`\inlinedef` 50, 97
`\inlineex` 97
`\input` 27, 77, 132, 40, 82, 85, 318,
 1031, 1921, 2012, 8303, 8307, 8310,
 8313, 8316, 9062, 9066, 9069, 9072,
 9075, 9169, 9242, 9245, 9251, 9255
`\inputassignment` 71, 113
`\inputref` 27, 28, 79, 80, 99, 1884,
 8144, 8145, 8207, 8232, 8522, 9135
`\inputref*` 104, 8144
`\inputreffalse` 1884, 1893
`\inputreftrue` 1891, 1928
`\insertframenumber` 8192, 8194
`\insertshortauthor` 8185
`\insertshortdate` 8196
`\insertshorttitle` 8188
`\inset` 45
 int commands:
 `\int_case:nn` 1369
 `\int_case:nnTF` 1327, 1396, 1429
 `\int_compare:nNnTF`
 249, 2026, 2098, 3549, 4196, 4231,
 4730, 5312, 5599, 5610, 5659, 5865,
 6355, 6365, 6457, 6840, 7148, 9149
`\int_compare_p:nNn`
 7118, 7130, 7142, 7159, 7171
`\int_decr:N` 7149, 7177
`\int_eval:n`
 266, 268, 270, 272, 274, 5468, 5474,
 5864, 6154, 6161, 6288, 8484, 8487,
 9102, 9145, 9146, 9163, 9196, 9203
`\int_gincr:N` 1510, 5453,
 8481, 8539, 8603, 8652, 8703, 9143
`\int_gset:Nn` 5417
`\int_gzero:N` 5402, 8404
`\int_incr:N` 1319, 1329, 1333,
 1357, 1371, 1374, 1377, 1398, 1402,
 1433, 1439, 3868, 3869, 3870, 3871,

4646, 4729, 5565, 7123, 7135, 7146,
 7163, 7175, 7591, 7865, 8764, 9150
`\int_new:N` 1277, 1506, 3824, 4012, 4716, 5395,
 5549, 5853, 8449, 8526, 8698, 9141
`\int_set:Nn` 1413, 1414, 1415, 1416,
 1417, 1418, 1420, 3045, 3833, 3837,
 3841, 3845, 3849, 3853, 3857, 3861,
 3865, 3940, 4019, 4060, 4092, 4619,
 4724, 4731, 4769, 4893, 5525, 5854,
 5887, 7116, 7128, 7140, 7157, 7169
`\int_step_function:nN` ... 5131, 5331
`\int_step_inline:nn` 3882, 4263, 4270, 4290, 4540
`\int_use:N` .. 228, 1303, 1383, 1423,
 1443, 1511, 3375, 3761, 4340, 4457,
 4619, 4798, 4811, 4993, 5006, 5241,
 5417, 5454, 5586, 5623, 5708, 5851,
 5852, 6831, 7594, 7655, 7858, 8497,
 8541, 8605, 8654, 8705, 8766, 9162
`\int_zero:N` 3827, 3828, 4601, 4727, 5555, 7584
`\c_max_int` 5851, 5852
`\l_tmpa_int` 4601, 4604, 4619, 4646, 4727, 4729,
 4730, 4731, 4735, 7116, 7119, 7122,
 7123, 7128, 7131, 7134, 7135, 7140,
 7143, 7146, 7148, 7149, 7151, 7152,
 7157, 7160, 7163, 7165, 7169, 7172,
 7175, 7177, 7178, 7584, 7591, 7594
intarray commands:
`\intarray_gset:Nnn` 7151, 7165, 7178, 7253
`\intarray_gzero:N` 7252
`\intarray_item:Nn` ... 7119, 7122,
 7131, 7134, 7143, 7152, 7160, 7172
`\intarray_new:Nn` 7114
`\interpretmod` 3627, 3629, 3631
`\interpretmodule` 3514, 3516
`\intestfalse` 8297
`\intesttrue` 8295
invokation commands:
`\invokation_macro` 120, 122, 128
ior commands:
`\ior_close:N` 659, 665, 1059, 1190
`\ior_map_inline:Nn` 1175
`\ior_new:N` 1171
`\ior_open:Nn` 653, 661, 1173
`\ior_open:NnTF` 1053
`\ior_str_get:NN` 1056
`\ior_str_map_inline:Nn` 655, 662
`\g_tmpa_ior` 653, 655,
 659, 661, 662, 665, 1053, 1056, 1059
iow commands:
`\iow_close:N` 648, 658, 1498
`\iow_new:N` 617, 1496
`\iow_now:Nn` 625, 656, 663, 1524, 1795, 7728, 8443
`\iow_open:Nn` 647, 654, 1497
`\g_tmpa_iow` 654, 656, 658
`\isassociative` 49
`\iscommutative` 49
`\item` 7219, 8497, 8773, 8824, 8910
`\itshape` 7212

J

`\jobname` 74, 138, 647, 652,
 653, 654, 661, 666, 675, 981, 1497, 1717
`\join` 63

K

keys commands:
`\l_keys_choice_tl` 3695
`\keys_define:nn` 27,
 379, 7743, 7785, 8215, 8264, 9051, 9221
`\l_keys_key_str` 4141, 4144, 6106, 6109
`\l_keys_key_tl` 4142, 6107
`\keys_set:nn` 383, 8224
keyval commands:
`\keyval_parse>NNn` 6377

L

`\label` 81, 118, 129, 1536, 8035
`\labelsep` 7994
`\labelwidth` 7995
`\lambda` 7524, 7535, 7536
`\langle` 7511
`\Large` 7803, 8154, 9113
`\LaTeX` 23
`\latext` 23
`\ldots` 7516, 7561, 7563, 7565
`\leaders` 8156
`\leavevmode` 8050
`\left` 5881
`\leftmargin` 7996
`\let` . 356, 1452, 1453, 1462, 1463, 1725,
 2083, 2165, 2176, 2183, 2194, 2206,
 3313, 3975, 3982, 4476, 4495, 4784,
 4979, 5208, 5926, 5936, 5950, 6012,
 6091, 7570, 7571, 7730, 7731, 8163
`\libinput` 20, 77, 1976
`\libusepackage` 77, 78, 2024, 7934
`\libusetheme` 7933
`\libusetikzlibrary` 77, 115, 2080
`\linewidth` 8580, 8587,
 8641, 8646, 8690, 8695, 8743, 8748
`\LoadClass` 15, 7771, 7773

\long	138, 588, 8167, 8176
\lstinputlisting	80, 2059
\lstinputmhlisting	80, 2035
M	
\macro	132–134, 139
\macroname	32
\magma	53
\maincomp	32, 33, 36, 54, 87, 94, 100, 3982, 4190, 4199, 4210, 4211, 4232, 4363, 4566, 5210, 5391, 5908, 6087, 6532, 6533, 6538, 6546, 6551
\mainmatter	1474, 1475, 1485, 1486
\makeatletter	2130, 8302, 9061, 9208
\makeatother	2131, 8319, 9078, 9208
\maketitle	1269, 1270
\mapsto	7533, 7534
\marginnote	7984
\marginpar	8419, 8424, 8501, 8506, 8798, 8803
\mathbb	7547
\mathbin	33, 7492, 7496, 7503, 7529, 7540
\mathclose	33, 5893, 7494, 7501, 7505, 7511, 7513, 7521, 7528, 7540, 7544
\mathhub	75, 135, 167, 30, 1042
\mathop	33, 7531, 7542
\mathopen	33, 5888, 7494, 7501, 7505, 7511, 7513, 7520, 7528, 7540, 7544
\mathord	33
\mathpunct	33, 4544, 5544, 7520, 7530, 7536
\mathrel	32, 33, 7497, 7534
\mathrm	4566, 7520
mathstructure (env.)	93, 6092
\mathstructure	6136, 6137
\mathtt	7500, 7501, 7550, 7553, 7557
mcb (env.)	1, 8806
\mcb	8349, 8812, 8837, 8898
\mcc	67, 109, 8818, 8858
\meaning	1024, 2396, 2659, 4236, 4510, 4595, 4621, 5148, 5153, 5356, 5366, 5367
\medskip	8011, 8025, 8044
\meet	63
\message	26, 7764, 7767, 7924
\mhframeimage	105
\mhgraphics	80, 2035, 8138, 8140
\mhinput	80, 1884
\mhtikzinput	115, 2035
\min	71, 113
\min	8801
min	70, 107, 112
\mmlarg	337
\mmlintent	337
\mmtdef	7631, 7641
\MMTinclude	7575
mmtinterface (env.)	
\MMTrule	7580
\mname	84, 93
mode commands:	
\mode_if_math:TF	3364, 3975, 3976, 3983, 5250, 9003
\mode_if_vertical:TF	3974, 3981
\MSC	7574
msg commands:	
\msg_error	139
\msg_error:nn	82, 1784, 7022, 7075, 8365
\msg_error:nnn	85, 154, 193, 241, 812, 948, 955, 1978, 1981, 2107, 3200, 3256, 3427, 3457, 3988, 4679, 4921, 6166, 6643, 6670
\msg_error:nnnn	352, 830, 835, 854, 889, 961, 1952, 2009, 2919, 3007, 3352, 3873, 4449, 4944, 5199, 5233, 5479, 5506, 6015, 6280, 6980
\msg_none:nn	79
\msg_redirect_module:nnn	101
\msg_redirect_name:nnn	105
\msg_set:nnn	76
\msg_warning:nn	1072
\msg_warning:nnn	1694, 1736
\msg_warning:nnnn	434, 1728
\mult	40, 41, 88, 89
\multicolumn	9163
\multiple	71, 112
N	
nassertion (env.)	1
\Nat	36
ndefinition (env.)	1
\neginfpref	41, 87, 88, 4241, 4244, 4253, 4256, 4269, 5070, 5306, 5317, 5851
\newcommand	445, 480, 1578, 1959, 1967, 1972, 2017, 2024, 2054, 2060, 2068, 2078, 2105, 7477, 7677, 7698, 7915, 7984, 8046, 8147, 8151, 8199, 8206, 8209, 8226, 8340, 8761, 8863, 8925, 8953, 8958, 8963, 8968, 9001, 9038, 9041, 9042, 9043, 9044, 9045, 9148, 9233, 9238, 9262
\newcounter	7774, 7775, 7776, 7777, 7899, 7911, 8339, 8357, 8358, 9091
\NewDocumentCommand	448, 1542, 1774, 1779, 1826, 1898, 1946, 2489, 2568, 4933, 5441, 5920, 5928, 5939, 5961, 5972, 5985, 6895, 6985, 6990, 6996, 7004, 7013, 7056, 7066, 7084, 7410, 7446, 7461, 7580, 7641, 7933, 8125, 8518, 9131

```

\NewDocumentEnvironment ..... 483, 1349, 1358, 7466, 7604
\newenvironment ..... 1471, 1482, 7180, 7387, 8098, 8100, 9191
\newif 297, 320, 1884, 7806, 8285, 8293, 9198
\newlabel ..... 7730, 7731
\newlength ..... 7936, 7937, 7940
\newline ..... 8580, 8587, 8641, 8646, 8690, 8695, 8743, 8748
\newpage ..... 8032, 9045, 9210
\newsavebox ..... 8068
nexample (env.) ..... 1
\ninputref ..... 8145, 8147, 8151
\nobreak ..... 7208
\noindent ..... 1304, 1909, 1917, 6879, 7198, 8011, 8025, 8048, 8066, 8416, 8581, 8642, 8691, 8744
\nointerlineskip ..... 8158
\notation .. 32, 33, 41, 85, 87, 92, 123, 126, 132, 2987, 4148, 7492, 7495, 7496, 7497, 7510, 7512, 7530, 7531, 7535, 7539, 7541, 7542, 7547, 7550
note (env.) ..... 1
notes ..... 70, 102, 107, 112
\notesfalse ..... 7818
notesslides commands:
  \c_notesslides_notes_bool ..... 7745, 7746, 7759, 7762, 7770, 7786, 7787, 7799, 7807, 7930, 8091, 8097, 8131, 8146, 8165, 8200, 8210
  \c_notesslides_sectocframes_bool ..... 7788, 7886
  \c_notesslides_topsect_str 7789, 7812, 7815, 7871, 7874, 7875, 7878
notesslides internal commands:
  \c__notesslides_class_str ..... 7741, 7744, 7750, 7771
  \__notesslides_define_chapter: ... 7876, 7879, 7891
  \__notesslides_define_part: ... 7880, 7903
  \__notesslides_do_label:n 7821, 7857
  \__notesslides_do_sectocframes: ... 7820, 7887
  \__notesslides_do_yes_param:Nn .. 7942, 7961, 7964, 7967, 7970, 7973, 7976
  \c__notesslides_docopt_str ... 7747
  \c__notesslides_document_str ... 7914, 7917
  \__notesslides_eat: . 8101, 8105, 8108
  \__notesslides_excursion_args:n ... 8220, 8227
\l__notesslides_excursion_id_str ..... 8216, 8222
\l__notesslides_excursion_intro_- tl ..... 8217, 8221, 8231, 8233
\l__notesslides_excursion_- mhrepos_str ..... 8218, 8223, 8232
\l__notesslides_frame_allowdisplaybreaks_- bool ..... 7953, 7964
\l__notesslides_frame_allowframebreaks_- bool ..... 7952, 7961
\__notesslides_frame_box_begin: . 8004, 8015, 8038
\__notesslides_frame_box_end: ... 8009, 8024, 8040
\l__notesslides_frame_fragile_- bool ..... 7954, 7967
\l__notesslides_frame_label_str . 7951, 7959, 8034, 8035
\l__notesslides_frame_shrink_- bool ..... 7955, 7970
\l__notesslides_frame_squeeze_- bool ..... 7956, 7973
\l__notesslides_frame_t_bool ... 7957, 7976
\__notesslides_inputref: ... 8144, 8145, 8148
\__notesslides_notes_env:nmmn ... 8096, 8116, 8117, 8118, 8119, 8120, 8121
\l__notesslides_num ..... 7824, 7826, 7829, 7831, 7834, 7835, 7838, 7839, 7842, 7843, 7846, 7847, 7850, 7851, 7858
\__notesslides_setup_itemize: ... 7980, 8037
\l__notesslides_tmp ..... 8249, 8254
\g__notesslides_variables_prop .. 8241, 8243, 8246, 8249
notesslidesfont ..... 8007, 8022, 8163
notesslidesfooter ..... 8011, 8025, 8161
notesslidestitleemph ..... 8044, 8153
notesttrue ..... 7808
nparagraph (env.) ..... 1, 1
nsproof (env.) ..... 1
\null ..... 7208
\num ..... 140
\number ..... 71, 113
\numberline ..... 7856
\numberproblemsin ..... 8339

```

O

```

\objective ..... 7698
\OMDoc ..... 26
\omdoc ..... 15

```

\oplus	7502, 7503	1
P		
\PackageError	8250	
\pagebreak	8431, 9129	
\pagenumbering	1458, 1468, 1479, 1490	
\par	139, 360, 1302, 1306, 7181, 7208, 7251, 7308, 8011, 8025, 8048, 8053, 8061, 8066, 8073, 8083, 8416, 8427, 8430, 8455, 8580, 8587, 8641, 8646, 8690, 8695, 8743, 8748, 8832, 8918, 9112, 9117, 9127, 9129	
\paragraph	27, 79	
\parsep	7217, 8495, 8771, 8822, 8908	
\part	27, 79, 7907, 7908	
\partname	7826, 7904, 7905	
\parttitlename	7826	
\PassOptionsToClass	7749, 7750	
\PassOptionsToPackage	10, 7751, 7752, 7763, 7766, 7791, 7792, 7809, 8282, 9055	
\pause	8046	
\PDF	15, 26	
\pdfbookmark	7858	
\pdfdest	1538	
peek commands:		
\peekCharCode:NNTF	3582, 5064, 5077, 5140, 5142, 5264, 5271, 6302, 6308, 6449, 6460	
\peekCharCode_remove:NNTF	5063, 5139, 5255, 5260, 5262, 5269, 5342, 5730, 6307, 6557	
\phantom	9030	
\Pi	7524	
\plus	39–41, 44, 87–89	
\precondition	7677	
\prematurestop	105, 7914	
\premise	52, 98, 6947, 7084, 7091	
prg commands:		
\prg_new_conditional:Nnn	232, 282, 534, 539, 748, 758, 2137, 2547, 2551, 3655, 3666, 3670, 5485, 5598, 5609	
\prg_new_protected_conditional:Nnn	768	
\prg_return_false:	234, 284, 537, 542, 749, 753, 759, 761, 784, 788, 2138, 2549, 2553, 3656, 3671, 5503, 5507, 5604, 5605, 5606, 5612, 5613	
\prg_return_true:	234, 284, 537, 542, 751, 753, 761, 788, 2138, 2549, 2553, 3667, 5501, 5604, 5612	
\printexcursion	106	
\printexcursions	8198, 8207, 8228, 8236	
problem (env.)	8526, 8539, 8541, 8603, 8605, 8652, 8654, 8703, 8705	
problem commands:		
\g_problem_id_counter	8367, 8368, 8370, 8520	
\l_problem_inputproblem_keys_tl	8809, 8824, 8834, 8858	
\problem_mcc_box_tl	8890, 8896, 8910, 8920	
\problem_scc_box_tl	8890, 8896, 8910, 8920	
problem@kw@points commands:		
\problem@kw@points:	8791	
problem@kw@schema commands:		
\problem@kw@schema:	8788	
\problemheader	8416, 8434	
problems internal commands:		
__problems_activate_macros:	8347, 8408, 8457	
\l__problems_anscls_int	8698, 8764, 8766	
\c__problems_boxed_bool	8278	
__problems_do_yes_param:Nn	8838	
__problems_exnote_start:n	8649, 8667, 8670	
\l__problems_fillin_solution_tl	8974, 8995, 9015, 9020, 9024	
__problems_fillinsol:nn	9004, 9006, 9010	
__problems_gnote_start:n	8700, 8719, 8722	
\c__problems_gnotes_bool	8268, 8721, 8733	
\l__problems_has_min_bool	8361, 8402, 8403, 8486, 8505	
\l__problems_has_pts_bool	8360, 8399, 8400, 8483, 8500	
__problems_hint_start:n	8600, 8618, 8621	
\c__problems_hints_bool	8270, 8620, 8631	
\l__problems_in_problem_bool	8359, 8362, 8364, 8395, 8453, 8458	
__problems_mccline:n	8819, 8834, 8876	
\c__problems_min_bool	8276, 8422, 8504, 8802	
\l__problems_min_tl	8397, 8401, 8411, 8423, 8424, 8444, 8460, 8487	
\c__problems_notes_bool	8266, 8669, 8680	
__problems_parse_fillin_- arg:nnnn	8978, 8979, 8980, 8983	
\l__problems_path_seq	8379, 8380, 8463, 8464, 8468, 8469	

```

\l__problems_prob_imports_tl . 8336
\l__problems_prob_refnum_int . 8337
\c__problems_pts_bool .....
..... 8274, 8417, 8499, 8797
\l__problems_pts_tl .. 8396, 8398,
8410, 8418, 8419, 8444, 8459, 8484
\__problems_record_problem: .....
..... 8412, 8442
\__problems_sccline:n 8905, 8920, 8938
\__problems_solution_start:n ...
..... 8536, 8555, 8558
\c__problems_solutions_bool .....
..... 8272, 8288
\g__problems_subproblem_int ...
..... 8404, 8449, 8481, 8497
\c__problems_test_bool .....
.. 8280, 8294, 8431, 9039, 9041, 9045
\ProcessKeysOptions .....
.... 39, 7755, 7795, 8287, 9058, 9227
\ProcessOptions ..... 11
\prod ..... 7531, 7542
\prop ..... 42
prop commands:
\prop_clear:N . 1174, 2881, 2882, 6648
\prop_const_from_keyval:Nn . 110, 122
\prop_gclear:N .....
..... 2320, 2321, 2322, 2432, 5400
\prop_get:NnN ..... 1208
\prop_get:NnNTF .....
..... 189, 903, 904, 1800, 1904,
1905, 1980, 2935, 2956, 5486, 8249
\prop_gput:Nnn .....
..... 2460, 2581, 2593, 2595, 2628, 5429,
5465, 5471, 5489, 7052, 8243, 9144
\prop_gset_eq:NN .... 1191, 1222, 1225
\prop_gset_from_keyval:Nn .....
..... 1194, 1219, 2425, 2427, 5413
\prop_if_exist:NTF .....
..... 901, 1100, 1117, 1603,
1697, 1799, 1864, 1977, 3080, 5412
\prop_if_in:NnTF .....
..... 138, 173, 805, 6608, 6665, 6695, 6709
\prop_item:Nn .....
..... 139, 921, 1101, 1214, 1621,
1700, 1866, 1869, 1878, 2042, 2056,
2071, 2861, 3017, 3022, 3083, 3108,
3119, 3134, 6600, 6640, 6668, 8246
\prop_map_break: ..... 2621
\prop_map_break:n 2640, 2676, 2679,
2757, 3041, 4057, 4909, 4912, 7039
\prop_map_function:NN ... 2418, 2948
\prop_map_inline:Nn .. 2428, 2605,
2638, 2682, 2700, 2703, 2725, 2728,
..... 2766, 2912, 2931, 2971, 2999, 3003,
4073, 4902, 6262, 6488, 7036, 9156
\prop_new:N ..... 8241, 9138
\prop_put:Nnn .. 1182, 1183, 1184,
1185, 1186, 2899, 2903, 3371, 3405,
4795, 4990, 6674, 6697, 6711, 9151
\prop_remove:Nn ..... 3467
\prop_set_eq:NN 1089, 1206, 1210, 1612
\prop_to_keyval:N .....
... 1192, 1195, 1220, 2397, 2398,
2399, 2406, 2410, 2926, 2929, 5414
\protect ..... 7856, 8252
\protected ..... 120,
138, 358, 572, 573, 1270, 5937, 5951
\providecommand .. 2040, 2047, 7922, 9232
\ProvidesExplClass ..... 5, 7738
\ProvidesExplPackage .....
..... 19, 7782, 8261, 9048, 9218
\pts ..... 8796
pts ..... 70, 107, 112

Q
\quad ..... 7835, 7839, 7843,
7847, 7851, 9002, 9008, 9120, 9123
quark commands:
\quark_new:N ..... 2161, 2444
quark internal commands:
\q_stex_smsmode_break .....
..... 2161, 2164, 2224

R
\range ..... 7511
\realization ..... 3538, 3540
\realize ..... 63, 64, 3648, 3650, 3652
\ref ..... 81, 1644
\refstepcounter ..... 8375, 9104
\relax ..... 359, 640, 641, 1453, 1463,
1538, 1695, 2082, 2168, 2206, 2210,
2435, 2436, 3975, 3982, 8132, 8134
\renamedecl .... 64, 126, 2993, 3391, 3475
\renewcommand ..... 7931, 8032, 8042
\renewenvironment ..... 7985, 8029, 8047, 8069, 8092, 8094
repo commands:
\repo_prop ..... 135, 136
\reqpts ..... 71, 113
\requiremodule ..... 90, 91, 2989, 3340
\RequirePackage .....
... 4, 13, 18, 23, 24, 147, 149, 198,
201, 5877, 7739, 7757, 7779, 7783,
7797, 7810, 7811, 8123, 8262, 8300,
9049, 9059, 9219, 9230, 9235, 9236
\resizebox ..... 9161, 9244, 9250, 9254
\rhd ..... 7987, 7990

```

```

\right ..... 5882
\Rrightarrow ..... 7443
\rightmargin . 7218, 8496, 8772, 8823, 8909
\rule ..... 8580, 8587,
          8641, 8646, 8690, 8695, 8743, 8748
rustex commands:
  \rustex_direct_HTML:n .....
    ..... 8054, 8062, 8074, 8084
  \rustex_if:TF ..... 8052, 8053,
          8060, 8061, 8072, 8073, 8082, 8083
\rustexBREAK ..... 2134

S
sassertion (env.) ..... 96
scb (env.) ..... 8889
\scb ..... 8350, 8813, 8899, 8923
\scct ..... 8904, 8924
\scriptsize ..... 7984
\scriptstyle ..... 7990
sdefinition (env.) ..... 96
\second ..... 134
\section ..... 26, 27, 79
\sectiontitleemph ..... 7802, 7861
sectocframes ..... 102
\selectlanguage ..... 137, 142
seq commands:
  \seq_clear:N ..... 183, 683, 721,
          1074, 1983, 2178, 2689, 2714, 2739,
          2844, 2883, 4194, 4539, 4700, 5554,
          5618, 5683, 5692, 6128, 6179, 6229,
          6387, 6649, 6650, 6770, 6806, 7581
  \seq_count:N .. 2026, 2098, 3549, 6840
  \seq_gclear:N ..... 2172, 2173, 5401
  \seq_gclear_new:N ..... 984, 985
  \seq_get:NN ..... 1015
  \seq_get_right:NN 159, 8380, 8464, 8469
  \seq_gpop:NN ..... 1010
  \seq_gpush:Nn ..... 998
  \seq_gput_left:Nn ..... 1547, 1552
  \seq_gput_right:Nn .....
    ..... 1550, 2128, 2147, 5430
  \seq_gset_eq:NN ..... 997, 1013, 1017, 2871, 2872, 2873, 7720
  \seq_gset_split:Nnn ..... 5419
  \seq_if_empty:NTF .. 170, 719, 737,
          749, 759, 788, 834, 840, 845, 850,
          1009, 1012, 1118, 1951, 2008, 4034,
          6855, 7188, 7239, 7395, 7589, 7719
  \seq_if_empty_p:N .... 775, 776, 1987
  \seq_if_exist:NTF ..... 1592, 5418
  \seq_if_in:NnTF .. 1545, 1546, 1659,
          1661, 1990, 2000, 2272, 2281, 2658,
          2698, 2723, 2764, 6500, 6611, 6734
  \seq_item:Nn ..... 750,
          760, 2027, 2099, 4307, 4313, 5487, 6841
\seq_map_break: ..... 1594, 1598, 2676
\seq_map_break:n .... 1598, 2679, 4057
\seq_map_function:NN .. 723, 727, 6399
\seq_map_inline:Nn .....
  ..... 265, 918, 922, 1597,
          1632, 1954, 2011, 2187, 2681, 4071,
          4701, 4728, 5693, 6192, 6239, 6248,
          6776, 6914, 6941, 7259, 7314, 7590
\seq_new:N ..... 226, 1505, 1551, 2118, 2141, 2291
\seq_pop:NN ..... 720
\seq_pop_left:NN ..... 169, 703,
          781, 782, 833, 838, 844, 849, 919,
          923, 925, 1994, 3551, 3553, 3560, 3564
\seq_pop_left:NNTF .....
  ..... 1178, 4281, 4283, 4291
\seq_pop_right:NN ..... 161, 171,
          738, 793, 795, 800, 802, 804, 1146,
          2366, 3076, 4033, 4559, 4561, 6390
\seq_push:Nn ..... 726
\seq_put_left:Nn .....
  ..... 704, 724, 2699, 2724, 6696, 6710
\seq_put_right:Nn .....
  ..... 165, 191, 229, 741, 796,
          806, 809, 981, 1991, 1995, 2001,
          2325, 2355, 2368, 2660, 2765, 2847,
          2978, 4264, 4271, 4292, 4294, 4541,
          4703, 4705, 5585, 5621, 5657, 5664,
          5689, 5695, 5699, 5706, 6129, 6184,
          6188, 6234, 6666, 6773, 6809, 6823
\seq_reverse:N ..... 6391
\seq_set_eq:NN .. 729, 769, 770, 792,
          799, 917, 980, 1984, 4710, 4738, 5704
\seq_set_split:Nnn ..... 160,
          684, 794, 801, 832, 837, 1138, 1177,
          1985, 2365, 3075, 3548, 3558, 4032,
          4280, 4284, 4558, 4560, 6389, 7582
\seq_use:Nn .....
  ..... 196, 199, 202, 766, 796, 809,
          837, 1180, 2370, 3077, 3556, 3562,
          3566, 4038, 4350, 4544, 4687, 4711,
          5420, 5543, 6856, 7189, 7240, 7396
\l_tmpa_seq ..... 160,
          161, 165, 169, 170, 171, 183, 191,
          196, 199, 202, 4539, 4541, 4544,
          4558, 4559, 4560, 4561, 4700, 4703,
          4705, 4710, 4738, 6387, 6389, 6390,
          6391, 6399, 7581, 7582, 7589, 7590
\seqmap ..... 93, 5156, 5611
\setbox ..... 132, 8094, 8109, 8562, 8623, 8672, 8724
\setcounter ..... 9102, 9105

```

\setkeys 2044, 2058, 2073, 8136, 9239
 \setlength 7216, 7217, 7218,
 7936, 7937, 7941, 7994, 7995, 7996,
 8494, 8495, 8496, 8770, 8771, 8772,
 8821, 8822, 8823, 8907, 8908, 8909
 \setlicensing 104
 \setmetatheory 2470
 \setnotation 33, 88, 4424
 \setsectionlevel 27,
 79, 118, 1411, 7813, 7815, 7872, 7874
 \setSGvar 106, 8242, 8252
 \setsidelogo 104
 \setsource 104
 sexample (env.) 96
 \sf 8154
 \sffamily 8163
 sfragment (env.) 79
 sfragment commands:
 _sfragment_do_level:nn
 1300, 1312, 1328, 1332,
 1336, 1337, 1338, 1339, 1340, 7854
 _sfragment_end: 1308, 1322, 1354
 \skipfragment 79, 1395, 1399, 1403
 \slideframewidth . 7940, 7941, 8017, 8132
 \slideheight 7937
 slides 102
 \slidewidth
 .. 7936, 8020, 8050, 8132, 8134, 8138
 \smacro 89, 90
 \small 7212
 \smallskip ... 7208, 8419, 8424, 8501,
 8506, 8580, 8641, 8690, 8743, 9117
 smodule (env.) 83, 2515
 \smodule 2990
 \Sn 20, 86, 5908
 \sn 20, 24, 86, 5908
 \Sns 20, 86, 5908
 \sns 20, 86, 5908
 solution (env.) 1, 8526
 \solution 8348, 8590, 8814, 8900
 solutions 70, 107, 112
 \solutionsfalse 8291, 8595
 \solutionstrue 8289, 8592
 \source 134
 sparagraph (env.) 96
 \spfblock 7271, 7471
 \spfjust 7272, 7477, 7480
 spfsketch 7180
 \spfsketchenvautorefname 7198
 \spfstep 7266, 7373
 \spfstepautorefname 7307, 7388
 \spfstepenvautorefname 7388
 sproblem (env.) 8347
 \sproblemautorefname 8435
 sproof (env.) 98, 7214
 \sproofautorefname 7286
 \sproofend 7200, 7288, 7305
 \square 7201, 8859
 \sr 20, 24, 85, 5908
 \sref 81, 82, 96, 1556, 8202
 \sreflabel 81, 83, 118, 1505
 \srefsetin 81, 1578
 \srefsym 86, 1826
 \srefsymuri 86, 1857
 \startsolutions 108, 8591
 \stepcounter 1392, 7855, 8031
 \sTeX 14, 15, 78
 \stex 15, 24, 78
 stex commands:
 \l_stex_aB_args_seq
 4687, 4701, 4710, 4711, 4728, 4738,
 5543, 5554, 5585, 5618, 5621, 5657,
 5664, 5683, 5689, 5693, 5704, 5706
 \stex_activate_module:n
 .. 119, 2336, 2385,
 2657, 2657, 2669, 2839, 3182, 3280,
 3323, 3523, 6200, 6256, 6781, 7627
 \stex_add_definiens:nn
 .. 2995, 7026, 7034, 7261, 7316
 \stex_add_definiens_inner:nnnnnnn
 .. 7039, 7050
 \stex_add_module_notation:nnnnn .. 119
 \l_stex_all_modules_seq
 .. 119, 2178, 2291,
 2325, 2355, 2658, 2660, 2681, 4071
 \l_stex_allow_semantic_bool ..
 4940, 5082, 5160, 5161, 5164, 5190,
 5224, 5390, 5399, 5515, 5526, 5624,
 5709, 5767, 5778, 5811, 5840, 6001,
 6082, 6493, 6524, 6541, 6586, 6967
 \stex_annotate:nn
 .. 339, 342, 1283, 1292,
 1304, 1748, 1968, 1969, 3364, 3795,
 3798, 3805, 3810, 3812, 4353, 4359,
 4368, 4379, 4390, 4391, 4394, 4395,
 4399, 4844, 4847, 4854, 4858, 4861,
 5037, 5040, 5047, 5051, 5054, 5085,
 5531, 5538, 5632, 5741, 5754, 5771,
 5796, 5804, 5825, 5833, 6027, 6413,
 6421, 6426, 6634, 6880, 6901, 6978,
 7044, 7077, 7089, 7186, 7228, 7231,
 7273, 7327, 7336, 7340, 7393, 7462,
 7478, 7587, 7592, 8192, 8775, 8798,
 8803, 8878, 8881, 8940, 8943, 9013
 \stex_annotate:nnn 137, 138
 \stex_annotate_force_break:n ..
 .. 328, 329, 335,
 1305, 3363, 3792, 3810, 4841, 4859,

5035, 5052, 5539, 5637, 5746, 5759,
 5776, 5801, 5809, 5830, 5838, 6425,
 6880, 6902, 7045, 7078, 7588, 9013
`\stex_annotation_invisible:n`
 138, 330, 332,
 1362, 1384, 2532, 2789, 2833, 5187,
 5443, 7226, 7585, 7622, 8387, 8476
`\stex_annotation_invisible:nn`
 1256, 1423, 1431, 1437, 1443, 1909,
 1917, 3061, 3285, 3326, 3362, 3396,
 3437, 3447, 3777, 4346, 4822, 5019,
 5777, 5810, 5839, 6194, 6250, 6778,
 7060, 7576, 7623, 7691, 7712, 8987
`\stex_annotation_invisible:nnn` ... 138
`\l_stex_argnames_seq` 121
`_stex_args_end:` 5404,
 5428, 5431, 5953, 5957, 6007, 6973
`_stex_assign_do:n`
 2854, 3344, 3349, 3577
`\l_stex_assoc_args_count`
 121, 3824, 3828, 3870, 3871
`\l_stex_brackets_dones_bool`
 .. 4608, 5857, 5860, 5861, 5885, 5886
`_stex_capitalize:n`
 1294, 5944, 5947, 5990, 7007
`\stex_check_term:n`
 121, 122, 2177, 3356,
 3675, 3676, 3684, 3890, 4326, 7484
`\c_stex_check_terms_bool`
 33, 3662, 3665, 7487
`\stex_close_module:` ... 119, 2392,
 2392, 2541, 2796, 7568, 7635, 7637
`\l_stex_current_archive` 1086
`\l_stex_current_archive_prop` ...
 131, 136, 921, 931,
 936, 1089, 1100, 1101, 1110, 1199,
 1603, 1604, 1697, 1700, 1799, 1800,
 1864, 1866, 1869, 1904, 1905, 1977,
 1980, 2042, 2056, 2071, 3080, 3083
`\l_stex_current_args_tl`
 128, 4530, 4534, 5213, 5403, 5404, 5407
`\l_stex_current_arity_str`
 128, 4540, 5102, 5129,
 5131, 5212, 5312, 5330, 5331, 5362
`\l_stex_current_doc_uri` 136, 965,
 967, 968, 1515, 1520, 1522, 1526,
 1592, 1594, 1634, 1651, 1652, 6859
`\l_stex_current_domain_str`
 126, 2803, 2806, 2808,
 2816, 2821, 2830, 2839, 2870, 2885,
 2897, 2934, 2946, 3414, 3462, 3483,
 3503, 3523, 3527, 3600, 3620, 3641
`\g_stex_current_file`
 . 134–136, 159, 930, 932, 937, 985,
 988, 990, 992, 993, 994, 997, 998,
 1013, 1016, 1017, 1607, 1705, 2344,
 3090, 3096, 3097, 3145, 3164, 3208,
 3234, 7719, 7720, 8379, 8463, 8468
`\l_stex_current_language_str` 136,
 137, 135, 137, 172, 177, 2526, 3195,
 3198, 3222, 3225, 7616, 8384, 8473
`\l_stex_current_module` 120, 122
`\l_stex_current_module_str`
 . 119, 140, 2179, 2290, 2307, 2324,
 2325, 2340, 2353, 2354, 2355, 2356,
 2395, 2396, 2397, 2398, 2399, 2405,
 2407, 2411, 2416, 2418, 2525, 2535,
 2548, 2556, 2560, 2581, 2587, 2593,
 2595, 2605, 2612, 2617, 2626, 2628,
 2638, 2662, 2665, 2784, 2795, 2829,
 2840, 2937, 2940, 3369, 3403, 3430,
 3439, 3449, 3731, 3769, 3778, 3874,
 3935, 3944, 3953, 3959, 4487, 4506,
 4507, 6085, 6123, 6128, 6129, 6149,
 6241, 6263, 6824, 6847, 7035, 7036,
 7037, 7038, 7051, 7052, 7606, 7607,
 7610, 7615, 7634, 7636, 7649, 7663
`\l_stex_current_ns_uri` 136,
 971, 972, 2362, 2505, 3146, 3151,
 3152, 3165, 3170, 3171, 7488, 7569
`\l_stex_current_redo_tl`
 5194, 5207, 5218, 5221,
 5228, 5358, 6358, 6396, 6431, 6581
`\l_stex_current_return_tl`
 128, 3982, 5214, 5280, 5292, 5313, 5363
`\stex_current_section_level`
 1279, 1285, 1294, 1320, 7866
`\l_stex_current_symbol_str`
 128, 4178, 4208,
 4327, 4367, 4378, 4476, 4495, 4558,
 4784, 4936, 4938, 4944, 4979, 5069,
 5072, 5087, 5115, 5116, 5199, 5211,
 5233, 5249, 5278, 5279, 5282, 5288,
 5304, 5305, 5308, 5397, 5479, 5506,
 5523, 5625, 5626, 5669, 5710, 5711,
 5743, 5756, 5773, 5798, 5806, 5827,
 5835, 5888, 5893, 5966, 5979, 5992,
 6002, 6015, 6023, 6027, 6029, 6084,
 6439, 6440, 6968, 6976, 6978, 6980
`\l_stex_current_term_tl` 4941, 5162,
 5216, 5524, 5627, 5628, 5629, 5712,
 5740, 5753, 5768, 5779, 5795, 5812,
 5824, 5841, 6010, 6432, 6438, 6628
`\stex_current_this:` . 5208, 6081, 6091
`\l_stex_current_this_tl`
 6083, 6087, 6101, 6104
`\l_stex_current_type_tl`
 128, 5084, 5215, 6300,

```

6355, 6362, 6365, 6369, 6684, 6733
\stex_deactivate_macro:Nn .....
..... 139, 349,
349, 2577, 2984, 2985, 2986, 2987,
2988, 2989, 2990, 3271, 3340, 3474,
3475, 3476, 3493, 3514, 3538, 3606,
3627, 3648, 3742, 3967, 4185, 4525,
6136, 6210, 6272, 6988, 6994, 7002,
7010, 7031, 7064, 7082, 7091, 7366,
7439, 7459, 7464, 7471, 7480, 7578,
7602, 7675, 8590, 8648, 8697, 8750,
8795, 8812, 8813, 8814, 8815, 8816,
8817, 8837, 8888, 8898, 8899, 8900,
8901, 8902, 8903, 8923, 8950, 9037
\stex_debug:nn .....
65, 65, 102, 106, 177, 196, 247,
514, 639, 706, 713, 860, 930, 933,
968, 972, 983, 1024, 1032, 1060,
1065, 1085, 1088, 1121, 1164, 1192,
1203, 1213, 1233, 1249, 1264, 1521,
1586, 1591, 1631, 1635, 1651, 1654,
1660, 1662, 1666, 1669, 1675, 1677,
1680, 1689, 1690, 1716, 1763, 1765,
1767, 1862, 1868, 1873, 1880, 2230,
2232, 2248, 2250, 2254, 2271, 2273,
2280, 2282, 2304, 2317, 2330, 2332,
2354, 2394, 2472, 2474, 2483, 2587,
2612, 2620, 2624, 2659, 2925, 2957,
2961, 2964, 3074, 3079, 3081, 3089,
3096, 3101, 3105, 3144, 3163, 3180,
3186, 3193, 3209, 3220, 3235, 3253,
3261, 3263, 3350, 3357, 3394, 3415,
3431, 3435, 3445, 3463, 3465, 3547,
3550, 3554, 3891, 3893, 3895, 3897,
4014, 4031, 4070, 4081, 4083, 4236,
4268, 4279, 4329, 4506, 4510, 4586,
4595, 4600, 4621, 4770, 4889, 4929,
5148, 5153, 5249, 5254, 5259, 5278,
5281, 5284, 5304, 5356, 5365, 5389,
5397, 5513, 5864, 6147, 6219, 6299,
6839, 6842, 6844, 6848, 7024, 7037,
7051, 7086, 7260, 7315, 7414, 7611
\c_stex_debug_clist .....
... 28, 38, 66, 69, 90, 94, 96, 100, 104
\c_stex_default_metatheory 2176, 7571
\l_stex_default_notation .....
... 4538, 4543, 4546, 4548, 4549,
4566, 5119, 5293, 5298, 5319, 5653
\stex_do_default_notation: .....
..... 4528, 5118, 5291, 5652
\stex_do_default_notation_op: ...
..... 4528, 4529, 4564, 5316
\stex_do_deprecation:n .....
.... 117, 432, 432, 2310, 3718, 3819
\stex_do_for_list: .....
... 2994, 6805, 6820, 7184, 7255, 7310
\stex_do_id: .....
129, 437,
437, 1352, 6884, 6899, 7185, 7264,
7324, 7333, 7342, 7422, 8407, 9108
\stex_do_up_to_module:n .....
120, 2555, 2555, 2558, 2565, 3322, 6287
\l_stex_docheader_sect .....
118, 1277, 1303, 1319, 1327, 1329,
1333, 1357, 1369, 1371, 1374, 1377,
1383, 1396, 1398, 1402, 1413, 1414,
1415, 1416, 1417, 1418, 1420, 1423,
1429, 1433, 1439, 1443, 7858, 7865
\stex_eat_exclamation_point: ...
..... 3773, 5282, 5293, 5729, 5731
\stex_end: .....
421, 429, 2892, 2900
\stex_every_file: .....
..... 999, 1002, 1007, 1019, 7723
\stex_every_module:n .....
..... 119, 2292, 2295,
2579, 3310, 3494, 3515, 3539, 3607,
3628, 3649, 3743, 3968, 4186, 4526,
6137, 6211, 6274, 6284, 7579, 7603
\g_stex_every_module_t1 .....
..... 2293, 2296, 2308
\l_stex_every_symbol_t1 5163, 5168,
5170, 5171, 5177, 5178, 5181, 5219
\stex_execute_in_module:n .....
..... 119, 2309, 2383, 2563,
2563, 2567, 2598, 2630, 3279, 3522,
3952, 6127, 6187, 6199, 6238, 6255
\stex_fatal_error:n 139, 81, 81, 1119
\stex_fatal_error:nnn .....
.... 139, 81, 84, 87, 1130, 2032, 2101
\l_stex_feature_name_str .....
..... 126, 2836, 2840,
2869, 2940, 2945, 2961, 2962, 2964,
2965, 3008, 3353, 3369, 3403, 3407
\stex_file_in_smemode:nn .....
... 132, 2152, 2171, 2200, 2346, 3249
\stex_file_resolve:Nn .....
134,
135, 688, 693, 709, 716, 878, 979,
988, 990, 993, 1048, 1076, 1078,
1606, 1624, 1714, 3097, 3191, 3218
\stex_file_set:Nn .. 134, 135, 682,
682, 687, 700, 711, 874, 921, 1016, 1073
\stex_file_split_off_ext:NN .....
.... 134, 791, 791, 896, 2344, 3090
\stex_file_split_off_lang:NN ...
..... 134, 791,
798, 893, 2345, 3091, 8379, 8463, 8468
\stex_file_use:N .....
134,
135, 706, 713, 765, 765, 842, 847,
852, 906, 912, 927, 930, 983, 994,

```

```

998, 1053, 1073, 1079, 1084, 1128,
1131, 1164, 1165, 1167, 1201, 1607,
1608, 1620, 1699, 1705, 1710, 1715,
1717, 1865, 1872, 1877, 1889, 1892,
1916, 1921, 1932, 1988, 1998, 2038,
2042, 2052, 2056, 2066, 2071, 2347,
3093, 3096, 3097, 3099, 3120, 3135,
3145, 3164, 3192, 3208, 3219, 3234
\stex_filestack_pop: .....
... 134, 1008, 1008, 1033, 1040, 2197
\stex_filestack_push:n .....
.... 134, 986, 986, 1026, 1028, 2181
\l_stex_fors_seq ..... 2844,
2847, 6806, 6809, 6823, 6840, 6841,
6855, 6856, 6914, 6941, 7188, 7189,
7239, 7240, 7259, 7314, 7395, 7396
\stex_get_env:Nn .....
.... 139, 51, 52, 60, 88, 301, 605,
611, 975, 977, 1044, 1046, 1051, 3659
\stex_get_in_morphism:n .....
128, 2846, 2997, 2997, 3343, 3387, 3572
\_stex_get_mathstructure:n .....
..... 2805, 6164, 6169
\stex_get_mathstructure:n .....
..... 6163, 6163, 6181, 6230, 6769
\l_stex_get_structure_module_str
.. 2806, 6165, 6170, 6174, 6188, 6239
\_stex_get_symbol:n . 3986, 3992, 6171
\stex_get_symbol:n .....
121,
122, 128, 1827, 3985, 3985, 4150,
4425, 4930, 5237, 5923, 5931, 5942,
6808, 6966, 7017, 7070, 7679, 7700
\l_stex_get_symbol_args_tl .....
..... 121, 123,
231, 3046, 3376, 3762, 3881, 3883,
3884, 4020, 4061, 4093, 4104, 4106,
4119, 4534, 4799, 4812, 4894, 4994,
5007, 5242, 5407, 6152, 6832, 7656
\l_stex_get_symbol_arity_int ...
..... 121, 123, 231,
3045, 3375, 3761, 3803, 3827, 3833,
3837, 3841, 3845, 3849, 3853, 3857,
3861, 3865, 3868, 3869, 3870, 3871,
3882, 3940, 4012, 4019, 4060, 4092,
4196, 4231, 4263, 4270, 4290, 4340,
4457, 4769, 4798, 4811, 4852, 4893,
4993, 5006, 5045, 5241, 6831, 7655
\l_stex_get_symbol_def_tl .....
..... 121, 2865, 3047,
3351, 4021, 4062, 4094, 4895, 5243
\l_stex_get_symbol_invoke_cs ...
..... 121, 3050, 3380,
4024, 4065, 4097, 4898, 5246, 6173
\l_stex_get_symbol_macro_str ...
.....
..... 128, 3044, 3374
\l_stex_get_symbol_mod_str .....
.....
..... 121, 1829, 2848,
2858, 3042, 3358, 3362, 3372, 3397,
3406, 3944, 3950, 3993, 4017, 4058,
4090, 4156, 4160, 4175, 4338, 4427,
4431, 4435, 4439, 4446, 4450, 4507,
4514, 4891, 5239, 6003, 6810, 6969,
7019, 7072, 7665, 7671, 7692, 7713
\l_stex_get_symbol_name_str .....
121, 1829, 2848, 2859, 2998, 3002,
3006, 3043, 3350, 3352, 3358, 3362,
3372, 3394, 3397, 3406, 3407, 3945,
3950, 3987, 3994, 4018, 4059, 4091,
4156, 4160, 4174, 4175, 4338, 4427,
4431, 4435, 4439, 4446, 4450, 4464,
4469, 4472, 4508, 4514, 4892, 4918,
4920, 4926, 4928, 5240, 5933, 5944,
5966, 5977, 5979, 5990, 5992, 6003,
6004, 6005, 6006, 6172, 6810, 6969,
6970, 6971, 6972, 6999, 7007, 7015,
7019, 7060, 7068, 7072, 7666, 7671,
7678, 7680, 7692, 7699, 7701, 7713
\stex_get_symbol_or_var:n .....
..... 122, 4888, 4925
\l_stex_get_symbol_return_tl .....
..... 121, 3049, 3379, 4023,
4064, 4096, 4771, 4897, 4960, 5245
\l_stex_get_symbol_type_tl .....
121, 3048, 3378, 4022, 4063, 4095,
4896, 5244, 6174, 6182, 6232, 6771
\stex_get_var:n .....
122, 4463,
4888, 4917, 5964, 5975, 5988, 7058
\c_stex_home_file .....
135, 1042
\stex_if_check_terms: .....
..... 121, 3655, 3666, 3670
\stex_if_check_terms:TF 121, 3654,
3675, 4152, 4466, 4773, 4969, 4972
\stex_if_check_terms_p: ... 121, 3654
\stex_if_do_html: .....
282
\stex_if_do_html:TF .....
137,
279, 291, 1255, 1282, 1291, 1537,
2523, 2543, 2781, 2797, 2827, 2875,
3060, 3284, 3325, 3361, 3395, 3752,
3936, 3948, 4154, 4513, 4767, 4775,
4821, 4968, 6193, 6249, 6777, 6835,
7059, 7256, 7274, 7284, 7301, 7311,
7345, 7352, 7354, 7392, 7613, 7638,
7670, 8377, 8414, 8466, 8491, 8544,
8554, 8566, 8572, 8608, 8617, 8627,
8633, 8657, 8666, 8676, 8682, 8709,
8718, 8729, 8735, 8774, 8808, 8829,
8877, 8894, 8915, 8939, 8984, 9012

```

```

\stex_if_do_html_p: ..... 137
\stex_if_file_absolute:N 134, 748, 758
\stex_if_file_absolute:NTF .....
..... 134, 746, 992, 1077
\stex_if_file_absolute_p:N . 134, 746
\stex_if_file_starts_with:NN 134, 768
\stex_if_file_starts_with:NNTF ..
..... 134, 768, 1200
\stex_if_html_backend:TF .....
..... 137, 296, 321,
328, 337, 620, 1255, 1299, 1361,
1380, 1427, 1939, 1966, 3654, 5191,
5225, 5737, 5793, 5822, 6000, 6026,
6878, 6888, 6964, 7758, 7798, 7923,
8003, 8051, 8059, 8071, 8081, 8191
\stex_if_html_backend_p: ... 137, 296
\stex_if_in_module: ..... 2547
\stex_if_in_module:TF .....
..... 119, 2299, 2547, 2563
\stex_if_in_module_p: .... 119, 2547
\stex_if_module_exists:n .... 2551
\stex_if_module_exists:nTF .....
..... 119, 2316, 2329, 2551, 3148, 3151,
3167, 3170, 3255, 6183, 6233, 6772
\stex_if_module_exists_p:n 119, 2551
\stex_if_smsmode: ..... 2137
\stex_if_smsmode:TF .....
..... 132, 438, 1247, 1523, 2135,
2202, 2335, 2534, 2542, 3064, 3207,
3233, 3288, 3329, 3481, 3488, 3501,
3509, 3525, 3533, 3598, 3618, 3639,
3729, 3957, 4163, 4516, 6871, 6887,
6900, 6913, 6940, 7043, 8482, 8490
\stex_if_smsmode_p: ..... 132, 2135
\stex_ignore_spaces_and_pars: ...
. 139, 358, 358, 361, 2573, 2574, 8428
\l_stex_import_archive_str . 127,
2475, 2480, 2812, 3057, 3082, 3086,
3106, 3107, 3108, 3276, 3303, 3319
\stex_import_module_uri:nn .....
..... 127, 2473, 2809,
3055, 3073, 3073, 3274, 3300, 3317
\l_stex_import_name_str .....
127, 2477, 2482, 2814, 3059, 3067,
3076, 3278, 3292, 3305, 3321, 3333
\l_stex_import_ns_str .... 2483,
2486, 2816, 3062, 3066, 3149, 3152,
3155, 3168, 3171, 3174, 3182, 3280,
3283, 3286, 3291, 3323, 3327, 3332
\l_stex_import_path_str .....
..... 127, 2476, 2481,
2813, 3058, 3077, 3088, 3092, 3096,
3097, 3098, 3101, 3277, 3304, 3320
\stex_import_require_module:nnn .
..... 127, 2479, 2811,
3056, 3111, 3111, 3124, 3275, 3318
\stex_import_require_module_-
safe:nnn ..... 3126, 3302
\l_stex_import_uri_str .....
..... 127, 3087, 3108, 3114,
3119, 3129, 3134, 3142, 3144, 3148,
3149, 3155, 3161, 3163, 3167, 3168,
3174, 3193, 3200, 3220, 3255, 3256
\stex_in_archive:nn 131, 1093, 1093,
1887, 1938, 1963, 2021, 2074, 2112
\stex_in_invisible_html_bool .....
..... 3793, 4842, 5735, 5736, 5766
\l_stex_in_meta_bool 2378, 2379, 2384
\l_stex_inpararray_bool ..... 5866
\stex_input_with_hooks:n .....
.. 1021, 1889, 1892, 1913, 1930, 1932
\_stex_invoke_notation:w .....
..... 5154, 5271, 5272, 5277, 5314
\stex_invoke_outer_field: .... 6157
\stex_invoke_sequence: .....
..... 4998, 5011, 5062, 5062, 5603
\stex_invoke_sequence_in: .... 129
\stex_invoke_sequence_range: ... 129
\stex_invoke_structure: .....
..... 6125, 6173, 6293
\stex_invoke_symbol ..... 128
\stex_invoke_symbol: .....
..... 122, 128, 3766,
4803, 4816, 5248, 5248, 6833, 7660
\stex_invoke_symbol:nnnnnnnnN .....
..... 120, 128,
2616, 4001, 4015, 4016, 5189, 5189,
5202, 5238, 6699, 6713, 6721, 6726
\stex_invoke_symbol:nnnnnnnnN\l_-
stex_current_symbol_str ... 128
\stex_invoke_text_symbol: 3932, 3980
\stex_invoke_variable:nnnnnn . 5223
\stex_invoke_variable:nnnnnnN ..
..... 128, 4809, 5004, 5223, 5601
\stex_is_sequentialized:n .....
..... 5595, 5622, 5697, 5707
\stex_iterate_break: 120, 2672, 2675
\stex_iterate_break:n . 120, 2672,
2678, 2755, 3436, 3446, 3461, 4089
\stex_iterate_morphisms:nn .....
..... 2738, 2738, 3414, 3430
\stex_iterate_notations:nn .....
..... 123, 2713, 2713, 2934, 6733
\stex_iterate_symbols:n .....
..... 120, 2671, 2671, 4082
\stex_iterate_symbols:nn .....
120, 2688, 2688, 2897, 3464, 6141, 6684

```

```

\l_stex_key_answerclass_str .....
..... 8529, 8533, 8547, 8582, 8583
\l_stex_key_archive_str .....
..... 129, 386, 389, 1570, 1588,
1602, 1611, 1612, 1689, 1696, 1710
\l_stex_key_argnames_clist .... 121
\l_stex_key_args_str .....
.... 121, 3687, 3692, 3701, 3735,
3779, 3829, 3834, 3838, 3842, 3846,
3850, 3854, 3858, 3862, 3866, 3885,
4491, 4824, 4955, 4956, 5021, 6790
\l_stex_key_argtypes_clist .....
..... 3705, 3710, 3809, 3811,
3898, 4857, 4860, 5050, 5053, 6794
\l_stex_key_assoc_str 3690, 3695,
3783, 3784, 4828, 4829, 5025, 5026
\l_stex_key_autogradable_bool ...
..... 8321, 8327, 8332, 8385, 8474
\l_stex_key_continues_tl . 7097, 7106
\l_stex_key_def_tl 122, 3703, 3713,
3734, 3763, 3797, 3798, 3894, 3905,
3909, 3929, 4490, 4800, 4813, 4846,
4847, 4995, 5008, 5039, 5040, 6792
\l_stex_key_deprecate_str .....
..... 412, 414, 433, 434
\l_stex_key_due_tl .....
..... 9083, 9087, 9122, 9123
\l_stex_key_feedback_tl .....
.. 8846, 8851, 8871, 8872, 8933, 8934
\l_stex_key_file_str .....
..... 129, 387, 390, 1571,
1587, 1590, 1607, 1622, 1650, 1665,
1676, 1689, 1701, 1705, 1711, 1783
\l_stex_key_for_clist 2845, 6789,
6797, 6807, 7094, 7103, 7375, 7380
\l_stex_key_for_str .....
6877
\l_stex_key_from_tl .... 7095, 7104
\l_stex_key_Ftext_tl .....
..... 8849, 8855, 8869, 8931
\l_stex_key_functions_tl . 7099, 7107
\l_stex_key_given_tl .....
..... 9082, 9086, 9119, 9120
\l_stex_key_hide_bool 7101, 7110, 7243
\l_stex_key_id_str .....
..... 129, 394, 396, 439,
440, 6858, 6859, 6926, 6928, 7321,
7330, 7419, 8538, 8540, 8546, 8602,
8604, 8610, 8651, 8653, 8659, 8702,
8704, 8711, 8763, 8765, 8773, 8776
\l_stex_key_intent_args_clist ...
..... 4133, 4139, 4305, 4314, 4316
\l_stex_key_intent_str .....
..... 3942, 4132, 4138, 4228, 4308
\l_stex_key_macroname_str .....
.. 6788, 6798, 6815, 6817, 6826, 6830
\l_stex_key_method_tl .....
.. 7100, 7109, 7230, 7231, 7377, 7381
\l_stex_key_mrepos_str .....
.. 8326, 8334, 8512, 8514, 8522, 9135
\l_stex_key_min_tl .....
..... 8324, 8330, 8397, 8487, 8506
\l_stex_key_name_str .....
..... 121, 122, 3700, 3707,
3731, 3732, 3746, 3747, 3760, 3769,
3778, 3819, 3874, 3903, 3907, 3916,
3917, 3919, 3927, 3935, 3945, 3953,
3959, 3960, 4464, 4487, 4488, 4506,
4508, 4759, 4760, 4770, 4777, 4780,
4795, 4797, 4810, 4823, 4872, 4875,
4876, 4880, 4882, 4883, 4952, 4953,
4975, 4990, 4992, 5005, 5020, 6787,
6796, 6816, 6817, 6821, 6824, 6830,
6838, 6847, 6876, 7376, 7383, 7399,
7400, 7413, 7414, 7415, 7644, 7645,
7654, 7666, 8325, 8331, 8378, 8380,
8383, 8462, 8464, 8467, 8469, 8472
\l_stex_key_number_tl .....
..... 9081, 9085, 9101, 9102
\l_stex_key_op_tl 3941, 4131, 4137,
4189, 4190, 4191, 4198, 4199, 4206,
4211, 4342, 4376, 4380, 4438, 4443,
4459, 4879, 4881, 4884, 4962, 4964
\l_stex_key_post_tl .. 5912, 5916,
5933, 5944, 5977, 5990, 6999, 7007
\l_stex_key_pre_tl ... 5911, 5915,
5933, 5944, 5977, 5990, 6999, 7007
\l_stex_key_prec_str .. 123, 3943,
4130, 4136, 4240, 4243, 4246, 4252,
4255, 4258, 4261, 4267, 4279, 4280
\l_stex_key_proofend_tl .....
..... 7096, 7105, 7207, 7208
\l_stex_key_proot_tl ..... 5913
\l_stex_key_pts_str .....
.. 8754, 8757, 8777, 8778, 8790, 8791
\l_stex_key_pts_tl .....
..... 8323, 8329, 8396, 8484, 8501
\l_stex_key_reorder_str 3689, 3693,
3786, 3787, 4834, 4835, 5031, 5032
\l_stex_key_return_tl .....
... 3704, 3709, 3765, 3800, 3803,
3896, 4771, 4802, 4815, 4849, 4852,
4960, 4997, 5010, 5042, 5045, 6793
\l_stex_key_role_str .....
..... 3688, 3696, 3789, 3790,
3921, 4831, 4832, 5028, 5029, 6827
\l_stex_key_root_tl ..... 5917

```

```

\l_stex_key_schema_str ..... 8755, 8758, 8780, 8781, 8787, 8788
\l_stex_key_short_t1 ..... 1314, 1317, 1344, 1346
\l_stex_key_sig_str ..... 119, 2305, 2347, 2495, 2510, 2527
\l_stex_key_style_clist ..... 130, 406, 408, 458, 462, 509, 513, 6861, 6862, 6955
\l_stex_key_T_bool ..... 8847, 8852, 8853, 8866, 8879, 8928, 8941
\l_stex_key_term_t1 ..... 7098, 7108, 7227, 7228, 7378, 7382
\l_stex_key_testspace_dim ..... 8527, 8530, 8532, 8561, 8975, 8977, 9029, 9032
\l_stex_key_title_str ..... 6799
\l_stex_key_title_t1 ..... 400, 402, 1572, 1750, 1751, 2517, 6875, 6880, 8388, 8389, 8393, 8477, 8478, 8581, 8582, 8642, 8643, 8691, 8692, 8744, 8745, 9114, 9115
\l_stex_key_Ttext_t1 ..... 8848, 8854, 8867, 8929
\l_stex_key_type_t1 ..... 122, 3702, 3712, 3733, 3764, 3794, 3795, 3892, 3904, 3908, 4489, 4801, 4814, 4843, 4844, 5036, 5037, 6791, 6800
\l_stex_key_variant_str ..... 123, 4129, 4135, 4142, 4144, 4173, 4226, 4339, 4348, 4380, 4474, 4493, 4782, 4873, 4881, 4977
\stex_keys_define:nnnn ..... 129, 366, 366, 385, 393, 399, 405, 411, 417, 1343, 1556, 1562, 2494, 3686, 3699, 3902, 4128, 4484, 4750, 5910, 6100, 6786, 7093, 7374, 7950, 8322, 8511, 8528, 8598, 8753, 8845, 8973, 9080, 9179
\stex_keys_set:nn ..... 129, 130, 381, 381, 1350, 1574, 1775, 1780, 2516, 3721, 3913, 3914, 4149, 4462, 4503, 4757, 4950, 5922, 5930, 5941, 5963, 5974, 5987, 6115, 6868, 6897, 6991, 6997, 7005, 7183, 7254, 7309, 7412, 7608, 7642, 8030, 8369, 8373, 8452, 8521, 8537, 8560, 8601, 8650, 8701, 8762, 8807, 8864, 8893, 8926, 9011, 9095, 9099, 9134, 9192
\stex_kpsewhich:Nn 139, 43, 43, 54, 61
\c_stex_language_abbrevs_prop ..... 136, 110
\stex_language_from_file: ..... 137, 158, 158, 1004
\c_stex_languages_clist . 29, 181, 184
\c_stex_languages_prop ..... 136, 110, 138, 139, 173, 189, 805
\g_stex_last_feature_str ..... 125, 2795, 6141
\stex_macro_body:N . 138, 544, 546, 564
\stex_macro_definition:N 138, 559, 559
\l_stex_macroname_str ..... 122, 224, 2785, 2786, 3719, 3723, 3725, 3759, 3780, 3781, 3915, 3926, 4504, 4758, 4796, 4808, 4825, 4826, 4951, 4991, 5003, 5022, 5023, 6126, 6826, 7643, 7653
\_stex_main_archive: .... 1199, 1238
\c_stex_main_archive_prop . 131, 1199
\c_stex_main_file ..... 134, 135, 974, 1013, 1079, 7720
\stex_map_args:N ..... 3804, 4103, 4103, 4205, 4301, 4372, 4536, 4853, 5046, 5409, 6155
\stex_map_notation_args:N ..... 4103, 4116, 4405
\stex_map_uri:Nnnnn 135, 811, 818, 2362
\c_stex_mathhub_file 135, 918, 1042, 1118, 1128, 1131, 1141, 1200, 1620, 1699, 1710, 1865, 1877, 1889, 1892, 1916, 1921, 1932, 1984, 2038, 2042, 2052, 2056, 2066, 2071, 3120, 3135
\c_stex_mathhub_main_manifest_- prop ..... 1206, 1207
\stex_mathml_arg:nn ..... 138
\stex_mathml_intent:nn ..... 138
\_stex_maybe_brackets:nn ..... 4589, 4606, 5070, 5306, 5317, 5859
\stex_metagroup_do_in:nn ..... 120, 140, 232, 237, 244, 2556, 3368, 3402, 7650, 7651, 7662
\stex_metagroup_new:n ..... 140, 226, 227, 231, 2307, 2356, 2840
\l_stex_metatheory_uri 2176, 2382, 2385, 2469, 2485, 2499, 2501, 2528, 2529, 7569, 7570, 7571, 7618, 7619
\c_stex_module_ ..... 120, 122
\stex_module_add_code:n ..... 120, 2559, 2559, 2562, 2564, 2574, 7626
\stex_module_add_morphism:nnm .. 119
\stex_module_add_morphism:nnnn .. 127, 2580, 2580, 2585, 2944, 3282, 6197, 6253, 7629
\stex_module_add_notation:nnnnn . 122, 2623, 2623, 2634, 2936, 2939, 4337, 4456
\stex_module_add_symbol:nnnnnnN .. 119, 2586

```

```

\stex_module_add_symbol:nnnnnnnN
    .... 120, 2586, 2958, 2962, 2965,
    3758, 3925, 6121, 6153, 6829, 7652
\stex_module_setup:n .....
    119, 2298, 2298, 2521, 2791, 7489, 7609
\_stex_module_setup_top_nosig:n .
    .....
        2306, 2315, 2375, 7605
\l_stex_morphism_morphisms_seq ..
    .....
        126, 2873, 2883, 2978
\l_stex_morphism_renames_prop ...
    .....
        126, 2872, 2882,
        2929, 2935, 2948, 2956, 3003, 3405
\l_stex_morphism_symbols_prop ...
    .....
        126, 2861,
        2871, 2881, 2899, 2903, 2912, 2926,
        2931, 2999, 3017, 3022, 3371, 3467
\stex_new_statement:nn .....
    6805
\stex_new_statement:nnn .....
    .....
        6866, 6932, 6938, 6953, 6954
\stex_new_stylistable_cmd:nnnn .....
    .....
        130, 444, 444,
        3054, 3267, 3316, 3342, 3386, 3412,
        3593, 3613, 3634, 3720, 3912, 4148,
        4461, 4502, 4756, 4949, 7181, 7575
\stex_new_stylistable_env:nnnnnnn ...
    .....
        130, 444, 479, 2515, 3477,
        3499, 3519, 6092, 6178, 6227, 6867,
        7278, 7296, 7308, 8363, 8451, 8553,
        8616, 8665, 8717, 8806, 8892, 9092
\_stex_next_symbol:n .....
    ..
        5166, 5167, 5185, 6504, 6626, 6986
\c_stex_no_frontmatter_bool 35, 1450
\l_stex_notation_ .....
    123
\_stex_notation_add: .....
    ..
        3947, 4153, 4325, 4336, 4512, 7669
\l_stex_notation_args_tl .. 4117,
    4195, 4229, 4300, 4306, 4312, 4510
\l_stex_notation_check: 4152, 4325,
    4325, 4466, 4511, 4773, 4972, 7668
\l_stex_notation_do_html:n . 3950,
    4156, 4325, 4345, 4514, 4777, 7671
\l_stex_notation_downprec . 5525,
    5853, 5854, 5864, 5865, 5885, 5887
\l_stex_notation_macrocode_cs ...
    .....
        122–124, 4179, 4224,
        4236, 4330, 4341, 4370, 4430, 4458,
        4477, 4496, 4785, 4874, 4877, 4980
\l_stex_notation_make_args: 4180,
    4325, 4331, 4404, 4478, 4497, 4786
\stex_notation_parse:n .....
    .....
        3946, 4151, 4188, 4188,
        4465, 4509, 4772, 4963, 4966, 7667
\stex_notation_parse_and_then:nw
    .....
        123, 125
\_stex_notation_set_default:n ...
    .....
        4159, 4424, 4445, 4455, 4469
\stex_notation_top:nw .....
    125
\stex_persist:n .....
    119, 138, 617, 621, 622, 624, 627,
    630, 631, 637, 638, 1193, 1218, 2404
\c_stex_persist_mode_bool .....
    .....
        138, 31, 608, 671, 1237
\_stex_persist_read_now: .....
    .....
        670, 1217, 7722
\c_stex_persist_write_mode_bool .
    138, 32, 614, 619, 672, 678, 1216, 2393
\stex_pseudogroup:nn .....
    139,
    140, 206, 206, 287, 1097, 2661, 5398
\stex_pseudogroup_restore:N .....
    .....
        139, 140, 206, 209, 1110, 2665
\stex_pseudogroup_with:nn .....
    .....
        139, 216, 216, 819,
        941, 2384, 2672, 2690, 2715, 4015,
        4685, 4694, 4719, 5885, 5899, 6021
\c_stex_pwd_file .....
    135, 974, 994, 1200, 1201, 1717, 1872
\stex_reactivate_macro:N 139, 349,
    355, 2579, 2991, 2992, 2993, 3310,
    3312, 3495, 3516, 3540, 3608, 3629,
    3650, 3743, 3968, 4186, 4526, 6137,
    6212, 6275, 6918, 6919, 6920, 6921,
    6922, 6936, 6945, 6946, 6947, 6948,
    6949, 6950, 6951, 7265, 7266, 7267,
    7268, 7269, 7270, 7271, 7272, 7579,
    7603, 7631, 8348, 8349, 8350, 8351,
    8352, 8353, 8354, 8727, 8818, 8904
\stex_ref_new_doc_target:n .....
    .....
        118, 129, 440, 1505, 1518, 1542
\stex_ref_new_id:n . 1508, 1519, 6927
\stex_ref_new_sym_target:n .....
    118, 1788, 1812, 1821, 6915, 6942, 6976
\stex_ref_new_sym_target:nn 118, 1819
\stex_ref_new_symbol:n .....
    .....
        118, 1788, 2429, 3768, 3934
\l_stex_ref_url_str . 1515, 1536, 1538
\stex_renamedecl_do:nn .....
    .....
        3388, 3393, 3590
\stex_require_archive:n .....
    .....
        127, 131, 1087, 1116, 1116,
        1126, 1611, 1876, 3107, 3117, 3132
\stex_resolve_path_pair:Nnn .....
    .....
        131, 1861, 1861, 1883
\stex_return_args:nn .....
    .....
        3772, 3804, 4853, 5046
\l_stex_return_notation_tl .....
    .....
        5206, 5383, 5384, 6342,
        6350, 6512, 6513, 6593, 6619, 6620

```

```

\stex_set_current_archive:n . . .
    ... 131, 1086, 1086, 1106, 1212, 3251
\stex_set_current_namespace: . . .
    ..... 136, 970, 970, 1005
\stex_set_document_uri: . . .
    ..... 136, 966, 966, 1003
\stex_set_language:n . . .
    ..... 137, 136, 136, 157, 174, 2508
\stex_set_notation_macro:nnnn . .
    122, 123, 2623, 2640, 2644, 2644, 2656
\stex_sms_allow:N . . .
    ... 132, 2119, 2119, 2130, 2131, 2132, 2133, 2134
\stex_sms_allow_env:n . . .
    ... 133, 2119, 2127, 2546, 3497, 3518, 3542, 6135, 6209, 6271, 6892, 7640
\stex_sms_allow_escape:N . . .
    ... 133, 2119, 2123, 2493, 2578, 3309, 3347, 3391, 3610, 3631, 3652, 3744, 3969, 4187, 4527, 6908, 7032, 7676
\stex_sms_allow_import:Nn . . .
    ... 133, 2140, 2142, 3311
\stex_sms_allow_import_env:nn . .
    ... 133, 2140, 2146
\g_stex_sms_import_code . . .
    ... 133, 2151, 2174, 2192, 3301
\stex_smsmode_do: . . .
    ... 133, 2165, 2167, 2201, 2206, 2491, 2539, 2575, 3269, 3338, 3345, 3389, 3486, 3506, 3530, 3604, 3625, 3646, 3740, 3965, 4169, 4522, 6094, 6203, 6260, 6885, 6906, 7029, 7632, 7673
\stex_split_slash: . . .
    ... 5953, 5957, 6006, 6972
\_stex_sref_do_aux:n . . .
    ... 1794, 1801, 1805, 1808, 7485
\stex_str_if_ends_with:nn .. 133, 534
\stex_str_if_ends_with:nnTF . .
    ... 133, 534, 987, 3421, 3444, 4072
\stex_str_if_ends_with_p:nn . .
    ... 133, 534, 4047, 4087
\stex_str_if_starts_with:nn .. 133, 539
\stex_str_if_starts_with:nnTF . .
    ... 133, 420, 539, 2898, 6320, 6324, 7035
\stex_str_if_starts_with_p:nn . .
    ... 133, 539
\stex_structural_feature_-
    module:nn ... 125, 2780, 2780, 6131
\stex_structural_feature_module_-
    end: 125, 2780, 2794, 6096, 6205, 6267
\stex_structural_feature_-
    morphism:nnn .. 2801
\stex_structural_feature_-
    morphism:nnnn .. 127
\stex_structural_feature_-
    morphism:nnnn .. 126, 2802, 3479, 3500, 3521, 3594, 3614, 3635
\stex_structural_feature_-
    morphism_check_total: . . .
    ... 2911, 3508, 3532, 3623, 3644
\stex_structural_feature_-
    morphism_end: . 126, 2801, 2868, 3491, 3512, 3536, 3603, 3624, 3645
\stex_style_apply: . . .
    ... 130, 131, 444, 449, 484, 489, 2537, 2542, 3069, 3294, 3335, 3484, 3489, 3504, 3510, 3528, 3534, 3601, 3621, 3642, 3737, 3962, 4166, 4481, 4519, 4789, 4984, 6882, 6888, 7193, 7263, 7283, 7300, 7320, 7338, 7341, 7350, 8406, 8413, 8482, 8490, 8550, 8567, 8571, 8613, 8628, 8632, 8662, 8677, 8681, 8714, 8730, 8734, 8826, 8828, 8912, 8914, 9107, 9110
\stex_suppress_html:n . . .
    ... 137, 286, 286, 2152, 4219
\_stex_symdecl_check_terms: . . .
    ... 122, 3750, 3889, 3889, 3924, 4764
\stex_symdecl_do: . . .
    ... 121, 122, 3749, 3818, 3818, 3923, 4763, 4958, 6828, 7647
\_stex_symdecl_html: . . .
    ... 3753, 3776, 3937, 6835
\stex_symdecl_top:n . . .
    ... 122, 3727, 3745, 3745, 4505
\stex_symdef_styledefs: . 4486, 4518
\stex_term_arg:nnn . . .
    ... 5516, 5520, 5527, 5537
\stex_term_arg:nnnn .. 4420, 4650, 4658, 5520, 5521, 5552, 5586, 5623, 5708
\stex_term_arg_aB:nnnn .. 4663, 4671, 4678, 5542, 5550
\stex_term_do_aB_clist: .. 4685, 4686, 4695, 4699, 4720, 4725, 5547, 5559, 5588, 5638, 5667, 5715, 5718
\stex_term_oma:nnn . . .
    ... 4602, 5408, 5793, 5794, 5820
\stex_term_oma_or_omb:nnn . . .
    ... 4602, 4607, 4655, 4668
\stex_term_omb:nnn .. 4655, 4668, 5436, 5437, 5822, 5823, 5849
\stex_term_oms:nnn . . .
    ... 4380, 5193, 5195, 5729, 5739, 5788, 5792, 5924, 5934, 5945
\stex_term_oms_or_omv:nnn . . .
    ... 3982, 4590, 5071, 5193
```

```

5195, 5227, 5229, 5307, 5318, 5391,
5729, 5792, 6359, 6397, 6435, 6491
\stex_term_omv:nnn .....
..... 4942, 5227, 5229,
5729, 5752, 5789, 5968, 5981, 5994
\stex_undefined: ..... 41
\stex_uri_add_module:NNn .....
..... 136, 939, 939, 964, 1520, 7569
\stex_uri_from_current_file:Nn ..
..... 136, 929, 929, 967
\stex_uri_from_current_file_-
nolang:Nn ..... 136, 929, 935, 971
\stex_uri_from_repo_file ..... 136
\stex_uri_from_repo_file:NNNn ...
..... 135, 136, 892, 895, 931, 1625
\stex_uri_from_repo_file_-
nolang:NNNn ..... 136, 892, 892, 936
\stex_uri_resolve:Nn ..... 135,
877, 877, 880, 2485, 2501, 2505, 7488
\stex_uri_set:Nn 135, 827, 873, 876, 927
\stex_uri_use:N ..... 135, 860,
881, 885, 933, 968, 972, 1515, 1521,
1522, 1526, 1592, 1594, 1627, 1634,
1651, 1652, 2385, 2529, 3146, 3151,
3152, 3165, 3170, 3171, 6859, 7619
\stex_vardecl_notation_macro: ...
... 122, 4467, 4774, 4870, 4870, 4973
\stex_variable:nnnnnnnN . 4749, 4890
\l_stex_variables_prop .....
..... 122, 4747, 4795, 4902, 4990
stex internal commands:
\stex_annotation_env_str ... 301, 302
\stex_aux_apply_patch:n .. 450, 457
\stex_aux_apply_patch_begin:n ..
..... 485, 508
\stex_aux_apply_patch_end:n ...
..... 490, 523
\stex_aux_args:n . 548, 552, 555, 558
\stex_aux_end: ..... 544, 545, 548
\stex_aux_params:n 562, 593, 600, 603
\stex_aux_patch:nnn ..... 446, 471
\stex_aux_patch:nnnn ..... 481, 498
\stex_aux_prefix:n .. 560, 568, 582
\stex_aux_prefix_long:n .....
..... 574, 578, 584, 591
\stex_aux_split_at_bracket:w ..
..... 421, 429
\stex_aux_start: ..... 544, 547
\l_stex_aux_tl .....
..... 371, 372, 373, 375, 378, 379
\stex_debug_nn ..... 67, 70, 75
\l_stex_debug_cl ..... 90, 92, 95
\stex_debug_env_str .... 88, 89, 92
\__stex_doc_check_topsect: .....
..... 1428, 1434, 1440, 1446
\__stex_doc_do_section:n .....
..... 1326, 1330, 1334, 1351
\__stex_doc_maketitle: ... 1269, 1274
\__stex_doc_orig_backmatter .....
..... 1462, 1465, 1483
\__stex_doc_orig_frontmatter ...
..... 1452, 1455, 1472
\__stex_doc_set_title:n .. 1246, 1262
\__stex_doc_skip_fragment:n .....
..... 1391, 1397,
1401, 1405, 1406, 1407, 1408, 1409
\__stex_doc_skip_section: .....
..... 1359, 1381, 1387
\__stex_doc_skip_section_i: .....
..... 1368, 1371, 1374, 1382, 1387
\__stex_doc_title_html: .....
..... 1251, 1254, 1266
\g_stex_doc_title_tl .....
... 1241, 1243, 1250, 1256, 1261, 1264
\__stex_expr_aB_arg:nnnnn 5557, 5564
\__stex_expr_aB_simple_arg:nnnn
..... 5575, 5584
\__stex_expr_add_prop_arg:nnw ...
..... 5404, 5428, 5431
\__stex_expr_arg:n ..... 5405, 5441
\l__stex_expr_arg_counter_int ...
..... 5395, 5402, 5417, 5453, 5454
\__stex_expr_arg_do:nnn .....
..... 5455, 5461, 5477, 5512, 5519
\__stex_expr_arg_inner:nn .....
..... 5444, 5447, 5451, 5457
\__stex_expr_assoc_make_seq:nnn ...
..... 5619, 5648, 5727
\__stex_expr_assoc_seq:nnnnnnn ...
..... 5568, 5616
\__stex_expr_check:n ..... 5485
\__stex_expr_check:nTF ... 5454, 5460
\__stex_expr_check_b:nn .. 5409, 5434
\l__stex_expr_count_int .....
... 5549, 5555, 5565, 5586, 5623, 5708
\l__stex_expr_cs .... 5650, 5653, 5673
\l__stex_expr_customs_prop .....
..... 5400, 5412, 5413,
5414, 5429, 5465, 5471, 5486, 5489
\l__stex_expr_customs_seq .....
... 5401, 5418, 5419, 5420, 5430, 5487
\__stex_expr_do_aB_clist: .....
..... 5542, 5547, 5588, 5667
\__stex_expr_do_ab_next:nnn ...
..... 5408, 5410, 5436, 5437
\__stex_expr_do_headerterm:nn ...
..... 5717, 5749, 5762, 5765, 5790

```

```

\__stex_expr_do_ref:nNn ... 5924,
  5932, 5943, 5965, 5976, 5989, 5999
\__stex_expr_do_seqmap:nnnnnn ...
  ..... 5573, 5680
\__stex_expr_end: ..... 5570, 5580
\__stex_expr_gobble:nnnnnnnn ...
  ..... 5570, 5580
\l__stex_expr_iarg_tl 5660, 5662, 5673
\__stex_expr_invoke_custom:n ...
  ..... 5255, 5269, 5396
\__stex_expr_invoke_math: 5250, 5258
\__stex_expr_invoke_op_custom:n .
  ..... 5255, 5262, 5388
\__stex_expr_invoke_op_notation:w
  ..... 5264, 5265, 5303
\__stex_expr_invoke_return: ...
  ..... 5344, 5347
\__stex_expr_invoke_return_-
  maybe:n ..... 5286, 5296, 5325
\__stex_expr_invoke_return_next:
  ..... 5332, 5341
\__stex_expr_invoke_text: 5250, 5253
\__stex_expr_is_seqmap:n ...
  ..... 5609
\__stex_expr_is_seqmap:nTF ...
  ..... 5572
\__stex_expr_is_varseq:n ...
  ..... 5598
\__stex_expr_is_varseq:nTF 5566, 5685
\l__stex_expr_left_bracket_str ..
  ..... 5855, 5881, 5889, 5899, 5900
\l__stex_expr_old_seq ...
  ..... 5692, 5695, 5699, 5704
\l__stex_expr_reset_tl ...
  ..... 5166, 5169, 5173, 5174, 5180
\__stex_expr_ret_cs ...
  ..... 5329, 5334, 5361, 5366, 5371
\__stex_expr_return_arg:n 5331, 5337
\l__stex_expr_return_args_tl ...
  .. 5326, 5338, 5343, 5352, 5368, 5373
\__stex_expr_return_notation:n ...
  ..... 5349, 5382
\l__stex_expr_return_this_tl ...
  ..... 5327, 5343,
  5348, 5352, 5356, 5357, 5367, 5375
\l__stex_expr_right_bracket_str ..
  ..... 5856, 5882, 5894, 5899, 5901
\__stex_expr_setup:nnnnnn ...
  ..... 5192, 5204, 5226
\__stex_expr_varseq_in_map:nnnnnnnn
  ..... 5687, 5726
\__stex_features_add_definiens:nn
  ..... 2852, 2995
\__stex_features_break: ... 2853, 2857
\__stex_features_check_break:nnnnnnnn
  ..... 3016, 3021, 3030, 3033, 3040
\__stex_features_clean:nnw 2892, 2900
\__stex_features_do_decls: 2884, 2896
\__stex_features_do_elaboration:
  ..... 2874, 2924
\__stex_features_do_for_list:
  ..... 2843, 2994
\__stex_features_do_morph:nnmn ...
  ..... 2972, 2976
\__stex_features_do_morphisms:n .
  ..... 2885, 2970, 2980
\__stex_features_elab_check: ...
  ..... 2932, 2955
\l__stex_features_feature_str ...
  ..... 2835, 2947
\__stex_features_get_check:nnnn ...
  ..... 3000, 3028
\l__stex_features_implicit_bool .
  ..... 2801, 2820, 2823, 2960
\__stex_features_reactivate: ...
  ..... 2838, 2983
\__stex_features_rename:nn 2948, 2951
\__stex_features_rename_all: . 2888
\__stex_features_renamed_-
  check:nnnn ..... 3004, 3014
\__stex_features_set_definiens_-
  macros: ..... 2853, 2857
\__stex_features_set_definiens_-
  macros_i:nnnnnnn ..... 2860, 2864
\__stex_features_setup: ... 2837, 2880
\__stex_features_split_qm:w ...
  ..... 2923, 2941
\l__stex_features_tmp ...
  ..... 2935, 2937, 2956, 2957, 2958
\__stex_features_total_check: ...
  ..... 2913, 2917
\__stex_groups_do: ... 257, 264, 273
\__stex_groups_do_in:nn 239, 246, 267
\__stex_groups_exists:n ...
  ..... 232
\__stex_groups_exists:nTF ...
  ..... 238
\l__stex_groups_ids_seq 226, 229, 265
\__stex_groups_tmp ... 248, 255, 261
\l__stex_importmodule_archive_-
  str 3113, 3118, 3128, 3133, 3250, 3251
\__stex_importmodule_check_-
  file:nn ..... 3194,
  3195, 3196, 3197, 3198, 3199, 3221,
  3222, 3223, 3224, 3225, 3226, 3260
\__stex_importmodule_get_from_-
  file:nnn ..... 3154, 3190
\__stex_importmodule_get_from_-
  file_safe:nnn ..... 3173, 3217
\__stex_importmodule_get_-
  module:nnn ..... 3115, 3121, 3179
\__stex_importmodule_get_module_-
  safe:nnn ..... 3130, 3136, 3185

```

```

\__stex_importmodule_get_module_-
    uri:nnn ..... 3140, 3181
\__stex_importmodule_get_module_-
    uri_safe:nnn ..... 3159, 3187
\__stex_importmodule_import_-
    module:nn ..... 3268, 3273, 3313
\__stex_importmodule_import_-
    module_presms:nn ..... 3299, 3313
\__stex_importmodule_load_file:n
    ..... 3211, 3214, 3238, 3243, 3248
\l__stex_importmodule_path_seq ..
    ..... 3090, 3091, 3093
\__stex_importmodule_seq .....
    ..... 3075, 3076, 3077
\l__stex_importmodule_seq .....
    .. 3097, 3099, 3191, 3192, 3218, 3219
\l__stex_importmodule_str .....
    ..... 3120, 3121, 3135,
    3136, 3192, 3208, 3219, 3234, 3237,
    3242, 3249, 3253, 3261, 3262, 3264
\__stex_inputs_bibresource:n ...
    ..... 1949, 1961, 1963
\l__stex_inputs_gin_repo_str ...
    ..... 2035, 2065, 2069, 2074
\l__stex_inputs_id_seq .....
    ..... 1985, 1987, 1994
\l__stex_inputs_id_str ... 1980, 1985
\__stex_inputs_inputref:nn 1937, 1947
\__stex_inputs_inputref_html:nn .
    ..... 1902, 1940
\__stex_inputs_inputref_pdf:nn ..
    ..... 1926, 1941
\__stex_inputs_libinput:n .....
    ..... 2006, 2019, 2021
\l__stex_inputs_libinput_files_-
    seq ..... 1951,
    1954, 1983, 1990, 1991, 2000, 2001,
    2008, 2011, 2026, 2027, 2098, 2099
\__stex_inputs_mhinput:nn 1886, 1899
\l__stex_inputs_path_seq .....
    ..... 1984, 1988, 1995, 1998
\l__stex_inputs_path_str .....
    ..... 1988, 1989, 1990, 1991,
    1994, 1995, 1998, 1999, 2000, 2001
\l__stex_inputs_tmp_str ... 2027, 2029
\__stex_inputs_up_archive:nn ...
    ..... 1950, 1976, 2007, 2025, 2097
\__stex_inputs_usetikzlibrary:n ..
    ..... 2096, 2110, 2112
\__stex_inputs_usetikzlibrary_-
    i:nn ..... 2080, 2099
\l__stex_iterate_continue_bool ..
    ..... 2740, 2743, 2756, 2776
\__stex_iterate_it_decl_check:nnnn
    ..... 2701, 2708
\__stex_iterate_it_decl_i:n .....
    ..... 2693, 2697, 2710
\__stex_iterate_it_not_check:nnnn
    ..... 2729, 2733
\__stex_iterate_it_not_i:n .....
    ..... 2718, 2722, 2735
\__stex_iterate_iterate_morphism:nn
    ..... 2746, 2750, 2759, 2762
\l__stex_iterate_mods_seq .....
    ..... 2689, 2698, 2699,
    2714, 2723, 2724, 2739, 2764, 2765
\__stex_iterate_morphism_cs:nnnn
    ..... 2741, 2767
\__stex_iterate_not_cs:nnnnn ...
    ..... 2715, 2716, 2726
\__stex_iterate_sym_cs:nnnnnnnn
    .. 2672, 2673, 2683, 2690, 2691, 2704
\l__stex_iterate_todo_tl .....
    ..... 2745, 2749, 2763, 2776
\l__stex_lang_lang_str . 139, 142, 149
\l__stex_lang_str .. 171, 172, 173, 174
\l__stex_lang_turkish_bool .....
    ..... 182, 187, 197
\l__stex_mathhub_bool .... 1139,
    1140, 1142, 1145, 1155, 1157, 1166
\__stex_mathhub_check_manifest: .
    ..... 1144, 1153
\__stex_mathhub_check_manifest:n
    ..... 1154, 1156, 1158, 1163
\l__stex_mathhub_cs .. 1094, 1095,
    1098, 1101, 1103, 1107, 1111, 1112
\__stex_mathhub_do_manifest:n ...
    ..... 1122, 1127
\__stex_mathhub_find_manifest:n .
    ..... 1128, 1136, 1151, 1201
\l__stex_mathhub_key 1178, 1179, 1181
\c__stex_mathhub_manifest_ior ...
    ..... 1171, 1173, 1175, 1190
\l__stex_mathhub_manifest_str ...
    ..... 1129, 1137, 1167, 1173, 1202
\__stex_mathhub_parse_manifest:n
    ..... 1133, 1172, 1205
\l__stex_mathhub_prop .....
    ..... 1174, 1182, 1183,
    1184, 1185, 1186, 1191, 1192, 1195
\l__stex_mathhub_seq .....
    ..... 1138, 1141, 1146,
    1164, 1165, 1167, 1177, 1178, 1180
\l__stex_mathhub_str .....
    1044, 1046, 1048, 1051, 1052, 1056,
    1057, 1062, 1068, 1071, 1076, 1079,
    1209, 1210, 1212, 1219, 1223, 1226

```

```

\l__stex_mathhub_tl ..... 1146
\l__stex_mathhub_val .....
... 1180, 1182, 1183, 1184, 1185, 1186
\__stex_module_setup_end: 2353, 2374
\__stex_module_setup_get_uri_-
str:n ..... 2303, 2360
\__stex_module_setup_load_meta: ..
..... 2312, 2381
\__stex_module_setup_load_sig: ...
..... 2333, 2343
\l__stex_module_setup_ns_str ...
..... 2304, 2306, 2361, 2363, 2370
\l__stex_module_setup_seg .....
..... 2366, 2367, 2368
\l__stex_module_setup_seq .....
..... 2365, 2366, 2368, 2370
\__stex_module_setup_setup_-
nested:n ..... 2299, 2351
\__stex_module_setup_setup_top:n
..... 2299, 2302
\__stex_module_setup_setup_top_-
sig:n ..... 2306, 2328
\l__stex_module_setup_sigfile ...
..... 2344, 2345, 2347
\__stex_module_setup_split_-
module:n ..... 2353, 2374
\__stex_modules_activate_-
i:nnnnnnnn ..... 2607, 2611
\__stex_modules_activate_not:nn ..
..... 2631, 2637
\__stex_modules_activate_sym:n ..
..... 2599, 2604
\__stex_modules_export:n .. 2570, 2572
\__stex_modules_persist_module: ..
..... 2393, 2403, 7483
\__stex_modules_persist_nots_-
i:nn ..... 2419, 2440
\l__stex_modules_restore_mod_str
..... 2433, 2461
\__stex_modules_restore_module:nnnn
..... 2405, 2424
\__stex_modules_restore_nots:n ..
..... 2437, 2446
\__stex_modules_restore_nots_i:n
..... 2447, 2450, 2467
\__stex_modules_restore_nots_-
ii:nnnn ..... 2454, 2458
\__stex_modules_set_metatheory:nn
..... 2470, 2490
\__stex_modules_tl ..... 2426, 2427
\l__stex_modules_tl ..... 2459, 2464
\l__stex_morphisms_ass_tl .....
..... 3546, 3562, 3566, 3576, 3577
\__stex_morphisms_do_morph_-
assign:nnn .... 3419, 3422, 3460
\__stex_morphisms_do_parsed_-
assign: ..... 3568, 3571
\__stex_morphisms_do_parsed_-
newname: ..... 3574, 3581
\__stex_morphisms_do_parsed_-
newname:w ..... 3583, 3585, 3589
\__stex_morphisms_end: ... 3574, 3589
\l__stex_morphisms_morphism_dom_-
str ... 3413, 3426, 3438, 3448, 3462
\l__stex_morphisms_name_str ...
..... 3544, 3553,
3554, 3555, 3559, 3560, 3561, 3572
\l__stex_morphisms_newname_str ...
..... 3545, 3564, 3565, 3573, 3574
\l__stex_morphisms_next_tl .....
..... 3551, 3556, 3558
\__stex_morphisms_parse_assign:n
..... 3543, 3596, 3616, 3637
\l__stex_morphisms_seq .....
..... 3548, 3549, 3551, 3553,
3556, 3558, 3560, 3562, 3564, 3566
\__stex_notations_add:nnnnn ...
..... 4636, 4641, 4645
\__stex_notations_add_last:nnnnm
..... 4629, 4640
\__stex_notations_add_missing_-
args:nn ..... 4205, 4320
\__stex_notations_add_next:nnnnn
..... 4631, 4635
\l__stex_notations_after_tl ...
..... 4615, 4642
\__stex_notations_args_end: ...
..... 4106, 4109, 4112,
4119, 4122, 4125, 4624, 4627, 4637
\l__stex_notations_args_tl ...
..... 4531, 4535, 4548
\__stex_notations_augment_arg:nn
..... 4536, 4553
\__stex_notations_check_aB_-
arg:Nn ..... 4676, 4684, 4693, 4718
\l__stex_notations_clist_count_-
int ..... 4716, 4724, 4730
\l__stex_notations_code_tl ...
... 4572, 4587, 4595, 4603, 4617,
4618, 4621, 4649, 4657, 4662, 4670
\__stex_notations_complex:nnnnnn
..... 4581, 4599
\__stex_notations_const_precs: ...
..... 4197, 4239
\l__stex_notations_cs ...
..... 4588, 4593, 4604, 4613

```

`__stex_notations_do_argname:nn .` 4301, 4304
`__stex_notations_do_argnames: ..` 4275, 4299
`__stex_notations_fun_precs: ...` 4202, 4251
`__stex_notations_make_arg:nnnn .` 4405, 4409
`__stex_notations_make_arg_- html:nn ..` 4372, 4387
`__stex_notations_make_name: ...` 4533, 4557, 4565
`__stex_notations_map_args_i:w ..` 4105, 4109, 4112
`__stex_notations_map_args_ii:w ..` 4118, 4122, 4125
`__stex_notations_map_cs: ..` 4358, 4360,
 4696, 4698, 4706, 4721, 4723, 4734
`\l__stex_notations_missing_str ..` 4203, 4321
`\l__stex_notations_missing_tl ..` 4204, 4233, 4322
`\l__stex_notations_name_str ..` 4559, 4560, 4561, 4566
`\l__stex_notations_opprec_tl` 4227,
 4241, 4244, 4246, 4253, 4256, 4258,
 4262, 4269, 4282, 4288, 4294, 4349
`__stex_notations_parse_notation_- args:nnnnw ..` 4624, 4627, 4637
`__stex_notations_parse_precs: ..` 4273, 4278
`\l__stex_notations_pre_tl ..` 4602, 4617, 4654, 4667
`\l__stex_notations_precs_seq ..` 4194, 4264,
 4271, 4292, 4294, 4307, 4313, 4350
`__stex_notations_process_- notation:nnnnnn ..` 4571, 4577
`\l__stex_notations_seq ..` 4280, 4281, 4283, 4284, 4291
`__stex_notations_simple:nnnnn ..` 4579, 4585
`\l__stex_notations_str ..` 4281, 4282, 4283, 4285, 4291, 4292
`__stex_notations_styledefs: ..` 4165, 4172
`__stex_path_:` 689, 698
`\l__stex_path_a_seq ..` 769, 775, 781
`\l__stex_path_a_tl ..` 781, 783
`__stex_path_auth:n ..` 135, 811, 814, 815,
 816, 819, 820, 865, 881, 886, 940, 942
`\l__stex_path_auth_str ..` 833, 841, 846, 851
`\l__stex_path_b_seq ..` 770, 776, 782, 788
`\l__stex_path_b_tl ..` 782, 783
`__stex_path_canonicalize:N ..` 701, 712, 718
`__stex_path_colonslash ..` 827, 832, 837
`\l__stex_path_do_hooks_pre_tl ..` 1022, 1023, 1035, 1038
`__stex_path_dodots:n ..` 723, 727, 733
`\l__stex_path_file ..` 900, 906, 912, 917
`__stex_path_from_repo_file:Nnnnn ..` 893, 896, 899
`\l__stex_path_maybewin_str ..` 750, 751, 752
`\l__stex_path_mod ..` 844, 847, 852
`__stex_path_module:n ..` 135,
 815, 819, 822, 867, 881, 942, 944, 946
`\l__stex_path_name ..` 849, 852
`__stex_path_name:n ..` 135,
 816, 819, 823, 868, 881, 942, 951, 953
`\l__stex_path_path ..` 838, 839, 842, 847, 852, 921, 922
`__stex_path_path:n ..` 135, 814, 819, 821, 866, 881, 942
`__stex_path_relativize:N ..` 909, 916
`\l__stex_path_return_tl ..` 771, 777, 784, 787, 789
`\l__stex_path_seq ..` 721, 724, 729, 737,
 738, 741, 794, 795, 796, 801, 802,
 804, 806, 809, 917, 919, 923, 925, 927
`\g__stex_path_stack ..` 984, 998, 1009, 1010, 1012, 1015
`\l__stex_path_str ..` 691, 694, 696, 697, 698, 700, 703,
 710, 711, 720, 722, 726, 738, 793,
 794, 795, 800, 801, 802, 804, 805,
 806, 975, 977, 979, 1010, 1015, 1016
`\l__stex_path_tl ..` 919, 923, 925
`\l__stex_path_uri ..` 902, 903, 904, 909, 927
`__stex_path_uri_set:NnN ..` 828, 874, 878
`__stex_path_uri_set:Nnnnn ..` 841, 846, 851, 863, 871, 905, 911
`__stex_path_uri_use:w ..` 881, 887
`\l__stex_path_win_drive ..` 690, 695, 702, 704
`__stex_path_win_take:w ..` 689, 698
`__stex_persist_env_str ..` 605, 606, 607, 611, 612, 613
`__stex_persist_load_file:n ..` 634, 666, 675
`__stex_persist_read_and_write: ..` 651, 673

```

\c__stex_persist_sms_iow .....
..... 617, 625, 647, 648, 663
\__stex_persist_write_only: .....
..... 646, 660, 667, 678
\__stex_proof_add_counter: 7156, 7337
\__stex_proof_begin_proof:nn ...
..... 7251, 7279, 7297
\l__stex_proof_counter_intarray ...
..... 7114, 7119,
7122, 7131, 7134, 7143, 7151, 7152,
7160, 7165, 7172, 7178, 7252, 7253
\__stex_proof_end_list: .....
.. 7221, 7292, 7353, 7363, 7429, 7452
\__stex_proof_html: . 7225, 7247, 7403
\__stex_proof_html_env:n .....
..... 7236, 7257, 7312
\l__stex_proof_in_spfblock_bool ..
... 7250, 7280, 7291, 7298, 7326,
7329, 7348, 7351, 7353, 7358, 7361,
7369, 7411, 7425, 7447, 7467, 7473
\__stex_proof_inblock_restore: ..
..... 7355, 7368, 7469
\__stex_proof_inc_counter: ...
..... 7139, 7351, 7442, 7444
\l__stex_proof_inc_counter_bool 7113
\__stex_proof_insert_number: ...
..... 7115, 7335, 7442, 7444
\__stex_proof_make_step_macro:Nnnnn
..... 7409, 7442, 7443, 7444
\__stex_proof_number_as_string:N
..... 7126, 7322, 7331, 7420
\__stex_proof_proof_box_tl 7096, 7200
\__stex_proof_remove_counter: ...
..... 7168, 7362
\__stex_proof_start_list:n .....
.... 7214, 7280, 7335, 7429, 7452
\__stex_proof_step_html:nn 7391,
7427, 7429, 7435, 7450, 7452, 7456
\__stex_refs_add_doc_ref:nn ...
..... 1522, 1544, 1555
\l__stex_refs_default_archive_-
str ..... 1564, 1570, 1579
\l__stex_refs_default_file_str ..
..... 1565, 1571, 1580
\l__stex_refs_default_title_tl ..
..... 1566, 1572, 1581
\__stex_refs_do_autoref:n .....
..... 1640, 1652, 1663,
1667, 1678, 1692, 1718, 1729, 1737
\__stex_refs_do_internal_link:nn
..... 1832, 1844, 1850
\__stex_refs_do_return:nnnn ...
..... 1746, 1760, 1768
\__stex_refs_do_sref:nn .. 1648, 1777
\__stex_refs_do_sref_in:n .....
..... 1656, 1671, 1682, 1688, 1786
\__stex_refs_do_url_link:nn ...
..... 1838, 1852
\l__stex_refs_file ...
.. 1606, 1608, 1624, 1626, 1714, 1715
\l__stex_refs_file_str ...
..... 1698, 1704, 1709, 1714, 1715, 1716,
1717, 1722, 1726, 1728, 1736, 1748
\g__stex_refs_files_seq .....
..... 1505, 1545, 1550, 1597, 1659
\__stex_refs_find_uri:n .....
..... 1584, 1776, 1781
\__stex_refs_find_uri_in_-
file:nnn .... 1593, 1598, 1630
\__stex_refs_find_uri_in_prop_-
file:N .... 1604, 1613, 1618
\l__stex_refs_id_str ...
..... 1724, 1759, 1765, 1766
\c__stex_refs_iow ...
..... 1496, 1497, 1498, 1524
\__stex_refs_new_symbol:n 1790, 1798
\l__stex_refs_prop ...
..... 1612, 1613
\__stex_refs_restore_target:nnnn
..... 1725, 1757
\l__stex_refs_return_tl ...
..... 1723, 1727, 1733, 1747
\__stex_refs_set_keys_b:n ...
..... 1568, 1655, 1670, 1681, 1782
\l__stex_refs_str 1511, 1513, 1515,
1520, 1522, 1527, 1800, 1802, 6928
\__stex_refs_sym_aux:nn ...
..... 1829, 1842, 1847, 1858
\l__stex_refs_unnamed_counter_-
int ...
..... 1506, 1510, 1511
\l__stex_refs_uri ...
..... 1520, 1521, 1625, 1627
\l__stex_refs_uri_str ...
... 1585, 1596, 1608, 1619, 1624,
1627, 1634, 1649, 1659, 1660, 1661,
1662, 1663, 1666, 1667, 1669, 1675,
1677, 1678, 1680, 1690, 1691, 1692,
1719, 1730, 1738, 1758, 1763, 1764
\__stex_seqs_add: .... 4970, 4988
\l__stex_seqs_args_tl ... 5046, 5048
\__stex_seqs_check_terms: 4969, 5016
\l__stex_seqs_clist ...
..... 5083, 5090, 5097, 5101
\l__stex_seqs_cs ...
..... 5044, 5048, 5104, 5108,
5116, 5119, 5128, 5135, 5148, 5149
\__stex_seqs_do_all:w ... 5142, 5152
\__stex_seqs_do_first: ... 5065, 5126

```

```

\__stex_seqs_do_first_arg:n . . .
..... 5124, 5131
\__stex_seqs_do_first_next: . . .
..... 5133, 5138
\__stex_seqs_do_one:w . . . 5140, 5146
\__stex_seqs_do_op:w . . . 5064, 5068
\__stex_seqs_doop_arg:n .. 5084, 5095
\__stex_seqs_doop_range:w 5077, 5081
\l__stex_seqs_first_args_tl . . .
..... 5130, 5148, 5149, 5153, 5154
\__stex_seqs_get_index_notation:n
..... 5076, 5114, 5147
\__stex_seqs_html: . . . 4968, 5018
\__stex_seqs_macro: . . . 4971, 5002
\__stex_seqs_make_args: .. 4981, 5015
\l__stex_seqs_range_clist . . .
..... 4961, 4996, 5009
\g__stex_smsemode_allowed_escape-
tl . . . . . 2117, 2124, 2253
\g__stex_smsemode_allowed_import-
env_seq 2141, 2147, 2187, 2236, 2239
\g__stex_smsemode_allowed_import-
tl . . . . . 2140, 2143, 2184, 2231
\g__stex_smsemode_allowed_tl . . .
..... 2116, 2120, 2249
\g__stex_smsemode_allowedenvs_seq
..... 2118, 2128, 2258, 2261
\g__stex_smsemode_bool . . . .
..... 2135, 2136, 2138, 2154
\__stex_smsemode_check_begin:Nn ..
..... 2236, 2258, 2270
\__stex_smsemode_check_cs:NNn . .
..... 2209, 2217
\__stex_smsemode_check_end:Nn . .
..... 2239, 2261, 2279
\__stex_smsemode_do:w . . .
..... 2203, 2215, 2217, 2219,
2241, 2251, 2263, 2276, 2283, 2286
\__stex_smsemode_do_aux:N . 2217, 2223
\__stex_smsemode_do_aux_curr:N . .
..... 2183, 2194, 2225
\__stex_smsemode_do_aux_imports:N
..... 2183, 2229
\__stex_smsemode_do_aux_normal:N .
..... 2194, 2247
\l__stex_smsemode_importmodules-
seq . . . . . 2172
\__stex_smsemode_in_smsemode:n . .
..... 2152, 2182, 2192, 2193
\l__stex_smsemode_sigmodules_seq 2173
\__stex_smsemode_smsemode_do: . .
..... 2165, 2201
\__stex_smsemode_start_smsemode:n .
..... 2163, 2190, 2195
\__stex_statements_do_defref:nn .
..... 6963, 6992, 6998, 7006
\__stex_statements_force_id: . .
..... 6822, 6925, 6933, 6956
\__stex_statements_html_keyvals:nn
..... 6852, 6873, 6901
\__stex_statements_setup:nn . .
.. 6814, 6934, 6939, 6953, 6957, 6960
\__stex_statements_setup_def: . .
..... 6912, 6935, 6958
\l__stex_statements_uri_str . .
6837, 6841, 6842, 6847, 6848, 7018,
7021, 7024, 7026, 7071, 7074, 7077
\l__stex_structures_assigned_seq
..... 6500, 6611, 6650, 6666
\__stex_structures_begin:nn . .
..... 6093, 6114, 6191, 6246
\__stex_structures_check-
def:nnnnnnnn . . . . . 6265, 6278
\l__stex_structures_clist . . .
..... 6474, 6495, 6501, 6503
\l__stex_structures_comp_cs . .
..... 6543, 6548
\l__stex_structures_cs . . .
..... 6476, 6482, 6489
\__stex_structures_current_type:
..... 6430, 6561, 6632
\l__stex_structures_current-
type_tl . . . 6357, 6393, 6433, 6438
\__stex_structures_do_assign:nn .
..... 6377, 6381
\__stex_structures_do_assign-
list:n . . . . . 6353, 6374
\__stex_structures_do_decl:nnnnnnnn
..... 6688, 6708
\__stex_structures_do_decl-
nomacro:nnnnnnnn . . . 6686, 6694
\__stex_structures_do_externals:
..... 6097, 6139, 6206, 6268
\__stex_structures_end: . . .
..... 6321, 6325, 6341, 6346
\l__stex_structures_exstruct-
name_str . . . . . 6241, 6246, 6290
\__stex_structures_extend-
structure:nn . . . . . 6218, 6241
\__stex_structures_extend-
structure_i:NnnnnnnnN . 6215, 6223
\__stex_structures_external-
decl:nnnn . . . . . 6142, 6146
\l__stex_structures_extmod_str . .
..... 6216, 6220
\l__stex_structures_extname-
count . . . . . 6284, 6288, 6290

```

```

\l__stex_structures_field_name_-
    str ..... 6598, 6602, 6603, 6634
\l__stex_structures_fields_clist
    ..... 6354,
    6375, 6382, 6400, 6403, 6657, 6658
\l__stex_structures_get_field_-
    name:n ..... 6597, 6609
\l__stex_structures_imports_seq .
    ..... 6179, 6184, 6192,
    6229, 6234, 6248, 6770, 6773, 6776
\l__stex_structures_invocation_-
    type:n ..... 6305, 6352
\l__stex_structures_invoke_-
    field:n ..... 6575, 6607
\l__stex_structures_invoke_maybe_-
    field:nn ..... 6564, 6568
\l__stex_structures_invoke_this:n
    ..... 6314, 6556
\l__stex_structures_invoke_top:n .
    ..... 6295,
    6298, 6322, 6326, 6329, 6332, 6334
\l__stex_structures_make_mod:n ...
    ..... 6399, 6411
\l__stex_structures_make_oml:n ...
    ..... 6403, 6417
\l__stex_structures_make_oml:nn ...
    ..... 6418, 6420
\l__stex_structures_make_prop: ...
    ..... 6442, 6569, 6647
\l__stex_structures_make_prop_-
    assign: ..... 6443, 6572, 6656
\l__stex_structures_make_prop_-
    assign:nn ..... 6659, 6664
\l__stex_structures_make_prop_-
    assign_replace:nnnn ... 6667, 6673
\l__stex_structures_make_type:n ..
    ..... 6362, 6367, 6369, 6385
\l__stex_structures_maybe_-
    notation:w ..... 6309, 6311, 6437
\l__stex_structures_merge:nw ...
    ..... 6303, 6319
\l__stex_structures_more_-
    nextsymbol_tl ... 6610, 6612, 6637
\l__stex_structures_name_str 6085,
    6102, 6107, 6109, 6116, 6117, 6122,
    6123, 6128, 6129, 6132, 6148, 6154
\l__stex_structures_new_extstruct_-
    name: ..... 6228, 6286
\l__stex_structures_present: ...
    ..... 6448, 6573
\l__stex_structures_present:nn ...
    ..... 6452, 6458, 6463, 6470, 6473
\l__stex_structures_present_-
    entry:nn ..... 6478, 6484, 6499
\l__stex_structures_present_i:w ..
    ..... 6444, 6450, 6456
\l__stex_structures_present_ii:nw
    ..... 6461, 6469
\l__stex_structures_prop .....
    6488, 6600, 6608, 6640, 6648, 6665,
    6668, 6674, 6695, 6697, 6709, 6711
\l__stex_structures_prop_do_-
    decls: ..... 6652, 6683
\l__stex_structures_prop_do_-
    notations: ..... 6653, 6732
\l__stex_structures_redo_tl ...
    .. 6627, 6651, 6677, 6724, 6735, 6750
\l__stex_structures_replace_-
    this_tl ..... 6140
\l__stex_structures_seq .....
    ..... 6649, 6696, 6710, 6734
\l__stex_structures_set_comp_tl .
    ..... 6294, 6343, 6347, 6505, 6622
\l__stex_structures_set_custom_-
    comp:n ..... 6348, 6537
\l__stex_structures_set_customcomp:
    ..... 6321, 6346
\l__stex_structures_set_this:n ...
    ..... 6570, 6579
\l__stex_structures_set_thiscomp:
    ..... 6294, 6531
\l__stex_structures_set_thisnotation:
    ..... 6325, 6341
\l__stex_structures_shift_-
    argls:nn ..... 6155, 6160
\l__stex_structures_this_tl ...
    6306, 6506, 6507, 6510, 6525, 6582,
    6585, 6592, 6613, 6614, 6617, 6631
\l__stex_symdecl_add_decl: 3751, 3757
\l__stex_symdecl_args_tl . 3804, 3806
\l__stex_symdecl_cs .....
    .. 3802, 3806, 3996, 3998, 4000, 4026
\l__stex_symdecl_do_args: . 3821, 3880
\l__stex_symdecl_env_str .....
    ..... 3659, 3660, 3661
\l__stex_symdecl_get_from_one_-
    string:n ..... 4035, 4080
\l__stex_symdecl_get_symbol_from_-
    cs: ..... 4002, 4013
\l__stex_symdecl_get_symbol_from_-
    modules:nn ..... 4037, 4069
\l__stex_symdecl_get_symbol_from_-
    string:n ... 4003, 4005, 4008, 4030
\l__stex_symdecl_name ... 4033, 4039
\l__stex_symdecl_parse_arity: ...
    ..... 3820, 3826
\l__stex_symdecl_seq .....
    ..... 4032, 4033, 4034, 4038

```

```

\__stex_symdecl_set_textsymdecl_-          \stexstyleextstructure ..... 101
   macro:nnn ..... 3953, 3971 \stexstyleimportmodule ..... 101, 130
\__stex_symdecl_sym_from_str_-           \stexstyleinterpretmod ..... 101
   i:nnnn ..... 4043, 4074 \stexstyleinterpretmodule ..... 101
\__stex_symdecl_sym_i_finish:nnnnnnN     \stexstylemathstructure ..... 101
   ..... 4049, 4056 \stexstylemcb ..... 8833
\__stex_symdecl_sym_i_gobble:nnnnnn      \stexstyleMMTinclude ..... 101
   ..... 4051, 4054 \stexstylemodule ..... 101, 131
\__stex_vars_add: ..... 4765, 4793 \stexstylenotation ..... 101
\l__stex_vars_args_tl ..... 4853, 4855 \stexstyleparagraph ..... 101
\l__stex_vars_bind_bool ..... 4748, 4751, 4753, 4837 \stexstyleproof ..... 101
\l__stex_vars_check_var:nnnnnnnnN       \stexstylerealization ..... 101
   ..... 4903, 4907 \stexstylerealize ..... 101
\l__stex_vars_cs ..... 4851, 4855 \stexstylerenamedecl ..... 101
\__stex_vars_get_var:n ..... 4901, 4919, 4927 \stexstylerequiremodule ..... 101
\__stex_vars_html: ..... 4767, 4820 \stexstylescb ..... 8919
\__stex_vars_macro: ..... 4766, 4807 \stexstylespfsketch ..... 101
\__stex_vars_set_vars:nnnnnn ..... 4888, 4909, 4912 \stexstylesubproof ..... 101
\sTeX/ComputerScience/Software ..... 15 \stexstylesymdecl ..... 101
stex_annotation_env (env.) ..... 138 \stexstylesymdef ..... 101
\stexcommentfont ..... 7211, 7281, 7348, 7370, 7433, 7454 \stexstyletextsymdecl ..... 101
\stexdoctitle ..... 1242, 1301, 1313, 2519, 8043, 8183, 8389, 8478 \stexstyleusemodule ..... 101
\STEXexport ..... 49, 83, 84, 118, 2568 \stexstylevardef ..... 101
\stexhtmlfalse ..... 325 \stexstylevarnotation ..... 101
\stexhtmltrue ..... 322 \stexstylevarseq ..... 101
\STEXInternalAssocArgMarkerI ..... 124 \stexsolutions ..... 108, 8591
\STEXInternalAssocArgMarkerII ..... 124
\STEXInternalNotation .. 4225, 4548, 4568
\STEXInternalSetSrefSymURL ..... 1501, 1802, 1805, 1808
\STEXInternalSrefRestoreTarget ..... 1499, 1525, 1725
\STEXInternalSymbolAfterInvokationTL ..... 124, 128
\STEXInternalTermMathArgiii ..... 124
\STEXInternalTermMathAssocArgiiii ..... 124
\STEXInternalTermMathOMAiii ..... 124
\STEXInternalTermMathOMBiii ..... 124
\STEXInternalTermMathOMSOrOMViii .. 124
\STEXinvisibile .. 78, 4322, 5186, 7261, 7316
\STEXRestoreNotsEnd .... 2420, 2444, 2451
\stexstyle ..... 130
\stexstyleassertion ..... 101
\stexstyleassign ..... 101
\stexstyleassignMorphism ..... 101
\stexstylecopymod ..... 101
\stexstylecopymodule ..... 101
\stexstyledefinition ..... 101
\stexstyleexample ..... 101
\stexstyleextstructure ..... 101
\stexstyleimportmodule ..... 101, 130
\stexstyleinterpretmod ..... 101
\stexstyleinterpretmodule ..... 101
\stexstylemathstructure ..... 101
\stexstylemcb ..... 8833
\stexstyleMMTinclude ..... 101
\stexstylemodule ..... 101, 131
\stexstylenotation ..... 101
\stexstyleparagraph ..... 101
\stexstyleproof ..... 101
\stexstylerealization ..... 101
\stexstylerealize ..... 101
\stexstylerenamedecl ..... 101
\stexstylerequiremodule ..... 101
\stexstylescb ..... 8919
\stexstylespfsketch ..... 101
\stexstylesubproof ..... 101
\stexstylesymdecl ..... 101
\stexstylesymdef ..... 101
\stexstyletextsymdecl ..... 101
\stexstyleusemodule ..... 101
\stexstylevardef ..... 101
\stexstylevarnotation ..... 101
\stexstylevarseq ..... 101
\stexsolutions ..... 108, 8591
str commands:
\c_backslash_str ..... 696
\c_colon_str ..... 827, 882, 1177, 2363
\c_dollar_str ..... 4314
\c_hash_str ..... 595, 2625, 4321
\c_percent_str ..... 54, 1044
\str_case:Nn ..... 1181
\str_case:nn ..... 4647, 5435
\str_case:nnTF .. 1412, 3830, 4388,
   4410, 5463, 5491, 7683, 7704, 7822
\str_clear:N ... 386, 387, 394, 412,
   695, 902, 1062, 1137, 1585, 1903,
   2069, 2179, 2361, 2495, 2803, 2998,
   3086, 3113, 3128, 3413, 3544, 3545,
   3687, 3688, 3689, 3690, 3700, 3701,
   3723, 3903, 3942, 3943, 3993, 3994,
   4129, 4130, 4131, 4132, 4891, 4918,
   4926, 5523, 6102, 6170, 6787, 6788,
   6790, 6837, 7015, 7068, 7127, 7376,
   7607, 7678, 7699, 7951, 8222, 8223,
   8325, 8326, 8512, 8529, 8754, 8755
\str_const:Nn ..... 7741
\str_count:n ..... 536, 541
\str_gset:Nn ..... 1502, 1815, 1823
\str_gset_eq:NN ..... 1057, 2869, 2870
\str_if_empty:NTF ..... 89, 433, 439, 606, 612, 702, 722,
   909, 1050, 1052, 1071, 1129, 1202,

```

1590, 1596, 1602, 1649, 1650, 1665,
 1676, 1696, 1719, 1730, 1738, 1758,
 1783, 1849, 2305, 2548, 2785, 2808,
 3002, 3006, 3088, 3250, 3426, 3559,
 3573, 3660, 3746, 3780, 3783, 3786,
 3789, 3916, 3987, 4141, 4240, 4252,
 4261, 4308, 4759, 4825, 4828, 4831,
 4834, 4920, 4928, 4952, 4955, 5022,
 5025, 5028, 5031, 6023, 6106, 6116,
 6165, 6172, 6602, 6815, 6816, 6821,
 6838, 6858, 6926, 7021, 7074, 7321,
 7330, 7399, 7413, 7419, 7644, 7680,
 7701, 7812, 7871, 8034, 8378, 8462,
 8467, 8538, 8582, 8602, 8651, 8702,
 8744, 8763, 8777, 8780, 8787, 8790
`\str_if_empty:nTF` 472, 499,
 683, 734, 829, 1509, 1863, 2498, 3399
`\str_if_eq:NNTF` 172, 783
`\str_if_eq:nnTF`
 146, 162, 163, 164, 186,
 302, 535, 540, 570, 586, 595, 607,
 613, 697, 735, 736, 752, 1633, 1717,
 1759, 1764, 1766, 1820, 2285, 2367,
 2606, 2639, 2744, 3015, 3020, 3029,
 3032, 3208, 3234, 3418, 3434, 3661,
 4243, 4255, 4267, 4908, 4911, 5102,
 7038, 7875, 7878, 7943, 8102, 8839
`\str_if_eq_p:nn` 4045, 4046, 4085, 4086
`\str_if_exist:NTF` 233, 1846
`\str_if_in:NnTF` 4321, 6004, 6970
`\str_item:Nn` 697, 752, 3885
`\str_lowercase:n` 8839
`\str_map_break:` 3831
`\str_map_break:n` 3832, 3836, 3840,
 3844, 3848, 3852, 3856, 3860, 3864
`\str_map_inline:Nn` 3829
`\str_new:N`
 135, 1564, 1565, 2035, 2290, 3719
`\str_put_right:Nn` 7134
`\str_range:Nnn` 2029
`\str_range:nnn` 536, 541, 575
`\str_replace_all:Nnn` 696
`\str_set:Nn`
 48, 56, 137, 185, 228, 690, 691,
 694, 710, 827, 1084, 1167, 1179,
 1180, 1511, 1513, 1515, 1579, 1580,
 1608, 1619, 1627, 1634, 1698, 1704,
 1709, 1715, 1724, 1865, 1872, 1877,
 1988, 1998, 2027, 2065, 2324, 2340,
 2363, 2433, 2662, 2816, 2821, 2824,
 2835, 2858, 2859, 3042, 3043, 3044,
 3077, 3082, 3087, 3092, 3098, 3106,
 3108, 3114, 3118, 3119, 3120, 3129,
 3133, 3134, 3135, 3142, 3149, 3152,
 3155, 3161, 3168, 3171, 3174, 3192,
 3219, 3264, 3462, 3555, 3561, 3565,
 3695, 3725, 3747, 3834, 3838, 3842,
 3846, 3850, 3854, 3858, 3862, 3866,
 3915, 3917, 3919, 3921, 4017, 4018,
 4058, 4059, 4090, 4091, 4142, 4174,
 4203, 4208, 4327, 4367, 4378, 4504,
 4758, 4760, 4892, 4936, 4938, 4951,
 4953, 4956, 5211, 5212, 5625, 5669,
 5710, 6002, 6005, 6084, 6107, 6117,
 6126, 6220, 6284, 6288, 6290, 6598,
 6603, 6827, 6841, 6847, 6968, 6971,
 7018, 7071, 7323, 7332, 7421, 7643,
 7645, 8540, 8604, 8653, 8704, 8765
`\str_set_eq:NN`
 1068, 1570, 1571, 2535, 2806, 2836,
 3944, 3945, 4144, 4173, 4464, 4474,
 4493, 4507, 4508, 4782, 4977, 5966,
 5979, 5992, 6109, 6174, 6817, 6826,
 6876, 6928, 7307, 7388, 7606, 7610,
 7634, 7636, 7649, 7663, 7665, 7666
`\str_uppercase:n` 7943
`\l_tmpa_str` 159, 160, 161,
 162, 163, 164, 165, 169, 185, 189,
 190, 191, 193, 1903, 1904, 1905,
 1910, 1918, 2821, 2824, 2829, 2836
`\subparagraph` 27, 79
`subproblem (env.)` 8449
`subproof (env.)` 7307
`\subproof` 7265, 7366
`\subproofautorefname` 7307
`\subsection` 26, 27, 79
`\subsubsection` 27, 79
`\svar` 92, 3773, 4933
`\symbol` 86
`\symdecl` 23,
 24, 31, 32, 39, 40, 47, 84–86, 96,
 101, 121, 126, 132, 139, 140, 2984,
 3719, 7507, 7509, 7538, 7546, 7549
`\symdef` 32, 33, 35, 40, 41, 85,
 92, 121, 126, 132, 2986, 4484, 7491,
 7494, 7500, 7502, 7504, 7506, 7516,
 7517, 7518, 7519, 7525, 7533, 7544,
 7553, 7554, 7557, 7560, 7562, 7564
`\Symname` 86, 97, 5908
`\symname` 19, 20, 24, 62, 86, 87, 97, 99, 5908
`\symref` 19, 20, 24, 48, 85, 86, 90, 97, 99, 5908
`\symrefemph` 99, 100, 6020
`\symuse` . 48, 86, 5157, 5236, 5713, 6394,
 6401, 6412, 6494, 6559, 6629, 6630
`sys commands:`
`\sys_get_shell:nnN` 45
`\sys_if_platform_windows:TF`
 51, 688, 747, 974, 1043

T

\target	131, 134–136	\Gin@ewidth	8136, 8137, 9240, 9250, 9254
\test	71, 112	\Gin@exclamation 9240, 9241, 9249
test	107	\Gin@mrepos 2038, 2041, 2045, 2066, 2070, 2075
\testbigspace	9044	\hwexam@bonuspts 9202
\testemptypage	70, 112	\hwexam@checktime 9196
\testemptypage	9038	\hwexam@duration	9172, 9175, 9181, 9186
testheading (env.)	71, 113	\hwexam@kw@due 9123
\testheading	9169	\hwexam@kw@forgrading 9164
\testmedspace	9043	\hwexam@kw@given 9120
\testnewpage	70, 112	\hwexam@kw@grade 9165
\testnewpage	9045	\hwexam@kw@probs 9165
\testsmsllspace	70, 70, 70, 112, 112, 112	\hwexam@kw@pts 9166
\testsmsllspace	9042	\hwexam@kw@reached 9167
\testspace	70, 112	\hwexam@kw@sum 9165
\testspace	8561, 9041	\hwexam@kw@testemptypage 9039
\TeX	23, 24	\hwexam@min	... 9173, 9180, 9185, 9196
\tex	23, 24	\hwexam@minutes@kw 9173
T _E X and L ^A T _E X 2 _E commands:		\hwexam@reqpts	9182, 9187, 9199, 9203
\@	2084, 2087, 2091	\hwexam@tools 9183, 9188
\@addtoreset	8341	\hwexam@totalmin 9195, 9196
\@arabic	8194	\hwexam@totalpts 9194, 9203
\@author	8173	\if@bonuspoints 9198
\@auxout	1795, 7728, 8443	\if@crustex 306
\@bonuspointsfalse	9200	\itemize@inner 7983, 7989, 7998
\@bonuspointstrue	9205	\itemize@label 7987, 7990, 7993
\@currentHref	7323, 7332, 7421	\itemize@level 7981, 7986, 7989, 7998
\@currentcounter	1528	\itemize@outer 7982, 7986
\@currentlabel	1529, 7322, 7323, 7331, 7332, 7420, 7421, 8033	\lstd@mrepos 2052, 2055, 2059
\@currentlabelname	1531, 1532	\ltx@ifpackageloaded	141, 145, 2036, 2050, 2063, 7201, 8304, 8859, 9063
\@currenvir	7917, 7918	\m@switch 5879
\@dblarg	8166, 8175	\mdf@patchamsthdm 8007
\@gobble	361	\metakeys@show@keys 7984
\@ifnextchar	360	\notesslides@slidelabel 8010, 8025
\@ifstar	8145	\ns@author 8166, 8167
\@mainmatterfalse	1457, 1467	\ns@title 8175, 8176
\@mainmattertrue	1478, 1489	\pgf@temp 2081, 2082, 2083
\@notprerr	140, 1422	\pgfkeys@spdef 2081
\@onlypreamble	140, 1422	\pgfutil@empty 2083
\@rustexfalse	298	\pgfutil@InputIfFileExists 2090
\@title	1271, 1272, 8182	\prematrestop@endsfragment 7916, 7919, 7925
\activate@excursion	8206, 8211	\problem@kw@* 8301
\bb@loaded	8305, 9064	\problem@kw@correct 8867, 8929
\beamer@shortauthor	8169, 8171, 8186	\problem@kw@grading 8744
\beamer@shorttitle	8178, 8180, 8189	\problem@kw@hint 8642
\c@chapter	7898	\problem@kw@minutes 8424, 8506, 8803
\c@framenumber	8194	\problem@kw@note 8691
\c@part	7910	\problem@kw@pts 8419, 8501, 8798
\compemph@uri	100, 128, 4364, 6012, 6020	\problem@kw@solution 8581
\correction@table	9148	\problem@kw@wrong 8869, 8931
\defemph@uri	100, 6020	\problem@restore 8444, 8448, 9142
\define@key	2037, 2051, 2064	\stex@backend 137, 296
\Gin@eheight	... 9241, 9244, 9249, 9254		

\symrefemph@uri	132
... 99, 100, 5924, 5934, 5945, 6020	
\varemph@uri	24
... 100, 128, 5965, 5978, 5991, 6020	
\texname	24
\text	20
\textbf 19, 6062, 7803, 8435, 9023, 9113, 9118	
\textcolor	9019
\textsymdecl ... 23, 24, 85, 126, 2985, 3901	
\textwarning	105
\textwidth	8134, 8156, 9161
\the 251, 252, 254, 256, 258, 2084, 2085, 2086	
\theassignment	9106, 9113
\thechapter ... 1332, 1373, 1401, 1436, 7829, 7834, 7838, 7842, 7846, 7850	
\theframenumber	8033
\theparagraph	7846, 7850
\thepart ... 1328, 1370, 1397, 1430, 7824	
\theplainsproblem	8345, 8346
\thesection	7834, 7838, 7842, 7846, 7850, 8346
\thesproblem . 8342, 8346, 8435, 8444, 9106	
\thes subparagraph	7850
\thes subsection ... 7838, 7842, 7846, 7850	
\thes subsection	7842, 7846, 7850
\this	57–59, 93, 94, 5208, 6081, 6510, 6587, 6588, 6592, 6617
\thisarchive	3290, 3331
\thisargs	3735, 4491
\thiscopyname	3482, 3502, 3526, 3599, 3619, 3640
\thisdeclname ... 3732, 3960, 4174, 4488	
\thisdecluri	3731, 3959, 4175, 4178, 4487, 4495
\thisdefiniens	3734, 4490
\thisfor	6877
\thismodulename 130, 2536, 3067, 3292, 3333	
\thismoduleuri	130, 2535, 3066, 3291, 3332, 3483, 3503, 3527, 3600, 3620, 3641
\thisname	6876
\thisnotation 4176, 4475, 4494, 4783, 4978	
\thisnotationvariant	4173, 4474, 4493, 4782, 4977
\ThisStyle	5879
\thisstyle	130, 459, 462, 463, 464, 510, 513, 514, 515, 516, 524, 527, 528, 3068, 3293, 3334, 3736, 3961, 4473, 4492, 4781, 4976
\thistitle	100, 130, 2517, 2518, 2519, 6875, 8393, 8436, 8437
\thistype	3733, 4489
\thisvarname	4472, 4476, 4780, 4784, 4975, 4979
\throwaway	132
\tikzinput ... 115, 2075, 9233, 9238, 9264	
tikzinput commands:	
\c_tikzinput_image_bool 34, 9222, 9229	
\tiny	8011, 8025
\title	71, 113
\title	8175
tl commands:	
\tl_clear:N	400, 459,
510, 771, 1344, 1723, 2174, 2293, 2499, 2763, 3068, 3293, 3334, 3546, 3702, 3703, 3704, 3736, 3881, 3904, 3905, 3941, 3961, 4195, 4204, 4300, 4443, 4473, 4492, 4531, 4726, 4781, 4941, 4964, 4976, 5206, 5216, 5313, 5326, 5524, 5911, 5912, 5913, 6010, 6101, 6582, 6610, 6651, 6791, 6792, 6793, 7094, 7095, 7097, 7098, 7099, 7100, 7377, 7378, 8221, 8846, 8848, 8849, 8974, 9081, 9082, 9083, 9153, 9154, 9155, 9180, 9181, 9182, 9183	
\tl_const:Nn	5851, 5852
\tl_count:N	5468, 5474
\tl_count:n	5599, 5610
\tl_gclear:N	2319
\tl_gput_left:Nn	372, 374, 375
\tl_gput_right:Nn	252, 2120, 2124, 2143, 2296, 2560, 3301, 8207
\tl_gset:Nn	254, 367, 368, 1023, 1243, 1250, 2144, 2148, 4211, 5169, 5173, 8484, 8487, 9145, 9146
\tl_gset_eq:NN	46, 2795
\tl_head:N .. 886, 940, 4000, 6532, 6538	
\tl_head:n	554, 595, 598, 4677, 5600, 5611
\tl_if_empty:NTF	
... 524, 787, 1025, 1261, 1271, 1314, 1750, 2382, 2518, 2528, 3351, 3576, 3763, 3794, 3797, 3800, 3929, 4104, 4117, 4189, 4198, 4206, 4376, 4530, 4843, 4846, 4849, 4857, 4879, 4962, 5036, 5039, 5042, 5050, 5280, 5292, 5383, 5403, 5488, 5627, 5740, 5753, 5795, 5824, 6083, 7207, 7227, 7230, 7618, 8137, 8228, 8388, 8436, 8477, 8581, 8642, 8691, 8867, 8869, 8871, 8929, 8931, 8933, 9020, 9101, 9114, 9119, 9122, 9172, 9199	
\tl_if_empty:nTF	369, 553, 569, 585, 594, 751, 760, 882, 883, 945, 952, 1099, 1569, 1727, 1749, 1907, 1929, 1960, 2018, 2109, 2216, 2582, 2597, 2649, 2709, 2734, 2804, 2819, 2918, 2977, 3078, 3112, 3127,

3141, 3160, 3191, 3218, 3407, 3998,
 4111, 4124, 4578, 4628, 4935, 5431,
 5452, 5528, 5551, 5954, 6155, 6279,
 6328, 6331, 6376, 6386, 6475, 6477,
 6571, 6580, 6675, 6685, 6749, 6894,
 7016, 7069, 7085, 7581, 8168, 8177
`\tl_if_empty_p:N` 777
`\tl_if_eq:NNTF` 1141, 6173, 6532, 6538, 8991
`\tl_if_eq:NnTF` 8254, 8398, 8401, 8418, 8423
`\tl_if_eq:nnTF` 2451, 4702, 5096, 5611, 5656, 5694
`\tl_if_exist:NTF` 210, 251, 266, 305, 463, 515, 527, 1287,
 1296, 1531, 1691, 2552, 6240, 7855
`\tl_if_in:NnTF` 2231, 2249, 2253
`\tl_item:nn` 5602
`\tl_map_inline:Nn` 2184
`\tl_map_inline:nn` 217, 221
`\tl_new:N` 965, 1035, 1241, 1566, 2116, 2117,
 2140, 2151, 2469, 4747, 5162, 5166
`\tl_put_left:Nn` 4617, 4618, 5357, 8368, 9094
`\tl_put_right:Nn` 1217, 2370, 2745, 2749, 3883, 3884,
 4306, 4312, 4322, 4538, 4543, 4546,
 4649, 4657, 4662, 4670, 4733, 5194,
 5218, 5228, 5338, 6581, 6677, 6724,
 6735, 6750, 8995, 9157, 9158, 9159
`\tl_range:nmn` 571, 587
`\tl_set:Nn` 211, 248, 351, 454,
 473, 475, 494, 495, 500, 501, 503,
 504, 750, 784, 864, 958, 1022, 1279,
 1320, 1455, 1465, 1581, 1747, 2038,
 2041, 2536, 2645, 2647, 2650, 2865,
 3046, 3047, 3048, 3049, 3050, 3290,
 3331, 3482, 3502, 3526, 3556, 3562,
 3566, 3599, 3619, 3640, 3731, 3804,
 3959, 4020, 4021, 4022, 4023, 4024,
 4061, 4062, 4063, 4064, 4065, 4093,
 4094, 4095, 4096, 4097, 4175, 4190,
 4199, 4224, 4241, 4244, 4253, 4256,
 4262, 4269, 4288, 4487, 4535, 4548,
 4566, 4587, 4602, 4603, 4615, 4654,
 4667, 4686, 4699, 4725, 4808, 4853,
 4894, 4895, 4896, 4897, 4898, 5003,
 5046, 5130, 5163, 5168, 5170, 5177,
 5207, 5213, 5214, 5215, 5327, 5348,
 5468, 5474, 5487, 5493, 5496, 5499,
 5542, 5628, 5660, 5662, 5712, 5855,
 5856, 5900, 5901, 6140, 6152, 6221,
 6294, 6306, 6342, 6343, 6347, 6350,
 6357, 6393, 6432, 6506, 6512, 6522,
 6585, 6587, 6612, 6613, 6619, 6628,
 6676, 6678, 6720, 6725, 6737, 6740,
 6744, 6747, 6752, 6755, 6759, 6762,
 7200, 7206, 7824, 7825, 7829, 7830,
 7834, 7835, 7838, 7839, 7842, 7843,
 7846, 7847, 7850, 7851, 7866, 8323,
 8324, 8434, 8459, 8460, 8520, 8809,
 8858, 8865, 8890, 8896, 8927, 8985,
 8986, 9133, 9139, 9140, 9196, 9202
`\tl_set_eq:NN` 350, 371, 373,
 378, 1572, 2517, 3066, 3067, 3291,
 3292, 3332, 3333, 3483, 3503, 3527,
 3600, 3620, 3641, 3732, 3733, 3734,
 3735, 3960, 4178, 4246, 4258, 4282,
 4430, 4438, 4472, 4488, 4489, 4490,
 4491, 4534, 4771, 4780, 4871, 4876,
 4880, 4960, 4975, 5407, 5547, 5588,
 5667, 6438, 6510, 6592, 6617, 6875,
 7096, 8393, 8396, 8397, 9194, 9195
`\tl_set_rescan:Nnn` 139, 190
`\tl_tail:n` 554, 555, 600, 2216, 5573, 6402
`\tl_to_str:n` 66, 69, 97, 100, 110,
 122, 218, 222, 223, 247, 350, 572,
 588, 684, 706, 713, 812, 830, 832,
 835, 854, 860, 911, 981, 1073, 1177,
 1249, 1264, 1502, 1789, 1814, 1815,
 1843, 1844, 1850, 1899, 1947, 2107,
 2128, 2144, 2147, 2148, 2185, 2188,
 2230, 2232, 2248, 2250, 2254, 2272,
 2281, 2425, 2427, 2428, 2431, 2432,
 2462, 2463, 2521, 2589, 2590, 2591,
 2613, 2846, 3000, 3075, 3350, 3359,
 3548, 4285, 4586, 4600, 4679, 4724,
 5513, 6150, 6183, 6233, 6300, 6320,
 6324, 6483, 6611, 6666, 6696, 6710,
 6734, 6772, 6808, 7725, 8305, 9064
`\tl_trim_spaces:N` 49
`\l_tmpa_tl` 45, 46, 48,
 4316, 4726, 4733, 4740, 5157, 5455,
 5461, 5466, 5468, 5472, 5474, 5477,
 5486, 5488, 5493, 5496, 5499, 5714,
 6390, 8865, 8881, 8884, 8927, 8943,
 8946, 8985, 8991, 9153, 9157, 9165
`\l_tmpb_tl` 5455, 5461, 5463, 5477, 5487,
 5491, 8986, 8991, 9154, 9158, 9166
tmpc commands:
`\l_tmpc_tl` 9155, 9159, 9167
`\to` 7526, 7529, 7539, 7540
`\today` 8196
`\TODO` 5015, 5016, 6377, 8252

todo internal commands:	
\l__todo_mmt_module_str	7606, 7611, 7624, 7627, 7630, 7634, 7649, 7665
__todo_newlabel:n	7724, 7731
\l__todo_old_metagroup_cd	7650, 7662
__todo_old_newlabel:	7725, 7730
\l__todo_stex_module_str	7610, 7611, 7636, 7663
token commands:	
\c_math_subscript_token	49, 84, 4412, 4413, 4416, 4417, 4421, 6522, 7531, 7542, 7544
\topsep	7216, 8494, 8770, 8821, 8907
\trueFfalseT	8968
\trueTfalseF	8963
\type	71, 113
U	
\undefined	131, 41
\univ	63
\unless	7917
\uppercase	5948
\uri	135, 136
use commands:	
\use:N	249, 382, 460, 464, 466, 511, 516, 518, 525, 528, 530, 1315, 1317, 1726, 1749, 1847, 1852, 2185, 2188, 2663, 3931, 5072, 5282, 5308, 6440, 7856, 7862
\use:n	1835, 2028, 2036, 2050, 2063, 3801, 4850, 5043, 5238, 5328, 5360, 6492, 6599
\use:nn	55, 198, 201, 207, 547, 1317, 2460, 2484, 2815, 2860, 3016, 3021, 3758, 3925, 4179, 4330, 4369, 4477, 4496, 4785, 4980, 5127, 5149, 5154, 5370, 5573, 5620, 5705, 5880, 6121, 6153, 6558, 6625, 6667, 6829, 7087, 7415, 7652, 8232, 8522, 9135
\use_i:nn	6599, 9158
\use_ii:nn	1839, 2937, 2952, 6639
\use_none:nnnnnn	3035
\usebox	8088
\usemodule	15, 18, 22, 23, 26, 37, 49, 90, 91, 93, 209, 423, 430, 3054
\usepackage	7, 21, 2028, 7931
\useSGvar	106, 8245
\usestructure	93, 6768
\usetHEME	7931
\usetikzlibrary	77, 2106, 9232
V	
\varbind	47, 52, 92, 97, 6922, 6945, 7056, 7064
\vardef	35, 47, 52, 92, 121, 122, 4747, 7087, 7415
\varemph	100, 6020
\Varname	5908
\varname	5908
\varnotation	92, 4461
\varref	5908
\varseq	43, 92, 4949
\vbox	132, 7202, 8005, 8094, 8109, 8562, 8623, 8672, 8724, 8860
vbox commands:	
\vbox_set:Nn	2153
\vdash	7554
\fill	7860, 9039
\VM	54
\VMs	55
\vn	43
\vp	45, 47
\vrule	7202, 8156, 8860
\vskip	7202, 8155, 8157, 8860
\vspace	9041
W	
\wff	20
\withbrackets	89, 5898, 5906
X	
\xspace	1287, 1296, 3975, 3976, 3982, 3983
Y	
\yesFnoT	8958
\yesTnoF	8953
\yield	7270, 7461, 7464