

The sTeX4 Package Collection*

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

2026-06-25

Contents

1	Introduction & Setup	8
1.1	What is sTeX?	8
1.2	The sTeX package	9
1.3	Math archives and the MathHub Directory	9
1.4	Setting Up the FvMf 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	17
2.3	Modules and Simple Symbol Declarations	20
2.4	Documenting Symbols	23
2.5	Sectioning and Reusing Document Fragments	25
2.6	Building and Exporting HTML	27
3	Mathematical Concepts	30
3.1	Simple Symbol Declarations	30
3.1.1	Semantic Macros and Notations	30
3.1.2	Types and Variables	33
3.1.3	Flexary Macros and Argument Modes	35
3.1.4	Precedences	37
3.1.5	Finishing Equality	38
3.1.6	Variable Sequences	38
3.2	Statements	38
3.2.1	Definitions	39
	Semantic Macros in Text Mode	40
	Definientia	41

*Version 4.0.1 (last revised 2026-06-25)

	Using <code>Symbols</code> Without <code>Semantic Macros</code> and Exporting Code in <code>Modules</code>	42
3.2.2	Assertions	43
3.2.3	Proofs	45
3.3	Mathematical Structures	45
3.3.1	Declaring and Using <code>Structures</code>	45
	Instantiating <code>Structures</code>	46
3.3.2	Extending <code>Structures</code> and Axioms	48
	Conservative Extensions	49
3.3.3	Nesting <code>Structures</code> and <code>\this</code>	50
3.4	Complex Inheritance and Theory Morphisms	52
3.4.1	Glueing <code>Structures</code> Together	55
3.4.2	Realizations	56
4	Extensions for Education	59
4.1	Slides and Course Notes	59
4.1.1	Introduction	59
4.1.2	Package Options	60
4.1.3	Notes and Slides	60
4.1.4	Managing and Customizing Themes	61
4.1.5	Frame Images	62
4.1.6	Ending Documents Prematurely	63
4.1.7	Global Document Variables	63
4.1.8	Excursions	63
4.2	Problems and Exercises	64
4.2.1	Background	64
4.2.2	The Package, Options, and Configuration	65
4.2.3	(Open) Problems	66
4.2.4	Solutions and Answer Classes	67
4.2.5	Grading Support	68
4.2.6	Structured Problems	69
4.2.7	Single/Multiple Choice Blocks	70
4.2.8	Filling-In Concrete Solutions	73
4.2.9	Including Problems	75
4.2.10	Testing and Spacing	75
4.3	Homework Assignments and Exams	76
4.3.1	Introduction	76
4.3.2	Package Options	76
4.3.3	Assignments	76
4.3.4	Including Assignments	76
4.3.5	Typesetting Exams	77
II	User Manual	78

5	Basics	79
5.1	Package and Class Options	79
5.2	Math Archives and the MathHub Directory	80
5.2.1	The Structure of Math Archives	81
5.2.2	MANIFEST.MF-Files	81
5.3	The <code>lib</code> -Directory	82
5.4	Basic Macros	83
6	Document Features	84
6.1	Document Fragments	84
6.2	Using and Referencing Document Fragments	85
6.3	Cross-Document References	85
7	Modules and Symbols	88
7.1	Modules	88
7.1.1	Signature Modules , Languages, and Multilinguality	89
7.2	Symbol Declarations	89
7.2.1	Returns	90
7.3	Referencing Symbols	90
7.4	Notations and Semantic Macros	92
7.4.1	Precedences and Bracketing	93
7.4.2	Notations for Argument Sequences	94
7.4.3	Semantic Macros	94
7.5	Simple Inheritance	95
7.6	Variables and Sequences	97
7.7	Structures	98
7.7.1	Semantic Macros for Structures	99
8	Statements	101
8.1	More on Definitions	102
8.2	More on Assertions	103
9	Customizing Typesetting	104
9.1	Highlighting Symbol References	104
9.2	Styling Environments and Macros	105
9.3	Custom CSS for Environments	107
10	Additional Packages	108
10.1	NotesSlides Manual	108
10.1.1	Introduction	108
10.1.2	Package Options	108
10.1.3	Notes and Slides	109
10.1.4	Customizing Header and Footer Lines	110
10.1.5	Frame Images	110
10.1.6	Ending Documents Prematurely	111
10.1.7	Global Document Variables	111
10.1.8	Excursions	112
10.2	Problem Manual	113
10.2.1	Introduction	113
10.2.2	Problems and Solutions	113

10.2.3	Markup for Added-Value Services	115
	Multiple Choice Blocks	115
	Filling-In Concrete Solutions	116
10.2.4	Including Problems	117
10.2.5	Testing and Spacing	118
10.3	HWExam Manual	118
10.3.1	Introduction	118
10.3.2	Package Options	118
10.3.3	Assignments	119
10.3.4	Including Assignments	119
10.3.5	Typesetting Exams	119
10.4	Tikzinput Manual	120
III	Documentation	122
11	Utilities	123
11.1	kpsewhich and Environment Variables	123
11.2	Logging	124
11.3	File Paths	124
11.4	Group-like Behaviours	125
11.5	Key Handling	126
11.6	Languages	127
11.7	Styling	127
11.8	Persisting Dependencies	127
12	HTML Output	128
13	Math Archives	132
14	URIs	134
14.1	Document URIs	134
14.2	Module URIs	135
14.3	Symbol URIs	136
15	Documents	138
15.1	Sectioning	139
15.2	Inter-Document References	140
15.3	Inputting From MathHub Resources	140
16	Modules	142
16.1	SMS Mode	143
16.2	Inheritance and Morphisms	145
16.3	Theory Morphisms	145

17 Symbols	147
17.1 Declarations	147
17.2 Retrieval	148
17.3 Variables	148
17.3.1 Variable Sequences	149
17.4 Expressions	149
17.4.1 Term Annotations	150
17.4.2 Symbol Arguments	151
17.4.3 Notation components	151
17.4.4 Symbol References	152
17.5 Checking	152
18 Notations	153
18.1 Automated Bracketing	154
19 Structures	156
20 Statements	157
21 Proofs	158
22 Metatheory	160
23 Others	161
24 Additional Packages	162
24.1 NotesSlides Documentation	162
24.2 Problem Documentation	162
24.3 HWExam Documentation	162
24.4 Tikzinput Documentation	162
IV Implementation	163
25 Setting up	164
26 Utilities	166
26.1 kpsewhich and Environment Variables	167
26.2 Logging	168
26.3 File Paths	169
26.4 Group-like Behaviours	173
26.5 Key Handling	174
26.6 Languages	176
26.7 Styling	177
26.8 Persisting Dependencies	179
26.9 Auxiliary Methods	181
27 HTML Output	182
28 Math Archives	187

29 URIs	194
29.1 Document URIs	194
29.2 Module URIs	197
29.3 Symbol URIs	201
30 Documents	204
30.1 Sectioning	206
30.2 Inter-Document References	211
30.3 Inputting From MathHub Resources	216
31 Modules	222
31.1 SMS Mode	230
31.2 Inheritance and Morphisms	234
31.3 Theory Morphisms	239
32 Symbols	254
32.1 Declarations	254
32.2 Retrieval	262
32.3 Variables	266
32.3.1 Variable Sequences	269
32.4 Expressions	271
32.4.1 Term Annotations	292
32.4.2 Symbol Arguments	294
32.4.3 Notation components	300
32.4.4 Symbol References	302
32.5 Checking	306
33 Notations	308
33.1 Automated Bracketing	323
34 Structures	325
35 Statements	331
36 Proofs	338
37 Metatheory	351
38 Others	354
39 Additional Packages	356
39.1 Implementation: The notesslides Package	356
39.1.1 Class and Package Options	356
39.1.2 Notes and Slides	360
39.1.3 Environment and Macro Patches	364
39.1.4 Styling Across Notes/Slides	365
39.1.5 Beamer Compatibility	366
39.1.6 TODO Excursions	367
39.2 Implementation: The problem Package	369
39.2.1 Package Options	369
39.2.2 Problems and Solutions	371

39.3	Implementation: The hwexam Package	389
39.3.1	Package Options	389
39.3.2	QR Codes	390
39.3.3	Assignments	396
39.3.4	Leftovers	400
39.4	Tikzinput Implementation	401

Index 403



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

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

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

Chapter 1

Introduction & Setup

1.1 What is $\text{\textcolor{teal}{s}TeX}$?

$\text{\textcolor{teal}{s}TeX}$ is a system for generating human-oriented documents in either $\text{\textcolor{teal}{P}DF}$ or $\text{\textcolor{teal}{H}TML}$ format, augmented with computer-actionable semantic information (conceptually) based on the $\text{\textcolor{teal}{O}MDoc}$ format and ontology.

At its core is the $\text{\textcolor{teal}{s}TeX}$ package for $\text{\textcolor{teal}{L}ATeX}$, that allows for semantically marking up document fragments; in particular concepts, $\text{\textcolor{teal}{f}ormulae}$ and $\text{\textcolor{teal}{m}athematical}$ statements (such as definitions, theorems and proofs). Running $\text{\textcolor{teal}{p}dflatex}$ over $\text{\textcolor{teal}{s}TeX}$ -annotated documents formats them into normal-looking $\text{\textcolor{teal}{P}DF}$.

But $\text{\textcolor{teal}{s}TeX}$ also comes with a conversion pipeline into semantically annotated $\text{\textcolor{teal}{H}TML}$ ($\text{\textcolor{teal}{F}TML}$), which can host semantic added-value services that make the documents *active* (i.e. interactive and user-adaptive) – essentially turning $\text{\textcolor{teal}{L}ATeX}$ into a document format for (mathematical) knowledge management (MKM).

The $\text{\textcolor{teal}{s}TeX}$ system consists of the following components:

- The $\text{\textcolor{teal}{s}TeX}$ package,
- the $\text{\textcolor{teal}{R}usTeX}$ system for converting $\text{\textcolor{teal}{L}ATeX}$ documents to (semantically enriched) $\text{\textcolor{teal}{F}TML}$,
- the $\text{\textcolor{teal}{F}LMf}$ system; a semantically informed back-end for managing and hosting the $\text{\textcolor{teal}{F}TML}$ with integrated added-value services,
- a collection of $\text{\textcolor{teal}{m}ath archives}$ of $\text{\textcolor{teal}{s}TeX}$ document fragments and $\text{\textcolor{teal}{s}ymbols}$ for reuse, available of $\text{\textcolor{teal}{m}athhub.info}$, and
- the $\text{\textcolor{teal}{F}LMf IDE}$: A $\text{\textcolor{teal}{V}S Code}$ extension that, besides the usual $\text{\textcolor{teal}{I}DE}$ functionalities like syntax highlighting, integrates $\text{\textcolor{teal}{F}LMf}$ and allows for accessing the public $\text{\textcolor{teal}{m}ath archives}$ on $\text{\textcolor{teal}{m}athhub.info}$ to make the entire toolchain easily accessible to authors.

If $\text{\textcolor{teal}{P}DF}$ is all you are interested in, the $\text{\textcolor{teal}{s}TeX}$ package is all you *need*. Either way, however, we recommend using the $\text{\textcolor{teal}{s}TeX IDE}$ – it very much helps with semantically annotating your $\text{\textcolor{teal}{L}ATeX}$ documents.

1.2 The $\text{\texttt{sTeX}}$ package

The $\text{\texttt{sTeX}}$ package extends $\text{\texttt{L\texttt{A}T\texttt{E}X}}$ 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 $\text{\texttt{sTeX}}$ 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 Math archives and the MathHub Directory

To make the most of $\text{\texttt{sTeX}}$, it is strongly encouraged to follow a workflow of small document fragments and **modules** to maximize reuse.

One considerable weakness of $\text{\texttt{L\texttt{A}T\texttt{E}X}}$ 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.

$\text{\texttt{sTeX}}$ therefore adds an organizational layer on top of $\text{\texttt{L\texttt{A}T\texttt{E}X}}$'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 $\text{\texttt{sTeX}}$ 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 **F\texttt{M}\texttt{J} IDE** do the work for you (see section 1.4).

For more on **math archives**, see Section 1 (**Math Archives** and the **MathHub** Directory) in the $\text{\texttt{sTeX}}$ Manual.

1.4 Setting Up the **F\texttt{M}\texttt{J} IDE**

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

Setting up $\text{\texttt{sTeX}}$ with the dedicated **IDE** is easy:

1. Download and install **VS Code** here: <https://code.visualstudio.com/download>

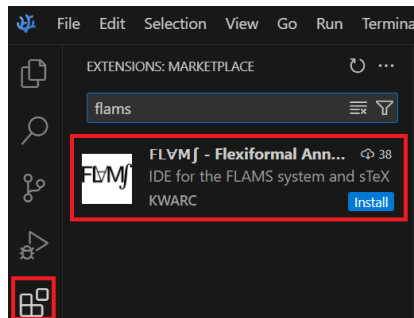


Figure 1: Installing the FLVMj IDE

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 FLVMj extension by *KWARC*, as in Figure 1.
3. Having done so, upon opening any folder in **VS Code** containing a `.tex`-file, you will be prompted to either download a new FLVMj or point the extension to an existing FLVMj executable. In the vast majority of cases, you'll want to download and choose a directory to install FLVMj to.

Part I

Tutorial

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

[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 [FvMf IDE](#), and we consequently assume that you have that one set up, as described in section 1.4.

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 [FLMf](#) 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 [L^AT_EX](#)-version of the above document fragment, without any [sTeX](#) markup, could look like this:

Example 1

Input:

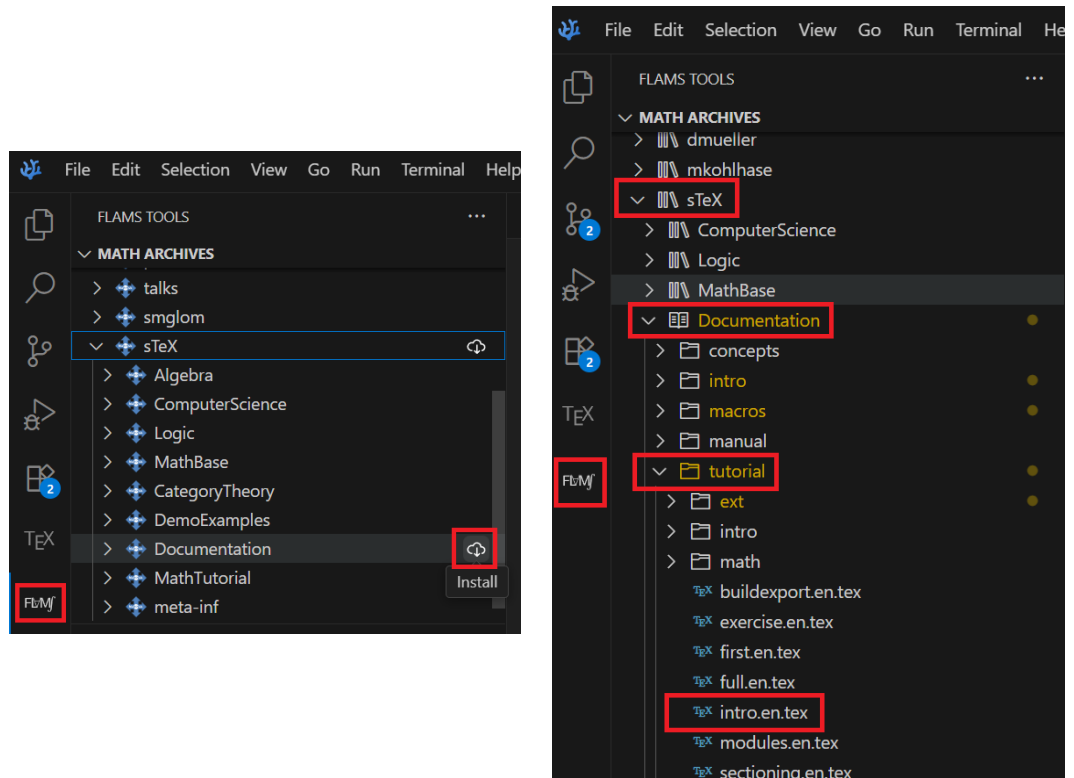


Figure 2: Installing Math Archives

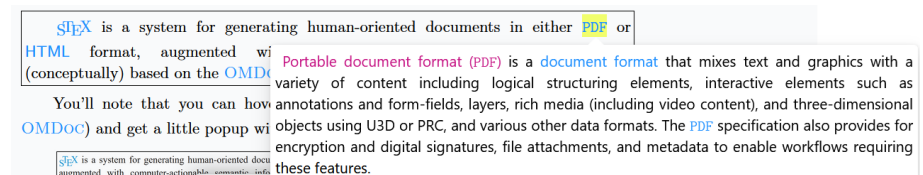


Figure 3: Definition on Hover

```

File 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).

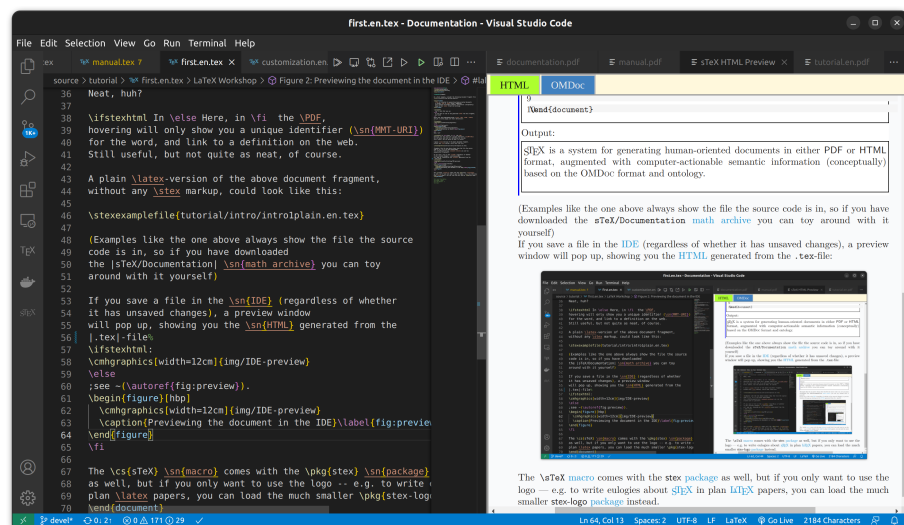


Figure 4: Previewing the document in the IDE

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 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ papers, you can load the much smaller **stex-logo** package instead.

2.1 Text symbols

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

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

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

2.1.1 Using Modules & Search in the IDE

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

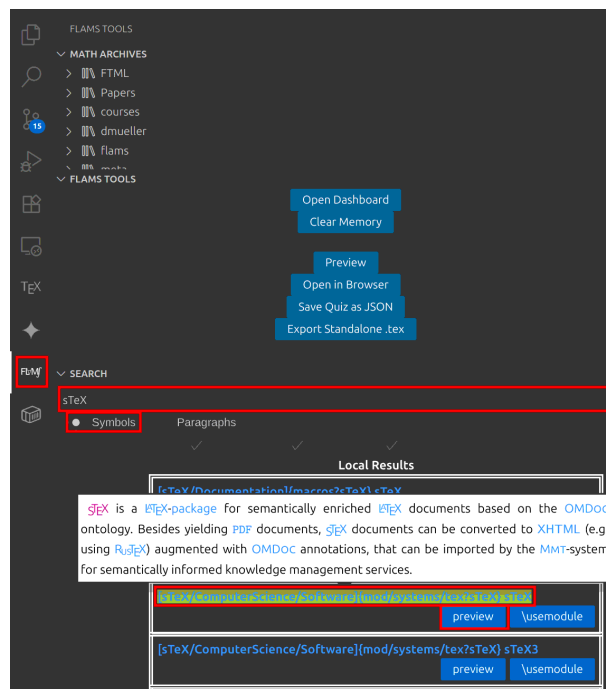


Figure 5: Search in the sTeX IDE

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 FTML for the document – the resulting window that pops up also has an OMDoc button you can click, which (among other things) shows you the semantic macros provided by the respective module: In this case, it tells us that there is a *text symbol* named “sTeX” with semantic macro `\stex` in the module `sTeX` in the same document. It produces the presentation “STeX” as we want (Figure 6).

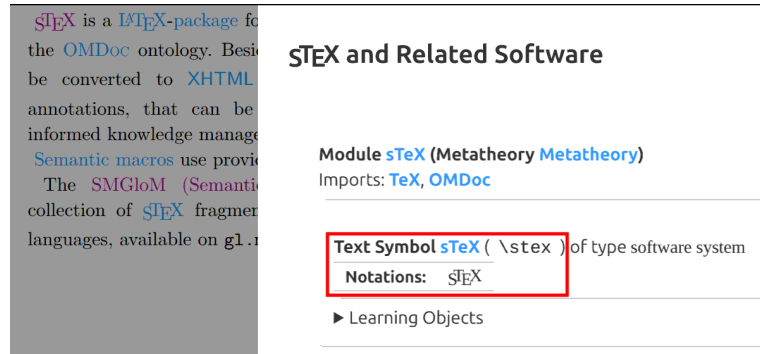


Figure 6: OMDoc Preview

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

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

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

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

Example 2

Input:

File tutorial/intro/intro1stex.en.tex

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

Output:

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

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

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

- 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 `stexmmt.mathhub.info/:sTeX`, where the [HTML](#) for all the [symbols](#) we use in this document are hosted.

2.2 [Symbol](#) References

Let's continue with the next paragraph of section 1.1; for now unannotated:

[Example 3](#)

Input:

File tutorial/intro/intro2plain.en.tex

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

Output:

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

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

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

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

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

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

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

In general, any `macro` that expects a `symbol` name can be given either

1. the *name* of the `symbol`,
2. the name of its `semantic macro`,
3. or any suffix of its `MMT-URI` containing at least the `module` name.

STEX

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

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

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

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

Example 4

Input:

```
File 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 `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`.

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 `LATEX` `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 [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](#).

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 7). You will then be asked for

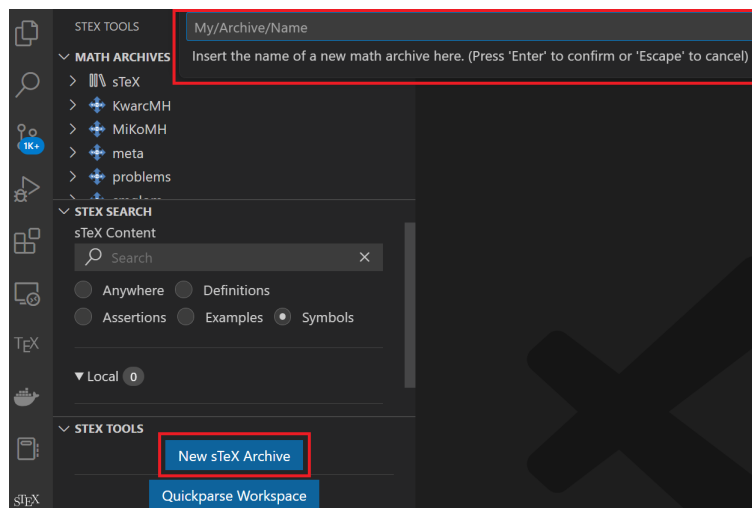


Figure 7: New [Math Archive](#) in the [IDE](#)

the name of the [archive](#), and a url-base, where the content is supposedly going to end up

online. You can safely keep the defaults for the latter. In the following, we assume that your archive is named `my/archive`.

The IDE will then create the following files and directories in your MathHub directory:

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

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

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

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

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

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

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

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


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

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

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

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

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

 Note, that the IDE immediately picks up on this and displays the full FTML URI of our new module over the `\begin{smodule}{LaTeX}` (Figure 8) –



```

my > archive > source > %X latex.tex > {} https://mathhub.info?a=my/archive&p=latex&m=LaTeX
1 \documentclass{stex}
2 \begin{smodule}{LaTeX}
3   https://mathhub.info?a=my/archive&p=latex&m=LaTeX
4 \begin{smodule}{LaTeX}
5 ...
6 \end{smodule}
7 \end{document}

```

Figure 8: VS Code URI on Hover

From this, we can glimpse that the namespace of the module is `http://mathhub.info?a=my/archive&p=latex&m=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?a=my/archive&m=LaTeX`, 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 9).



```

my > archive > source > %X LaTeX.en.tex > ...
1 \documentclass{stex}
2 \begin{smodule}{LaTeX}
3   https://mathhub.info?a=my/archive&m=LaTeX
4 \begin{smodule}{LaTeX}
5 ...
6 \end{smodule}
7 \end{document}

```

Figure 9: VS Code URI on Hover



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

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

```

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

```

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

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

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

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



As we cannot rely on the underlying `\TeX` engine treating UniCode charaters natively, symbol names can only consist of printable ASCII characters.

2.4 Documenting Symbols

We can now use the two new macros, `\symname/\sn`, `\symref/\sr` etc. to mark up the above two paragraphs. However, the system does not yet know what to show a reader when hovering over the `symbol` in the `HTML`. We can fix that by using 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 `\TeX` to use this paragraph as a documentation for the symbols provided in the `for=` option.

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 `\TeX` 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 \definame{macro} expansion engine.
\end{sparagraph}

```

Exercise

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

Mathematics.en.tex

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

– Jordan Ellenberg

PDF.en.tex

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

HTML.en.tex

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

OMDoc.en.tex

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

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

sTeX.en.tex

```

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

```

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

The preamble of every file should only be

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

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

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

2.5 Sectioning and Reusing Document Fragments

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

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

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

In there, we want sections as follows:

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

We could of course do the following:

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

...but this approach has two drawbacks:

Firstly, we need to manually keep track of the section levels, by explicitly writing `\section`, `\subsection` etc. This is fine as long as we are just interested in this particular article. But what if we want to *reuse* the article’s content in another document at some point? The section levels might be entirely different then – e.g. we might want the “Preliminaries” section to be a subsection instead.

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

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

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

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

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

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

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

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

Finally, we can make two more minor changes:

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

We can achieve this using the `\ifinputref` conditional: by wrapping the header in an `\ifinputref \else...\fi`, it will only be processed if the file is *not* being loaded using `\inputref`. `\ifinputref` is a “classic” `TeX` conditional and is treated as such in both `PDF` and `HTML` compilation. A smarter `macro` to use is `\IfInputref`, which takes two arguments for the *true* and *false* cases, respectively. Additionally, when compiling to `HTML`, *both* arguments to `\IfInputref` will be processed, and the back-end 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 10 Where did all of our fragments go?



Figure 10: Missing Fragments in the `HTML` Preview

Well, they don’t exist yet as `HTML`. The `HTML` Preview window in the `IDE` is really just that: A *preview*. But when using `\inputref`, it has to find the `HTML` of the `\inputrefed` fragment *somewhere*. Meaning: we have to compile all of the fragments we used to `HTML` first. Individually, we can compile the currently open file and all its dependencies in `VS Code` using the button in Figure 11.

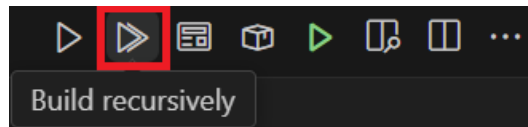


Figure 11: The Build PDF/XHTML/OMDoc Button

This will queue all dependencies of the file in a new build queue and open the [FLaMj](#) dashboard for us (see Figure 12). Clicking on the run button in the dashboard will then

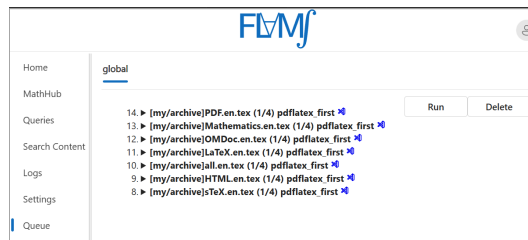


Figure 12: The FLAMS build queue

for every queued file:

1. Run `pdflatex` twice,
2. Convert the file to [HTML](#),
3. Extract all the semantics, and
4. Construct a search index.

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

At this point, [FLaMj](#) knows about the [HTML](#) and the semantics, and we can look at the [HTML](#) in [VS Code](#) or the [FLaMj](#) dashboard – of course, features like the fancy pop-up windows require a semantically informed back-end infrastructure, in the form of the [FLaMj](#) system. However, [FLaMj](#) can dump a standalone and self-contained [HTML](#) 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 13).

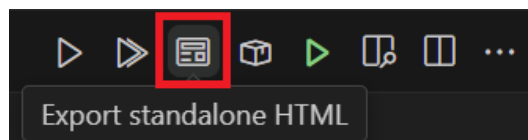


Figure 13: Exporting [HTML](#) in the [IDE](#)

In the dialog box that opens now, select an **empty** directory and [FVMf](#) 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 without issues, because most browsers forbid javascript modules on the `file://` protocol, but opening the file via `http` will yield the desired result, and you can now upload the directory's content to wherever you might want to use it.

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

Chapter 3

Mathematical Concepts

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

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

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

The example snippets in this chapter can be found in the [math archive sTeX/MathTutorial](#). If you downloaded the [sTeX/Documentation archive](#) in the [sTeX IDE](#), you already have that [archive](#). If not, you can download it from within the [IDE](#), as described in Part 1 (The Basics) in the [sTeX](#) tutorial.

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

STEX

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

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

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

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

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

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

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

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

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

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

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

STEX

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

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

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

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

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

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



You may be wondering about the role of the `\mathrel` macro in the example above: \TeX determines spacing/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	T_EX macro	examples
ordinary (default class)	<code>\mathord</code>	$\alpha \ i \ \Diamond$
large operator	<code>\mathop</code>	$\sum \prod \int$
opening	<code>\mathopen</code>	$([\langle$
closing	<code>\mathclose</code>	$)] \rangle$
binary relation	<code>\mathrel</code>	$\leq > =$
binary operator	<code>\mathbin</code>	$+ \cdot \circ$
punctuation	<code>\mathpunct</code>	$, ;$

T_EX “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 T_EX macro for the desired class (e.g. `\mathrel{\comp{\leq}}`).

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

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

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

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

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

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

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

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

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

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

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

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

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

Exercise

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

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

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

3.1.2 Types and Variables

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

Types are particularly useful for *variables*:

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

Output:

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

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

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

The above fragment uses two variables a and b . In fact, `FLMj` 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}
\Definame{addition} $\defnotation{\plus{va}{vb}}$
  refers to the process of adding two \sn{Nat}.
\end{sparagraph}
```

Output:

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

Okay, so now a and b are gray, but besides that, we haven’t achieved much yet. Let’s change that by giving the variables the type \mathbb{N} :

Example 7

Input:

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

Output:

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

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

We can of course also assign types to symbols. In the IDE, find the symbol “function space” with semantic macro `\funspace` (in `[sTeX/MathBase/Functions]{mod?Function}`). The OMDoc preview window shows you how to use this symbol (Figure 14). This tells us that if we write `\funspace{a_1, \dots, a_n}{b}` (depending on which notation we use), we will get $a_1 \times \dots \times a_n \rightarrow b$.

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

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

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

Symbol function space (`\funspace{a_1^1, \dots, a_1^n}{i_2}`)
of type `bind((A,B),SET)`

Notations: $a_1^1 \rightarrow \dots \rightarrow a_1^{i_1} \rightarrow i_2$ $a_1^1 \times \dots \times a_1^{i_1} \rightarrow i_2$ $a_1^1 \rightarrow \dots \rightarrow a_1^{i_1} \rightarrow i_2$ $a_1^1 \times \dots \times a_1^{i_1} \rightarrow i_2$

▼ Associated Paragraphs

Figure 14: Syntax Preview



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 15.

addition

Select Definition or Example

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

Force Notation

▼ Formal Details

Symbol addition (`\plus{a_1^1, \dots, a_1^n}`)
of type `ℕ⇒ℕ⇒ℕ`

Notations: $a_1^1 + \dots + a_1^{i_1}$

Figure 15: On-Click Popup in the `HTML`

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}
```

3.1.3 Flexary Macros and Argument Modes

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

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

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

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

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

Example 8

- For `\plus{a,b,c}` yielding $a + b + c$, we do `\symdecl{plus}[args=a]`,
- for `\inset{a,b,c}{A}` yielding $a, b, c \in A$, we do `\symdecl{inset}[args=ai]`,
- in `\add{i}{1}{n}{f(i)}` yielding $\sum_{i=1}^n f(i)$, the variable i is **bound** in the expression, we hence do `\symdecl{add}[args=biii]`,
- in `\forall{x,y,z}{P(x,y,z)}` yielding $\forall x, y, z. P(x, y, z)$, the variables x, y, z are all **bound** by the \forall , we hence do `\symdecl{forall}[args=Bii]`.

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

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

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

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

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

TODO² TODO³

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

²TODO: argmap

³TODO: argarraymap

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.

sTeX

`\notation` (and hence `\symdef`) take an optional argument `prec=<opprec>;<argprec1>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 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}}}
```

3.1.6 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}}
```

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

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

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

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

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

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`.

⁴TODO: `seqmap`

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.

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

A first variant of a corresponding *module* could look like this:

Example 9

Input:

```

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

```

Output:

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

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



Note that the above is more than good enough, if you merely want to produce nice-looking, “wikified” *HTML* and *PDF* documents. The rest of this subsection will cover how to add more flexiformal semantics to the above. If this seems laborious and/or difficult, keep in mind that this is to some degree experimental still, and you are not forced to go overboard with semantic annota-



tions!

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

Semantic Macros in Text Mode

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

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

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

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

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

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

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

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

So we could now do

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

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

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


```

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

```

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

Definientia

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

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

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

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

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

```

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

```

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

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

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

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

Example 10

Input:

```

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

```

Output:

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

And indeed, if we look at the [OMDOC](#) tab of the [HTML](#) preview, we can see that not only does [MMT](#) attach the `definiens` to the `symbol`, it has also inferred the `type` of “`associative`” from the `definiens` (Figure 16).

▼ Symbol `associative`

Definiens

$$\{A: \text{SET}\}_I (\circ: A \rightarrow A \rightarrow A) \rightarrow \forall a, b, c: A. \left(\frac{(a \circ b) \circ c = a \circ (b \circ c)}{A} \right)$$

Type

$$\{A: \text{SET}\}_I (\circ: A \rightarrow A \rightarrow A) \rightarrow \text{Prop}$$

Figure 16: 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- \LaTeX 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 17).

Assertion	<code>assertion</code>
Term term	<code>associative (+)</code>

Figure 17: Assertion Statement in `OMDoc`

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

3.2.3 Proofs



\LaTeX 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 \LaTeX , 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 \definame{magma} is a \sr{mathstruct}{structure} $\magma!$,
15 where $\univ$ is a \sn{collection} and $\op!$
16 a binary operation $\funspace{\univ,\univ}\univ$.
17 \end{sdefinition}
18 \end{mathstructure}
```

Output:

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

Instantiating Structures

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

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

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

Syntax	Result
$\$ \backslash \mathbf{M} ! \$$	M
$\$ \backslash \mathbf{M} \{ \} \$$	$\langle U_M, \circ_M \rangle$
$\$ \backslash \mathbf{M} \{ \text{univ} \} \$$	U_M
$\$ \backslash \mathbf{M} \{ \text{op} \} ! \$$	\circ_M
$\$ \backslash \mathbf{M} \{ \text{op} \} \{ a, b, c \} \$$	$a \circ_M b \circ_M c$

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

By default, [S_{TE}X](#) 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 `\$ \backslash \mathbf{M} \{ \text{op} \} \{ a, b \} \$` ultimately outputs `a \circ_{\backslash \mathbf{M} !} b`. Hence, we get $a \circ_M b$.

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

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

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

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

This allows us to do things like:

```
Let \$ \backslash \mathbf{M} ! := \backslash \mathbf{M} \{ \} \$ and \$ \backslash \mathbf{M} p ! := \backslash \mathbf{M} p \{ \} \$ \sns{magma}. Then...
```

yielding

Let $M := \langle U, \circ \rangle$ and $M' := \langle U', \circ' \rangle$ [magmas](#). Then...

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

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

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

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

```
Let \$ \backslash \mathbf{M} ! := \backslash \mathbf{M} \{ \} \$ and \$ \backslash \mathbf{M} p ! := \backslash \mathbf{M} p \{ \} \$ \sns{magma} on \$ \backslash \mathbf{A} \$....
```

Let $M := \langle A, \circ \rangle$ and $M' := \langle A, \circ' \rangle$ [magmas](#).

Of course, we can also use `return=` with [variable](#) sequences – for example:

```
\varseq{\vMs}[name=M,return={\magma[comp={##1}_{#1] },op=(M_i)_1^n]{1,\ellipses,n}{\maincomp{M}_{#1} }
```

```
Let \$ \backslash \mathbf{M} s ! := \backslash \mathbf{M} s \{ i \} _1^n \$ a sequence of \sns{magma}...
```

Let $(M_i)_1^n := \langle U_i, \circ_i \rangle_1^n$ a sequence of [magmas](#)...

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

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

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



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

3.3.2 Extending `Structures` and Axioms

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

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

Example 12

Input:

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

Output:

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

Note our usage of `\inlineass` to generate a new `symbol` for the `associative axiom`.

If we look at the `OMDOC` tab in the `HTML` preview window, we can see the output in Figure 18.

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

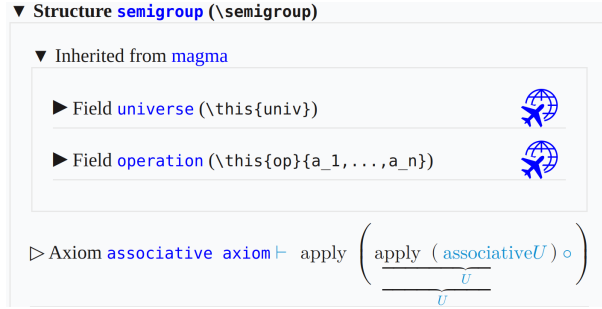


Figure 18: Axioms in OMDoc

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

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

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

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

Consider the following conservative extension:

Example 13

Input:

```

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

```

Output:

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

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

```

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

```

...producing a^2 .

3.3.3 Nesting Structures and `\this`

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

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

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

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

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

```

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

```

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

```

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

```

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

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

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

```

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

```

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

Example 14

Input:

```

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

```

Output:

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

Now if we instantiate our `magma homomorphism`:

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

Here is a list of what we can do now:

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

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



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

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

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

3.4 Complex Inheritance and Theory Morphisms



We are starting to approach seriously experimental territory.

While the theory behind all the following is relatively well understood, and their implementation in [MMT](#) is mature, the same can not be said out the implementation in [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 $\langle U, R \rangle$ is called a **partial order** (or **poset**), if R is antisymmetric.

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

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

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



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.

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

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

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

Exercise

Try to implement all of the above yourself!

3.4.1 Glueing Structures Together

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

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

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

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

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

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

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

```

\copymod{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 $a \mathrel{R} b$ iff $a \circ b = a$.

Satz 3.4.22. $\langle U, R \rangle$ is a *meet semilattice*.

Proof:

We need to prove the following

1. **reflexivity** ($a R a$):

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

2. **antisymmetry** ($a R b$ and $b R a$ implies $a = b$):

Assume $a \circ b = a$ and $b \circ a = b = a \circ b$ (by the **commutative axiom**). Hence, $a = b$

3. **transitivity** (If $a R b$ and $b R c$, then $a R c$.):

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, $a R c$.

4. $a \circ b$ is the **infimum** of $\{a, b\}$:

By definition (and the **commutative axiom**), $a \circ b R a$ and $a \circ b R b$. We need to show, that if $x R a$ and $x R b$, then $x R a \circ b$. 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 $x R a \circ b$

□

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

Output:

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

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

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

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

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

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

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

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

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

Chapter 4

Extensions for Education

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

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

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

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

4.1 Slides and Course Notes

4.1.1 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 `sTeX`. 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.

4.1.2 Package Options

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

<hr/> slides notes <hr/>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see subsection 10.1.3).
<hr/> sectocframes topsect <hr/>	If the option <code>sectocframes</code> is given, then for the <code>sfragments</code> , special frames with the <code>sfragment</code> title (and number) are generated. The <code>topsect</code> allows to specify the top section level (e.g. <code>chapter</code> , <code>section</code> , ...) for documents that are formally stand-alone, but intended to be chapters, sections, or
<hr/> frameimages fiboxed <hr/>	If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code> -generated frames (see ???). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.

4.1.3 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. Content of this environment is only shown in *notes mode*.

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 ...

```

¹EDNOTE: MK: maybe this option should be used in the other `STEX` classes as well. Actually I think this already is treated by `stex.sty`; where to document?

`\ifnotes` Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notesttrue` or `\notesfalse` is strongly discouraged however.



We need to give the title frame the `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.



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

`\inputref*` If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

`nparagraph (env.)` There are some environments that tend to occur at the top-level of `note` environments.
`nparagraph (env.)` We make convenience versions of these: e.g. the `nparagraph` environment is just an
`ndefinition (env.)` `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one
`nexample (env.)` level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,
`nsproof (env.)` `nsproof`, and `nassertion` environments.
`nassertion (env.)`

4.1.4 Managing and Customizing Themes


As the `notesslides` package is based on the `beamer` class, it can in principle (in slides mode) take advantage of all `beamer` themes. Themes can be specified/loaded with the normal `\usetheme`, but the `notesslides` package provides the `\libusetheme` macro:

`\libusetheme` In analogy to the `\libinput` functionality of the `stex` package, this macro allows loading `beamer` themes from the `lib` directories leading up to the current archive.

Unfortunately, there is no `beamer` functionality of using a themed `frame` environment in `article` mode. Therefore `notesslides` has to hand-craft one – for a very limited set of themes. The `notesslides` package and class comes with a simple default theme named `sTeX` that provided by the `beamterthemesTeX.sty` style file. 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

Test Slide

▷ Some text


©: MiKo
1
sTeX

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

\setslidelogo The default logo provided by the `notesslides` package is the \TeX logo it can be customized using `\setslidelogo{<logo name>}`.

\setsource The default footer line of the `notesslides` package mentions copyright and licensing. In `notesslides \source` stores the author's name as the copyright holder. By default it is the author's name as defined in the `\author` macro in the preamble. `\setsource{<name>}` can change the writer's name.

\setlicensing For licensing, we use the Creative Commons Attribution-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

These macros can be used set up a institution-specific theme; for instance the one for FAU simply customizes the logo:

```
1 % Beamer slide theme for FAU;
2 \typeout{Beamer FAU theme}
3 \RequirePackage{beamerthemesTeX}
4 \setslidelogo[courses/FAU/admin]{PIC/FAU_Logo_Bildmarke.png}
```

This could be placed in one of the `lib` folders available to the current archive and could then be activated with

```
1 \documentclass[slides]{notesslides}
2 \libusetheme{FAU}
```

in the preamble of the main `notesslides` file.

4.1.5 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add \TeX notes.

\frameimage In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\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: 

4.1.6 Ending Documents Prematurely

`\prematurestop`
`\afterprematurestop` For prematurely stopping the formatting of a document, \TeX 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 \TeX preamble of the course notes file.

4.1.7 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>}`
`\useSGvar` 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.

4.1.8 Excursions

In course notes, we sometimes want to point to an “excursion” – material that is either presupposed or tangential to the course at the moment – e.g. in an appendix. The typical setup is the following:

```

1 \excursion{founif}{../fragments/founif.en}
2 {We will cover first-order unification in}
3 ...
4 \begin{appendix}\printexcursions\end{appendix}

```

It generates a paragraph that references the excursion whose source is in the file `../fragments/founif.en.tex` and automatically books the file for the `\printexcursions` command that is used here to put it into the appendix. We will look at the mechanics now.

`\excursion` The `\excursion{<ref>}{<path>}{<text>}` is syntactic sugar for

```

1 \begin{nparagraph}[title=Excursion]
2   \activateexcursion{founif}{../ex/founif}
3   We will cover first-order unification in \sref{founif}.
4 \end{nparagraph}

```

`\activateexcursion` Here `\activateexcursion{<path>}` augments the `\printexcursions` macro by a call
`\printexcursion` `\inputref{<path>}`. In this way, the `\printexcursions` macro (usually in the appendix)
`\excursionref` will collect up all excursions that are specified in the main text.

Sometimes, we want to reference – in an excursion – part of another. We can use `\excursionref{<label>}` for that.

`\excursiongroup` Finally, we usually want to put the excursions into an `sfragment` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro: `\excursiongroup[id=<id>,intro=<path>]` is equivalent to

```

1 \begin{note}
2 \begin{sfragment}[id=<id>]{Excursions}
3   \inputref{<path>}
4   \printexcursions
5 \end{sfragment}
6 \end{note}

```

Contents

4.2 Problems and Exercises

4.2.1 Background

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

- *homework assignments and formal exams*, where they are graded and points are awarded for course credit,

- *self-study materials* that allow learners explore applications of some concepts and/or practice,
- *automated tutoring systems* to determine learner competencies to trigger computer supported learning services.

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

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

- open problems, where expected answers are free-form, and classification into answer classes is usually a manual process; see subsection 4.2.3
- single/multiple choice questions where the answer classes and thus feedback/grading correspond to the selection pattern of items; see subsection 4.2.7.
- fill-in-the-blanks questions where we may want to specify how the answer string is interpreted; see subsection 4.2.8.

Additionally, problems and exercises often come with text fragments that serve auxiliary functions: hints, notes, and and grading specifications. Furthermore, we can specify how long solving a given problem is estimated to take and how many points will be awarded for a perfect solution – e.g. in an exam. See subsection 4.2.10 for details.

The `problem` package gives semantic markup support for all these aspects of problems/exercises with a focus on problem libraries that collect carefully formulated and tested problems/exercises, so that they can be re-used in many contexts. It also tries to support all possible stakeholders in dealing with problems/exercises:

- *learners*, who try to solve problems and may profit from feedback
- *instructors*, who pose problems in homeworks, quiz, and exams,
- *graders*, who try to assess learner’s competencies from the answers given
- *authors* of learning objects and (automated) *course generators*, who may use problems to diagnose learner’s competencies (e.g. as prerequisites).

4.2.2 The Package, Options, and Configuration

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

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

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

The `problem` package generates content and labels according to the language specified in the `lang` key of the main document², these can be localized by in the files `ldf/problem-<lang>.ldf`⁶.

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

Note that documents (or document fragments) with problems are often formatted in different versions that can be configured by these options. Therefore the following variant of the preamble prefix above can be very useful:

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

We add a level of indirection via the `\probopts` macro (`\providecommand` sets it unless it already exists). If the current file is `foo.tex`, then we can make a L^AT_EX file `foo-test.tex` that formats in test mode as

```
\newcommand\probopts{,test}\input{foo}
```

Alternatively, we do not need a file at all: we can just (e.g. in a Makefile or a PDF generation script) issue the following command in the terminal:

```
pdflatex "\def\probopts{,test}\input{foo}"
```

4.2.3 (Open) Problems

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

Example 16

Input:

²EdNOTE: MK: document the attribute somewhere

⁶The current crop of language definition files is quite limited. Please feel free to contribute the to the localization effort

```

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}[title=deduct 5pts if justification is missing]
11    \anscls[feedback=you forget that elephants could be small]{none}
12    \anscls[feedback=you forget the back seats]{2}
13  \end{gnote}
14  \begin{solution}
15    Four, two in the front seats, and two in the back.

```

Output:

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

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

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

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

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

4.2.4 Solutions and Answer Classes

Most problem libraries also contain (master) solutions, which show the intended way of solving a problem. The `solution` environment allows to specify solutions. It is only formatted if the `solutions` options is specified for the `problem` package.

We observe that many problems have more than one possible solution, so the `problem` package allows multiple `solution` environments; they can be distinguished by an `id` key in the optional first argument.

We also observe that in problem libraries we may want to keep track of classes of answers that have been observed e.g. in previous assessment situations, e.g. to provide

feedback. In this setting, the `solution` environment can be used to encode (typical representative of) **answer classes** (see also ??? below).

`solution` The `solution` environment takes an optional argument that allows the keys `id`, `title` and `style` keys that we have already seen above, further more

1. `testspace=`, which is equivalent to a `\testspace{}` (see subsection 4.2.10).
2. `answerclass=`, which gives the name of an answer class, this solution corresponds to.³

EdN:3

4.2.5 Grading Support

When problems are graded by multiple persons (e.g. for large university exams), then consistency of grading is a major issue. Both within a cohort and between cohorts e.g. in successive instances of a course.

Therefore keeping records of how to grade a problem (which problem is worth how many “points” and where to deduct how many points in which situations) and making them accessible to graders in a “master solution” – a version of an exam formatted to have the known solutions/answer classes (see above) and the grading specifications is a good practice. And it makes sense to include grading information into a problem library.

The simplest and arguably most effective measure is to specify the points that can be reached for a problem in the problem itself via the ‘pts=’ key (but see subsection 4.2.9)

The `problem` package also supplies the `gnotes` environment, which is only formatted when the `gnotes=` key is given. This environment takes an optional argument that allows the `id`, `style`, and `title` keys. The latter is used to give a short version of the grading specification. More structured grading support can be expressed in `\anscls` markup which we discuss next.

We have already encountered the notion of answer classes – classes of answers that are supposed equivalent in terms of learner’s competencies inscribed in them – above. These are the natural carriers of both grading feedback and points.

In the `gnote` environment we use the `\anscls` macro for specifying such information using meta-level descriptions rather than typical representatives. It takes an optional keyword argument for the grading/feedback information and a required one for the answer class description.

We distinguish two usages of `\anscls[...]{<desc>}`: in

- *answer classes* `<desc>` is a specification of a class of answers and the `pts=` key the points to be awarded to that.
- *answer traits* `<desc>` specifies the deviation from an assumed correct solution, and the value of the `pts=` key specifies changes to the points specified in the problem itself.

Consider for instance the following situation:

```
\begin{sproblem}[pts=3]
  Give an example for a foo and explain it in one sentence.
  \begin{gnote}[title=1 pt for the example and 2 for the explanation]
    \anscls[pts=0,feedback=I did not understand this]{complete gibberish}
    \anscls[pts=-2,feedback=You forgot the explanation]{no explanation}
  \end{gnote}
\end{sproblem}
```

³EdNOTE: This mechanism needs to be further worked out or deprecated; if we keep it we probably need to add `pts` and `feedback` attributes for parity with `\anscls`.

Here we have a problem that is worth three points, and the grading note explains how to treat deviations from a correct solutions – which is not given, since there may be hundreds of examples for foos. The `\anscls` specify this out for grading support systems⁷. The first is an answer class description for the class of answers that cannot be made sense of by the graders and specifies a default grade and feedback. The second specifies the trait of a missing explanation and specifies a deduction commensurate with the grading note title and a feedback to the learner. Note that a grading support system can distinguish answer classes from traits by the form of the value of points: absolute values vs. differences like +5 or -3.

4.2.6 Structured Problems

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

Example 17

Input:

File tutorial/ext/structured-problem.en.tex

```

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

```

Output:

⁷like e.g. <https://gitos.rrze.fau.de/yn06uhoc/vollkorn-prototype>

Exercise (Fitting Elephants Modularly)

Consider the problem of fitting elephants into a Volkswagen beetle.

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

Lösung: Two on each side one.

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

Lösung: Three little elephants.

3. What is the total number?

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

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

Note that neither `sproblem` nor `subproblem` can be nested, if you want to have further structure, you need to resort to regular \LaTeX functionality. Note that you can add `solution` and `gnote` environments wherever you want, so that this need not force you to forego structure.

The `subproblem` environment is however very useful for modularizing problems: `subproblem` environments can appear outside of `sproblem` and in particular at the top-level of a file. This allows them to be shared between `sproblem` via `\inputref`.

4.2.7 Single/Multiple Choice Blocks

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

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

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

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

Example 18

Input:

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

Output:

Exercise (Functions)

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

- ☒ def
- ☐ function^a
- ☐ fun^b
- ☐ public static void^c

^a

that is for C and C++

^b

that is for Standard ML

^cNoooooooooo

that is for Java

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

Example 19

Input:

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

Output:

Exercise (Functions)

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

- ☐ def
- ☐ function
- ☐ fun
- ☐ public static void

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

Example 20

Input:

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

^a

that is for C and C++

^b

that is for Standard ML

^cNoooooooooooo

that is for Java

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

Analogously, single choice blocks are marked up by the `scc` environment and the individual choices by the `\scc` macro. In PDF, there is little difference to the multiple choice case. In interactive formats like HTML, single choice blocks are typically realized via radio buttons to enforce single choice.⁴

Often we only want to ask whether a statement is true or false. This is essentially a single choice block. \TeX provides the macros `\yesTnoF`, `\yesFnoT`, `\trueTfalseF`, and `\trueFfalseT` for that. They are used as in the example below:

⁴EdNOTE: MK: are there any differences between `mcc` and `scc` we need to discuss here?

Example 21

Input:

```
1 \begin{sproblem}\startsolutions
2   Mark the following statements as true or false:
3   \begin{enumerate}
4     \item The moon is made of green cheese \trueFfalseT
5     \item \ldots
6   \end{enumerate}
7 \end{sproblem}
```

Output:

Exercise

Mark the following statements as true or false:

1. The moon is made of green cheese ☐ true ☒ false
2. ...

4.2.8 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 22

Input:

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

Output:

Exercise (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle?

and the actual solution in solutions mode:

Example 23

Input:

```

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

```

Output:

Exercise (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle? 4

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

Obviously, the required argument of `\fillinsol` can be used for auto-grading. For concrete data like numbers, this is immediate, for more complex data like strings “soft comparisons” might be in order. Indeed the default matching of answers against the string provided in the required argument of the `\fillinsol` is up to whitespace normalization.

The `problem` package allows to specify different behavior via three additional keys whose arguments are all triples of the form

```
{\langle target \rangle}{\langle TF \rangle}{\langle feedback \rangle}
```

where $\langle target \rangle$ is the target specification, $\langle TF \rangle$ is the verdict T or F on the correctness of an answer that is accepted by $\langle target \rangle$, and $\langle feedback \rangle$ a feedback string. In the following (somewhat contrived) example, we see all three at work:

Example 24

Input:

```

1 \begin{sproblem}
2   In 2019, Erlangen is a city of
3   \fillinsol[testspace=3cm,
4   exact={111962}T{Wow, you known your numbers, according to
5     Wikipedia that is exactly correct},
6   numrange={0-1000}F{No,that would be a village},
7   numrange={1001-100000}F{No,that would be a town},
8   numrange={100000-120000}T{Yes, it is close to 109000},
9   numrange={1200001-}F{No, that is too large},
10  regex={-[0-9]+}F{What do negative Inhabitants even mean?},
11  regex={.*}F{Please give a natural number}]
12  {111962}
13  inhabitants.
14 \end{sproblem}

```

Output:

Exercise

In 2019, Erlangen is a city of 111962^a inhabitants.

^a

type	case	verdict	feedback
exact	111962	T	
exact	111962	T	Wow, you know your numbers, according to Wikipedia that is exactly correct
numrange	0-1000	F	No, that would be a village
numrange	1001-100000	F	No, that would be a town
numrange	100000-120000	T	Yes, it is close to 109000
numrange	1200001-	F	No, that is too large
regex	-[0-9]+	F	What do negative inhabitants even mean?
regex	.*	F	Please give a natural number

- **exact=** gives the exact string (no whitespace normalization).
- **numrange=** specifies a number range, i.e. a comma-separated list of pairs $\langle low \rangle - \langle high \rangle$, where $\langle low \rangle$ and $\langle high \rangle$ are number representations or empty.
- **regex=** gives a POSIX regular expression that specifies all acceptable strings.

4.2.9 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.

4.2.10 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`

TODO⁸

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

<hr/> <code>solutions</code>	The <code>hwexam</code> package and class take the options <code>solutions</code> , <code>notes</code> , <code>hints</code> , <code>gnotes</code> , <code>pts</code> , <code>min</code> , and <code>boxed</code> that are just passed on to the <code>problems</code> package (cf. its documentation for a description of the intended behavior).
<code>notes</code>	
<code>hints</code>	
<code>gnotes</code>	
<code>pts</code>	
<code>min</code>	
<hr/> <code>multiple</code>	Furthermore, the <code>hwexam</code> package takes the option <code>multiple</code> that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).
<code>test</code>	Finally, there is the option <code>test</code> that modifies the behavior to facilitate formatting tests. Only in <code>test</code> mode, the macros <code>\testspace</code> , <code>\testnewpage</code> , and <code>\testemptypage</code> have an effect: they generate space for the students to solve the given problems. Thus they can be left in the L ^A T _E X source.

4.3.3 Assignments

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

4.3.4 Including Assignments

<hr/> <code>\includeassignment</code> <hr/>	The <code>\includeassignment</code> 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 <code>assignment</code> environment in the included file). The keys <code>number</code> , <code>title</code> , <code>type</code> , <code>given</code> , and <code>due</code> are just as for the <code>assignment</code> environment and (if given) overwrite the ones specified in the <code>assignment</code> environment in the included file.
---	---

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

4.3.5 Typesetting Exams

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

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

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2026-06-25

You have one hour (sharp) for the test;

Write the solutions to the sheet.

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

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

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

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

good luck

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](https://mathhub.info/?a=sTeX/Documentation&d=manual&l=en) version of this part can be found at
<https://mathhub.info/?a=sTeX/Documentation&d=manual&l=en>

Chapter 5

Basics

5.1 Package and Class Options

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

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

- `mathhub=<path>`: Uses the provided file path as MathHub directory (see Section 1 ([Math Archives](#) and the MathHub Directory) in the [sTeX](#) Manual).
- `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`3.3 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,
- `courses/*` – 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 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://mathhub.info
dependencies:sTeX/ComputerScience/Software,sTeX/MathTutorial
ignore: */code/*|*/tikz/*|*/tutorial/solution/*
```

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

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

source-base or

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

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

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

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

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

5.3 The lib-Directory

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

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

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

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

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

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

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

\libusepackage **\libusepackage** [package-options]{some/file} searches for a file **some/file.sty** in the same way that **\libinput** does, but will call **\usepackage**[package-options]{path/to/some/file} instead of **\input**.
\libusepackage throws an error if not *exactly one* candidate for **some/file.sty** is found.

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

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

throws an error if not *exactly one* candidate for 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

<code>\sTeX</code> <code>\stex</code>	The <code>\sTeX</code> macro produces this \sTeX logo. It is provided by the <code>stex-logo</code> package , included with the <code>stex</code> package .
--	--

<code>\ifstexhtml</code>	The TeX conditional <code>\ifstexhtml</code> is <i>true</i> if the current compilation generates HTML , and <i>false</i> otherwise (i.e. generates PDF).
--------------------------	---

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

Chapter 6

Document Features

6.1 Document Fragments

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

The `<id>` can be used for cross-document references (see Section 0.3 (Cross-Document References) in the `TeX` Manual).

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

`\skipfragment` The `\skipfragment` macro “skips an `sfragment`”, i.e. it just steps the respective sectioning counter. This macro is useful, when we want to keep two documents in sync structurally, so that section numbers match up: Any section that is left out in one becomes a `\skipfragment`.

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

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

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

6.2 Using and Referencing Document Fragments

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

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

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

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

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

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

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

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

`\mhgraphics` If the `graphics` package is loaded, `\mhgraphics` takes the same arguments as `\includegraphics`,
`\cmhgraphics` 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` additionally wraps the image in a `center`-environment.

`\lstinputmhlisting` Like `\mhgraphics`, but for `\lstinputlisting` instead of `\includegraphics`. Only de-
`\clstinputmhlisting` fined 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>}`

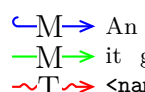
Like `\sref`, but with the third argument mandatory, `\extref` will *always* produce the output as if `<label>` would *not* occur in the current document.

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 `module` 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 Part 1 (Customizing Typesetting) in the `sTeX` Manual

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

7.2 Symbol Declarations

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

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

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

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

`\symdecl` takes the following optional arguments:

`name` see above,

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

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

`def` the `symbol`'s definiens,

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

`\textsymdecl` [\textsymdecl](#){*mname*}[*options*]{*code*}

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

Additionally, a [macro](#) [\mname](#)name is generated that produces *code* without any semantic markup.

`\symdef` [\symdef](#){*mname*}[*options*]{*notation*}

Combines the functionalities and optional arguments of [\symdecl](#) and [\notation](#) in one.

7.2.1 Returns

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

1. [S_{TE}X](#) will replace `#1` and `#2` in *code* by `a` and `b`, yielding *retcode*.
2. [S_{TE}X](#) 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*}

`\sr`

The [\symref](#) [macro](#) (and its short version [\sr](#)) is the most general variant to mark-up arbitrary [L^AT_EX](#) code *text* with the [symbol](#).

¹¹[TODO: experimental](#)



This is as good a place as any other to explain how $\text{\texttt{S\TeX}}$ resolves a string `symbol` to an actual `symbol`.

If `\symbol` is a `semantic macro`, then $\text{\texttt{S\TeX}}$ 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...`. $\text{\texttt{S\TeX}}$ attempts to handle this case thusly:

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

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

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

`\sn` If the `symbol` referenced by `<symbol>` has name `name`, this is a shortcut for

`\Symname` `\symref{<symbol>}{<pre>name<post>}`.

`\Sn` For example, given a symbol `agroup` with name `abelian group`, we can do

`\sns` `\symname[pre=Non-,post=s]{agroup}` to produce `Non-abelian groups`.

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

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

`\srefsymuri` turns `<text>` into a link to

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

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

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

7.4 Notations and Semantic Macros

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

introduces a new notation for the referenced symbol.

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

The optional arguments are:

- `prec=<opprec>;<argprec 1>x...x<argprec n>`: An operator precedence and one argument precedence for each argument of the semantic macro. If no argument precedences are given, all argument precedences are equal to the operator precedence. By default, all precedences are 0, unless the symbol takes no argument, in which case the operator precedence is `\neginfprec` (negative infinity).
`prec=nobrackets` is an abbreviation for `prec=\neginfprec;\infprec x...x\infprec`.
- `op=<code>`: An operator notation. If none is given, the notation component marked with `\maincomp` is used. If no `\maincomp` occurs in the notation, the default operator notation is `\symname{<symbol>}`.
- `variant=<id>`: An id for this notation. The key `variant=` can be omitted; i.e. `\notation[foo]` is equivalent to `\notation[variant=foo]`.

`\comp` `\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 **TeX** users, like us), and might want to give those another thought - quite likely, the concept you aim to implement does not actually represent a semantically meaningful (mathematical) concept, and you will want to use `\def` and similar native **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` `\neginfprec` and `\neginfprec` represent *infinitely large* and *infinitely small* **precedences**, respectively.



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

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

Example 25

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

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

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

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

`\dobracket` `\dobracket{\langle code \rangle}` wraps parentheses around `\langle code \rangle`.

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

7.4.2 Notations for Argument Sequences

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

`\argsep` `\argsep{\langle parameter token \rangle}{\langle separator \rangle}`

takes the elements of the argument sequence in position `\langle parameter token \rangle` and separates them by `\langle separator \rangle`.

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{\langle parameter token \rangle}{\langle code \rangle}{\langle separator \rangle}`

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

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

`\argarraymap` **TODO**¹²

7.4.3 Semantic Macros

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

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

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

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

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

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

\arg The `\arg` macro can be used to explicitly mark the arguments of a `semantic macro` in text mode.

By default, they are numbered consecutively; e.g. `\smacro{... \arg{a}... \arg{b}}` determines `a` and `b` to be the (first and second) arguments.

The starred variant `\arg*` allows for marking up the arguments, but does not produce any output. This can be used to provide arguments that are not mentioned in the text we want to mark up, because they are implicitly obvious or mentioned elsewhere.

If we want to change the order of the arguments, we can provide the precise argument number as an optional argument; e.g. `\smacro{... \arg[2]{a}... \arg[1]{b}}` determines `b` to be the first and `a` to be the second argument.

An argument number may be used repeatedly, if the corresponding `argument mode` is `a` or `B`.

Applications of `semantic macros` with arguments are translated to $\textcolor{blue}{\text{M}} \rightarrow \textcolor{teal}{\text{MMT/OMDOC}}$ as OMA-terms with head `<OMS name="<symbol>" />`, or $\textcolor{green}{\text{M}} \rightarrow \textcolor{teal}{\text{<OMBIND name="<symbol>" />}}$, 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{<module>}` is allowed anywhere and makes the `module`'s contents available up to the current `TeX` group. This is the right `macro` to use outside of `modules`, or when none of its contents use any of the used `module`'s `symbols` directly (e.g. in `types` or `definientia`).

\requiremodule `\requiremodule{<module>}` is only allowed in `modules` and makes the required `module`'s contents available within the current `module`. The imported `symbols` can be safely used in `types` and `definientia`, but not in the `return` code of `symbols`, and the imported content is not exported further – i.e. using the current `module` does not also open the required `module`.

`\importmodule` `\importmodule{\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 In `MMT`, every *document* and every `module` induces an `MMT` theory. `\usemodule`
 \rightarrow induces and `MMT` `include` in the document *theory*, `\importmodule` and
 \rightsquigarrow `\requiremodule` both induce an `include` in the `module`'s *theory*.

It is worth going into some detail how exactly `\usemodule`, `\importmodule` and `\requiremodule` resolve their arguments to find the desired `module` – which is closely related to the *namespace* generated for a `module`, that is used to generate its `MMT-URI`.

Ideally, `gTeX` 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 `gTeX` can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:



- If `\begin{smodule}{Foo}` occurs in a file `/path/to/file/Foo[.<lang>].tex` which does not belong to an `math archive`, the namespace is `file://path/to/file`.
- If the same statement occurs in a file `/path/to/file/bar[.<lang>].tex`, the namespace is `file://path/to/file/bar`.

In other words: outside of `math archives`, the namespace corresponds to the file URI with the filename dropped iff it is equal to the `module` name, and ignoring the (optional) language suffix.

If the current file is in an `archive`, the procedure is the same except that the initial segment of the file path up to the `archive`'s `source` directory is replaced by the `archive`'s namespace URI.

Conversely, here is how namespaces/URIs and file paths are computed in import statements, exemplary `\importmodule`:



- `\importmodule{Foo}` outside of an `archive` refers to `module Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same file or, if not, in a file `Foo[.<lang>].tex` in the same directory.
- The same statement *within* an `archive` refers to either the `module Foo` declared earlier in the same file, or otherwise to the `module Foo` in the `archive`'s top-level namespace. In the latter case, it has to be declared in a file `Foo[.<lang>].tex` directly in the `archive`'s `source` directory.
- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory



and namespace outside of an `archive`, or relative to the current `archive`'s top-level namespace and `source` directory, respectively.

The `module` `Foo` must either be declared in the file `<top-directory>/some/path/Foo[.<lang>].tex`, or in `<top-directory>/some/path[.<lang>].tex` (which are checked in that order).

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the `archive` `Some/Archive` in the MathHub directory.

7.6 Variables and Sequences

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

Takes the same arguments as `\symdef`, but produces a `variable` rather than a `symbol`. `Variables` definitions are always local to the current `TeX` group and are allowed anywhere (i.e. outside of `modules`).

`<options>` may include the additional keyword `bind`, in which case the `variable` will be appropriately abstracted away in statements (see also `\varbind`).

Unlike `\symdef`, there is no starred variant `\vardef*` – `variables` always generate a `semantic macro`.

The `semantic macro` for a `variable` behaves analogously to that of a `symbol`.

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

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

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

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

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

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

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

`\varseq` `\varseq{⟨mname⟩}[⟨options⟩]{⟨range⟩}{⟨notation⟩}`

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

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

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

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

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

`\seqmap` `\seqmap{⟨code⟩}{⟨sequence⟩}`

Maps the function `⟨code⟩` (containing `#1`) over every element of the `⟨sequence⟩`.

Is allowed anywhere `gTeX` expects an `argument mode` a or B argument.

7.7 Structures

`Mathematical structure` bundle interdependent `symbols` together.

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

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

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

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

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

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

\hookrightarrow `\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 26

```

\begin{mathstructure}{\struct}
  \symdef{fieldda}{a}
  \symdef{fielddb}[args=2]{#1 \maincomp{b} #2}
  \symdef{fielddc}[args=2,def=\sn{fielddb}]{#1 \maincomp{c} #2}
  \inlineass[name=axiom1]{\conclusion{some axiom}}
\end{mathstructure}
\notation{\struct}{\StRuCt}

```

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

More precisely: $\mathcal{\struct}\{\langle code \rangle\}$ 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 $\mathcal{\struct}$ does when using `\struct` in the *return* of a *symbol* (or *variable*).

Continuing:

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

Chapter 8

Statements

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

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

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

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

`sexample (env.)` The `sexample environment` represents examples, the `for=` attribute specifies the concept this is an example for. For `counterexamples` use `style=counterexample`

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

All of these take the same arguments:

- `for=<cs1>`: a comma-separated list of `symbols`.
- The same optional arguments as `\symdecl`, with `macro=` replacing the name of the `semantic macro`. All of them are only relevant, if either `name=` or `macro=` are provided.

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

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

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

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

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

<code>\varbind</code>	<code>\varbind{<cls>}</code>
-----------------------	------------------------------------

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:

<code>\definiendum</code>	The <code>\definiendum</code> macro behaves largely like <code>\symref</code> , but it uses a dedicated highlighting for <i>definienda</i> and adds the referenced <code>symbol</code> to the <code>for=</code> list of the <code>environment</code> .
<code>\definame</code>	
<code>\Definame</code>	
<code>\defnotation</code>	

`\definame` is to `\definiendum` as `\symname` is to `\symref`. Analogously, `\Definame` behaves like `\Symname`.

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

<code>\definiens</code>	The <code>\definiens</code> macro can be used to semantically annotate the <i>definiens</i> in a <code>sdefinition</code> .
-------------------------	---

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

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

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

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

$\backslash\text{premise}$ $\backslash\text{conclusion}$	<p>The $\backslash\text{conclusion}$ macro can be used to mark up the actual statement within an sassertion. The $\backslash\text{premise}$ macro can be used to additionally mark up <i>premises</i>.</p>
---	--

If the sassertion environment has several elements in its `for` list, an optional argument $\backslash\text{conclusion}[\langle\text{symbol}\rangle]\{\dots\}$ can be used to tell \LaTeX 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 $\backslash\text{conclusion}$ and $\backslash\text{premise}$:

Firstly, it will attempt to translate the contents of $\backslash\text{conclusion}$ into an MMT/OMDOC term c . This succeeds easily if the $\backslash\text{conclusion}$ is some *semantic macro* (applied to arguments).

Secondly, it will collect all fragments marked up with $\backslash\text{premise}$ 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 \LaTeX 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 $\backslash\text{definiens}$ occurrence.

$\text{sproof}(\text{env.})$

TODO¹³

¹³TODO: proofs

Chapter 9

Customizing Typesetting

There are two kinds of typesetting that can be customized in \LaTeX : **symbol** references (notation components, $\text{\textbackslash symref}$, 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 $\text{\textbackslash inputref}$), 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 0.3 (The `lib`-Directory) in the \LaTeX Manual)

9.1 Highlighting Symbol References

 $\text{\textbackslash symrefemph}$
 $\text{\textbackslash symrefemph@uri}$

$\text{\textbackslash symrefemph}$ governs how references via $\text{\textbackslash symref}$ (or $\text{\textbackslash symname}$, or their short variants) are highlighted;

Doing $\text{\textbackslash symref}\{\langle symbol \rangle\}\{\langle text \rangle\}$ ultimately expands to $\text{\textbackslash symrefemph@uri}\{\langle text \rangle\}\{\langle symbol URI \rangle\}$, the default implementation of which is just $\text{\textbackslash symrefemph}\{\langle text \rangle\}$. The default implementation of $\text{\textbackslash symrefemph}$, in turn, is just $\text{\textbackslash emph}\{\langle text \rangle\}$.

If you only want to change e.g. the color of $\text{\textbackslash symrefs}$, you only need to redefine $\text{\textbackslash symrefemph}$, 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
}

```

<code>\compemph</code> <code>\compemph@uri</code>	Like <code>\symrefemph</code> and <code>\symrefemph@uri</code> , but governs the highlighting of components (marked with <code>\comp</code> or <code>\maincomp</code>) in <i>notations</i> .
--	---

<code>\defemph</code> <code>\defemph@uri</code>	Like <code>\symrefemph</code> and <code>\symrefemph@uri</code> , but governs the highlighting of definienda marked with <code>\definiendum</code> (or <code>\definame</code>).
--	---

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

9.2 Styling Environments and Macros

A variety of *environments* and *macros* provided by \TeX are *stylable* using the *macros* `\stexstyle<name>[<style>]{...}`. These stylable *environments* and *macros* bind various of their parameters to *macros* `\this<parameter>`, which can then be used in the styles.

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

```
\stexstyledefinition{\begin{definition}}{\end{definition}}
```

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

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

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

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

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

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

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

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

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

TODO¹⁴

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

We can achieve that by doing, for example:

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

in which case

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

will yield

Symbol foo (with arity ii) of type N.

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

TODO¹⁵

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 \LaTeX 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:

<code>slides</code> <code>notes</code>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see subsection 4.1.3).
---	--

<code>sectocframes</code>	If the option <code>sectocframes</code> is given, then for the <code>sfragments</code> , special frames with the <code>sfragment</code> title (and number) are generated.
---------------------------	---

<code>frameimages</code> <code>fiboxed</code>	If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code> -generated frames (see ???). If also the <code>fiboxed</code> 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 **\notestru**e or **\notesfalse** is strongly discouraged however.



We need to give the title frame the **noframenumbering** option so that the frame numbering is kept in sync between the slides and the course notes.



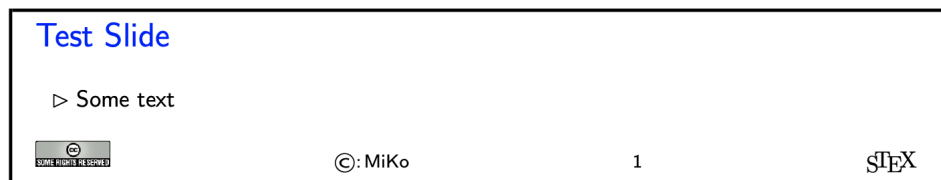
The **beamer** class recommends not to use the **allowframebreaks** option on frames (even though it is very convenient). This holds even more in the **notesslides** case: At least in conjunction with **\newpage**, frame numbering behaves funnily (we have tried to fix this, but who knows).

\inputref* If we want to transclude a the contents of a file as a note, we can use a new variant **\inputref*** of the **\inputref** macro: **\inputref*{foo}** is equivalent to **\begin{note}\inputref{foo}\end{note}**.

nparagraph (*env.*) There are some environments that tend to occur at the top-level of **note** environments. We make convenience versions of these: e.g. the **nparagraph** environment is just an **nparagraph** (*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{<logo name>}**.

\setsource The default footer line of the **notesslides** package mentions copyright and licensing. In **notesslides** **\source** stores the author's name as the copyright holder. By default it is the author's name as defined in the **\author** macro in the preamble. **\setsource{<name>}** can change the writer's name.

\setlicensing For licensing, we use the Creative Commons Attribution-ShareAlike license by default to strengthen the public domain. If package **hyperref** is loaded, then we can attach a hyperlink to the license logo. **\setlicensing[<url>]{<logo name>}** is used for customization, where **<url>** is optional.

10.1.5 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add **sTeX** notes.

<code>\frameimage</code> <code>\mhframeimage</code>	<p>In this case we can use <code>\frameimage[⟨opt⟩]{⟨path⟩}</code>, where <code>⟨opt⟩</code> are the options of <code>\includegraphics</code> from the <code>graphicx</code> package [CarRah:tpp99] and <code>⟨path⟩</code> is the file path (extension can be left off like in <code>\includegraphics</code>). We have added the <code>label</code> key that allows to give a frame label that can be referenced like a regular <code>beamer</code> frame.</p>
--	---

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}
```

<code>\textwarning</code>	<p>The <code>\textwarning</code> macro generates a warning sign: </p>
---------------------------	---

10.1.6 Ending Documents Prematurely

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

`\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 `TeX` preamble of the course notes file.

10.1.7 Global Document Variables

To make document fragments more reusable, we sometimes want to make the content depend on the context. We use **document variables** for that.

<code>\setSGvar</code> <code>\useSGvar</code>	<code>\setSGvar{<vname>}{<text>}</code> to set the global variable <code><vname></code> to <code><text></code> and <code>\useSGvar{<vname>}</code> to reference it.
--	---

<code>\ifSGvar</code>	With <code>\ifSGvar</code> we can test for the contents of a global variable: the macro call <code>\ifSGvar{<vname>}{<val>}{<ctext>}</code> tests the content of the global variable <code><vname></code> , only if (after expansion) it is equal to <code><val></code> , the conditional text <code><ctext></code> 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.

<code>\excursion</code>	The <code>\excursion{<ref>}{<path>}{<text>}</code> 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}

```

<code>\activateexcursion</code> <code>\printexcursion</code> <code>\excursionref</code>	Here <code>\activateexcursion{<path>}</code> augments the <code>\printexcursions</code> macro by a call <code>\inputref{<path>}</code> . In this way, the <code>\printexcursions</code> 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.

<code>\excursiongroup</code>	Finally, we usually want to put the excursions into an <code>sfragment</code> environment and add an introduction, therefore we provide the a variant of the <code>\printexcursions</code> macro: <code>\excursiongroup[id=<id>,intro=<path>]</code> 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

<hr/> <div style="display: flex; flex-direction: column; gap: 2px;"><div><code>solutions</code></div><div><code>notes</code></div><div><code>hints</code></div><div><code>gnotes</code></div><div><code>pts</code></div><div><code>min</code></div><div><code>boxed</code></div><div><code>test</code></div></div> <hr/>	<p>The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code> (should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?), <code>min</code> (do we display the estimated minutes for problem solving). If theses are specified, then the corresponding auxiliary parts of the problems are output, otherwise, they remain invisible.</p>
--	---

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

Example 27

Input:

```

1 \documentclass{article}
2 \usepackage[solutions,hints,pts,min]{problem}
3 \begin{document}
4   \begin{sproblem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
5     How many Elephants 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 Elephants)
 How many Elephants can you fit into a Volkswagen beetle?

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

`solution (env.)` The `solution` environment can be used to specify a solution to a problem. If the package option `solutions` is set or `\solutionstrue` is set in the text, then the solution will be presented in the output. The `solution` environment takes an optional KeyVal argument with the keys `id` for an identifier that can be reference `for` to specify which problem this is a solution for, and `height` that allows to specify the amount of space to be left in test situations (i.e. if the `test` option is set in the `\usepackage` statement).

`hint (env.)` The `hint` and `exnote` environments can be used in a `problem` environment to give hints and to make notes that elaborate certain aspects of the problem. The `gnote` (grading notes) environment can be used to document situations that may arise in grading.

`\startsolutions`
`\stopsolutions`

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

`\ifsolutions`

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

10.2.3 Markup for Added-Value Services

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

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

Multiple Choice Blocks

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

`\mcc` `\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 28

Input:

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

Output:

Exercise (Functions)

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

- ☒ def
- ☐ function^a
- ☐ fun^b
- ☐ public static void^c

^a

that is for C and C++

^b

that is for Standard ML

^cNoooooooooo

that is for Java

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

Example 29

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 30

Input:

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

Output:

Exercise (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle?

Example 31

Input:

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

Output:

Exercise (Fitting Elephants)

How many Elephants 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.

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

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

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

10.2.5 Testing and Spacing

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

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

10.3 HWExam Manual

10.3.1 Introduction

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

10.3.2 Package Options

<code>solutions</code>	The <code>hwexam</code> package and class take the options <code>solutions</code> , <code>notes</code> , <code>hints</code> , <code>gnotes</code> , <code>pts</code> , <code>min</code> , and <code>boxed</code> that are just passed on to the <code>problems</code> package (cf. its documentation for a description of the intended behavior).
<code>notes</code>	
<code>hints</code>	
<code>gnotes</code>	
<code>pts</code>	
<code>min</code>	
<code>multiple</code>	Furthermore, the <code>hwexam</code> package takes the option <code>multiple</code> that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).
<code>test</code>	Finally, there is the option <code>test</code> that modifies the behavior to facilitate formatting tests. Only in <code>test</code> mode, the macros <code>\testspace</code> , <code>\testnewpage</code> , and <code>\testemptypage</code>

have an effect: they generate space for the students to solve the given problems. Thus they can be left in the 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 the assignment is due).

10.3.4 Including Assignments

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

10.3.5 Typesetting Exams

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

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

Will result in

Name: Matriculation Number:

320101 General Computer Science (Fall 2010)
2026-06-25

You have one hour (sharp) for the test;
Write the solutions to the sheet.
The estimated time for solving this exam is 23 minutes, leaving you 37 minutes for revising your exam.
You can reach 23 points if you solve all problems. You will only need 27 points for a perfect score, i.e. -4 points are bonus points.

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

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

good luck

¹⁷ Local Variables: mode: latex TeX-master: ../stex-manual End:
LocalWords: hwexam solutions,notes,hints,gnotes,pts,min gnotes testemptypage re-
qpts LocalWords: inputassignment reqpts hour,min 60,reqpts

10.4 Tikzinput Manual

image The behavior of the `ikzinput` package is determined by whether the `image` option is given. If it is not, then the `tikz` package is loaded, all other options are passed on to it and `\tikzinput{<file>}` inputs the TIKZ file `<file>.tex`; if not, only the `graphicx` package is loaded and `\tikzinput{<file>}` loads an image file `<file>.<ext>` generated from `<file>.tex`.
The selective input functionality of the `tikzinput` package assumes that the TIKZ pictures are externalized into a standalone picture file, such as the following one

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


```

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

```

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

\tikzinput \ctikzinput	This is exactly where the <code>tikzinput</code> package comes in: it supplies the <code>\tikzinput</code> macro, which – depending on the <code>image</code> 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[$\langle opt \rangle$]{ $\langle file \rangle$ }` disregards the optional argument $\langle opt \rangle$ and inputs $\langle file \rangle.tex$ via `\input` and resizes it to as specified in the `width` and `height` keys. If it is, `\tikzinput[$\langle opt \rangle$]{ $\langle file \rangle$ }` expands to `\includegraphics[$\langle opt \rangle$]{ $\langle file \rangle$ }`.

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

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

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

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

Part III
Documentation

Chapter 11

Utilities

Various utility methods

`\stex_ignore_spaces_and_pars:`

Ignores space characters and empty lines (which would close the current paragraph)

`\stex_undefine:N` Locally undefines a macro
`\stex_undefine:c`

`\stex_str_if_starts_with_p:nn` * `\stex_str_if_starts_with:nn` `{<str1>}` `{<str2>}`
`\stex_str_if_starts_with:nnTF` *

Tests if `<str1>` starts with `<str2>`.

`\stex_str_if_ends_with_p:nn` * `\stex_str_if_ends_with:nn` `{<str1>}` `{<str2>}`
`\stex_str_if_ends_with:nnTF` *

Tests if `<str1>` ends with `<str2>`.

`\stex_deactivate_macro:Nn` `\stex_deactivate_macro:Nn` `\macro` `{<scope>}`
`\stex_reactivate_macro:N`

redefines `\macro` to throw an error, that the `\macro` is only allowed in `<scope>`. This allows for more informative error messages than `undefined control sequence`.

`\stex_reactivate_macro:N` makes the macro valid again.

11.1 kpsewhich and Environment Variables

`\stex_kpsewhich:Nn` `\stex_kpsewhich:Nn` `\macro` `{<args>}`

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

T_EXhackers note: (Usually) does not require `shell-escape`, since by default, `kpsewhich` is in the list of “allowed” commands.

<code>\stex_get_env:Nn</code>	<code>\stex_get_env:Nn \macro {<envvar>}</code>
-------------------------------	---

Stores the value of the environment variable `<envvar>` in `\macro`.

11.2 Logging

<code>\stex_debug:nn</code>	<code>\stex_debug:nn {<prefix>} {<msg>}</code>
-----------------------------	--

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.

11.3 File Paths

Since `STEX` needs to handle files from various *math archives* in a user-file-system-independent manner in arbitrary *MathHub* directories, we primarily use *absolute* file paths, for operations such as `\inputref`, `\usemodule`, `\importmodule`, etc.

We therefore provide macros to explicitly handle file paths independent of the underlying operating system, resolving and canonicalizing `..` segments and relative paths, etc.

<code>\stex_file_set:Nn</code>	<code>\stex_file_set:Nn \macro {<string>}</code>
--------------------------------	--

<code>\stex_file_set:(No Ne)</code>	represents an already canonicalized file path string as a <code>L^AT_EX3</code> sequence and stores it in <code>\macro</code> .
-------------------------------------	---

<code>\stex_file_resolve:Nn</code>	<code>\stex_file_resolve:Nn \macro {<string>}</code>
------------------------------------	--

<code>\stex_file_resolve:(No Ne)</code>	resolves and canonicalizes the file path string <code><string></code> and stores the result in <code>\macro</code> .
---	--

<code>\stex_file_use:N</code>	★ expands to a string representation of the given file path (using <code>/</code> as separator, regardless of file system).
-------------------------------	---

<code>\stex_file_split_off_ext:NN</code>	<code>\stex_file_split_off_ext:NN \target \source</code>
--	--

splits off the file extension of `\source` and stores the resulting file path in `\target`.

<code>\stex_file_split_off_lang:NN</code>	<code>\stex_file_split_off_lang:NN \target \source</code>
---	---

splits off *both* the file extension *and* language component of `\source` and stores the resulting file path in `\target`; e.g. if `\source` is `foo.en.tex`, `\target` will be `foo`. Explicitly checks that the segment before the file extension is a valid language identifier; otherwise, it will not be stripped.

<code>\c_stex_pwd_file</code>	represent the <i>parent working directory</i> and absolute path to the top-level <code>.tex</code> file currently being processed (as absolute, canonicalized paths)
-------------------------------	--

`\c_stex_home_file` represents the absolute path of the user’s home directory (as absolute, canonicalized path)

`\g_stex_current_file` the current file (as absolute, canonicalized path)

`\stex_input_with_hooks:Nn` `\stex_input_with_hooks:Nn \document-URI {<file string>}`

`\stex_input_with_hooks:Ne` Inputs the given file (as a string) by passing it on to `\input`, setting `\g_stex_current_file`, `\l_stex_current_language_str` and `\l_stex_current_document_uri`, and reverting them again afterwards

`\stex_with_file_hooks:Nnn` `\stex_with_file_hooks:Nnn \document-URI {<file string>} {<code>}`

sets `\g_stex_current_file`, `\l_stex_current_language_str` and `\l_stex_current_document_uri`, executes `<code>`, and reverts them again afterwards

`\stex_if_file_absolute_p:N` *

`\stex_if_file_absolute:N \underline{T} \underline{F}` *

tests whether the given file path (represented as a canonicalized L^AT_EX3 sequence) is absolute.

`\stex_if_file_starts_with:NN \underline{T} \underline{F}` `\stex_if_file_starts_with:NN \child \parent`

tests whether the file path `\child` is a child of `\parent`.

11.4 Group-like Behaviours

Macros for mimicking the behaviour of T_EX groups without actually opening a T_EX group. This is relevant for two purposes:

- Temporarily redefining a macro, doing a bunch of stuff, and then reverting it to its previous definition; that is what a “*pseudogroup*” does.
- Defining macros at a specific group level further up. That is what a “*metagroup*” does – code wrapped in `\stex_metagroup_do_in:n` will be repeatedly executed in `\aftergroup` calls, up to the group level of the innermost metagroup. That allows for e.g. declaring new symbols in `sdefinition` paragraphs that are valid in the entire `mathstructure` or module.

`\stex_pseudogroup:nn` `\stex_pseudogroup:nn {<code>} {<reset code>}`

will *expand* `<reset code>` first, then process `<code>`, then process the expanded `<reset code>`.

`\stex_pseudogroup_restore:N` ★ `\stex_pseudogroup_restore:N` `\macro`

will expand to code that will redefine `\macro` as its current definition. Only works for *token lists* and similar macros that do not take arguments.

In combination, that allows for things like

Example 32

```
1 % \stex_pseudogroup:nn{
2 % \def \foo {..}
3 % ...
4 % }\stex_pseudogroup_restore:N \foo}
5 %
```

`\stex_pseudogroup_with:nn` `\stex_pseudogroup_with:nn` `{\macro-list}` `{\code}`

will store the definitions of the macros in `\macro-list`, execute `\code`, and then revert the macros to their previous definitions without opening a \TeX group. **TODO: this might be unnecessary? Check/profile**

`\stex_metagroup_new:` opens a new *metagroup*. All code executed in `\stex_metagroup_do_in:n` will survive up to the current group level. Metagroups can be nested.

`\stex_metagroup_do_in:n` Executes the given code in the current metagroup. If there is no metagroup currently, the code will simply be executed once, so there is little danger in calling this spuriously.

\TeX hackers note: This repeatedly stores the provided code in a macro and uses `\aftergroup` until the group level of the current metagroup is reached.

11.5 Key Handling

Key handling mechanisms on top of `l3keys` for inheriting key sets and initializing the associated macros

`\stex_keys_define:nnnn` `\stex_keys_define:nnnn` `{\id}` `{\init}` `{\spec}` `{\exts}`

defined the key set `\id` with the `l3keys`-style key-parsing code `{\spec}`; `\init` is called every time before parsing and is intended to e.g. clear the relevant macros. `\exts` may be a comma-separated list of key set ids from which this key set inherits.

`\stex_keys_set:nn` Like `\keys_set:nn`, but additionally calling the relevant initialization code.

11.6 Languages

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

Two inverse property lists; `\c_stex_languages_prop` maps language abbreviations (e.g. `en`, `de`) to their (babel) names (e.g. `english`, `ngerman`), `\c_stex_language_abbrevs_prop` does the inverse.

`\l_stex_current_language_str`

The language of the current file; may change, if a file with a different language is imported.

11.7 Styling

`\stex_new_stylable_cmd:nnnn` `\stex_new_stylable_cmd:nnnn {<macroname>} {<argument spec>} {<definition>}`
`\stex_style_apply:` `{<default>}`

Defines a new stylable command `\macroname` with `<argument spec>` (passed on to `\NewDocumentCommand`) with `<definition>` as its macro body and default typesetting style `<default>`.

`<default>` may be empty. `<definition>` should at some point use `\stex_style_apply:` to do the typesetting according to the chosen style for this macro, which by default will be defined as `<default>`.

Also generates the macro `\stexstyle<macroname>[<name>]{<code>}` to define a new typesetting style for `\macroname`.

`\stex_new_stylable_env:nnnnnn` `\stex_new_stylable_env:nnnnnn {<environment name>} {<argument spec>} {<begin code>} {<end code>} {<default begin>} {<default end>} {<prefix>}`

Defines a new stylable environment `<prefix><environment name>` with `<argument spec>` (passed on to `\NewDocumentEnvironment`) with `<begin code>` and `<end code>` as its macro body and default typesetting style dictated by `<default begin>` and `<default end>`.

`<begin code>` and `<end code>` should at some point use `\stex_style_apply:` to do the typesetting according to the chosen style for this environment.

Also generates the macro `\stexstyle<environmentname>[<name>]{<begin>}{<end>}` to define a new typesetting style for `environment name`.

A prefix `s` is e.g. used to generate the environment `sparagraph` and the macro `\stexstyleparagraph`.

11.8 Persisting Dependencies

`\stex_persist:n`
`\stex_persist:e`
`\loadsms`

Writes the given code in the `.sms`-file (if `smsmode` is on)

Chapter 12

HTML Output

Macros for dealing with HTML output.

`\stex@backend` Which engine is running; possible values are `pdflatex`, `rustex`, `tex4ht` (**TODO**) and `latexml` (**TODO**).

`\stex_if_html_backend_p: *` L^AT_EX3 conditional testing whether the engine running compiles to HTML.
`\stex_if_html_backend:TF *`

`\ifstexhtml *` L^AT_EX2 conditional testing whether the engine running compiles to HTML.

`\stex_if_do_html_p: *` Whether to (locally) produce HTML output. May be turned off locally even if the back-
`\stex_if_do_html:TF *` end produces HTML; e.g. when parsing dependencies in `\importmodule` or `\usemodule`.

`\stex_suppress_html:n` temporarily suppresses HTML output when processing the provided code.

`\stex_annotate:nn` `\stex_annotate:nn {<keyvals>} {code}`
 attaches the provided comma-separated key-value-pairs (as `key=val`) as HTML attributes to the HTML node(s) generated from `<code>`. If `<code>` results in multiple nodes, this needs to insert a new `<div>`, `` or `<mrow>` node for the attributes. Multiple nested calls to `\stex_annotate:nn` may be merged into a single node.

`stex_annotate_env (env.)` like `\stex_annotate:nn`, but as an environment; i.e. `\begin{stex_annotate_env}{<keyvals>}{<code>}` is equivalent to `\stex_annotate:nn{<keyvals>}{<code>}`

`\stex_html_node:nnn` `\stex_html_node:nnn {<tag>} {<keyvals>} {code}`
 like `\stex_annotate:nn`, but makes sure to wrap `<code>` in a new `<tag>` node.

`stex_env_node (env.)` like `\stex_html_node:nnn`, but as an environment; i.e. `\begin{stex_env_node}{<tag>}{<keyvals>}{<code>}`

is equivalent to `\stex_html_node:nnn{<tag>}{<keyvals>}{<code>}`

`\stex_annotate_invisible:nn` `\stex_annotate_invisible:nn {<keyvals>} code`
`\stex_annotate_invisible:n`

like `\stex_annotate:nn`, but additionally adds the key values `data-ftml-invisible="true"` and `style="display:none;"`. `\stex_annotate_invisible:n` does not take additional attribute-value-pairs.

`\STEXinvisible` public wrapper for `\stex_annotate_invisible:n`.

`\stex_css_link:n` `\stex_css_link:n {<url>}`

inserts a `<link rel="stylesheet" href="<url">` node in the header of the generated HTML.

`\stex_css_literal:n` `\stex_css_literal:n {<text>}`

inserts a `<style><text></style>` node in the header of the generated HTML.

`_stex_annotate_force_break:n`

should guarantee that the HTML nodes generated by the provided code are not merged with any outer nodes with respect to attached attributes.

`\mmlintent` `\mmlintent {<value>} {<code>}`

`\mmlarg` annotates `<code>` with the provided MathML *intent* attribute (or *intent arg* attribute, respectively).

and style patching:

```

1 <@@=stex_styles>
2 \stex_if_html_backend:TF{
3   \cs_new_protected:Nn \stex_style_apply: {}
4   \cs_new_protected:Nn \_stex_styles_patch:nnnn {}
5   \stex_keys_define:nnnn{ csspatch }{
6     \str_clear:N \l_stex_keys_cls_id
7     \str_clear:N \l_stex_keys_counter_id
8     \str_clear:N \l_stex_keys_counter_parent
9   }{
10    counter      .str_set_x:N = \l_stex_keys_counter_id,
11    parent       .str_set_x:N = \l_stex_keys_counter_parent,
12    unknown      .code:n      = {
13      \exp_args:NNo \str_set:Nn \l_stex_keys_cls_id {\l_keys_key_tl}
14    }
15  }{}
16  \cs_new_protected:Npn \_stex_styles_css_patch:nnnn #1 #2 {
17    \stex_keys_set:nn{ csspatch }{ #2 }
18    \group_begin:
19    \catcode'\ =12\relax
20    \catcode'\^J=12\relax
21    \_stex_styles_patch_i:nnn {#1}
22  }
```

```

23
24 \seq_new:N \g__stex_styles_counters_seq
25
26 \cs_new:Nn \__stex_styles_patch_html_c: {
27   \stex_html_literal:n{
28     <span~data-ftml-counter="\l_stex_keys_counter_id"~style="display:none;"~
29     data-ftml-counter-parent="\l_stex_keys_counter_parent"
30     ></span>
31   }
32 }
33
34 \cs_new:Nn \__stex_styles_patch_html: {
35   \stex_html_literal:n{
36     <span~data-ftml-style="\l_stex_keys_cls_id"~style="display:none;"~
37     data-ftml-counter="\l_stex_keys_counter_id"
38     ></span>
39   }
40 }
41
42 \cs_new_protected:Nn \__stex_styles_patch_i:nnn {
43   \str_if_empty:NTF \l_stex_keys_cls_id {
44     \str_set:Nn \l_stex_keys_cls_id { #1 }
45   }{
46     \str_set:Ne \l_stex_keys_cls_id { #1-\l_stex_keys_cls_id }
47   }
48   \exp_args:No \str_case:nnF \l_stex_keys_counter_parent {
49     {}{}
50     {part}{\str_set:Nn \l_stex_keys_counter_parent { 0 }}
51     {chapter}{\str_set:Nn \l_stex_keys_counter_parent { 1 }}
52     {section}{\str_set:Nn \l_stex_keys_counter_parent { 2 }}
53     {subsection}{\str_set:Nn \l_stex_keys_counter_parent { 3 }}
54     {subsubsection}{\str_set:Nn \l_stex_keys_counter_parent { 4 }}
55     {paragraph}{\str_set:Nn \l_stex_keys_counter_parent { 5 }}
56     {subparagraph}{\str_set:Nn \l_stex_keys_counter_parent { 6 }}
57   }{
58     \msg_error:nnx{stex}{error/notasection}\l_stex_keys_counter_parent
59   }
60   \str_set:Ne \l__stex_styles_first_str { &~.ftml-title-paragraph~ { display:inline-block}
61   \str_if_empty:NF \l_stex_keys_counter_id {
62     %\str_put_left:Ne \l__stex_styles_first_str { counter-increment:~ ftml-\l_stex_keys_c
63     \exp_args:NNo \seq_if_in:NnF \g__stex_styles_counters_seq \l_stex_keys_counter_id {
64       \seq_gpush:No \g__stex_styles_counters_seq \l_stex_keys_counter_id
65       \cs_if_eq:ccTF{@onlypreamble}{@notprerr}{
66         \__stex_styles_patch_html_c:
67         % in body
68       }{
69         \exp_args:Ne \AtBeginDocument \__stex_styles_patch_html_c:
70         % in preamble
71       }
72     }
73     \cs_if_eq:ccTF{@onlypreamble}{@notprerr}{
74       \__stex_styles_patch_html:
75       % in body
76     }{

```

```

77     \exp_args:Ne \AtBeginDocument \__stex_styles_patch_html:
78     % in preamble
79   }
80 }
81 \exp_args:Ne \stex_css_literal:n { .ftml-\l_stex_keys_cls_id { \l__stex_styles_first_st
82
83 % TODO export counter reset somehow
84
85 \group_end:
86 }
87 }{
88 \cs_new_protected:Nn \__stex_styles_patch:nnnn {
89   \str_if_empty:nTF {#2}{
90     \tl_set:cn{\__stex_styles_style_#1_start:}{#3}
91     \tl_set:cn{\__stex_styles_style_#1_end:}{#4}
92   }{
93     \tl_set:cn{\__stex_styles_style_#1_#2_start:}{#3}
94     \tl_set:cn{\__stex_styles_style_#1_#2_end:}{#4}
95   }
96 }
97 \cs_new_protected:Nn \__stex_styles_css_patch:nnnn {}
98 }

```

Chapter 13

Math Archives

<div><code>\c_stex_mathhub_files</code></div> <div><code>\mathhub</code></div>	<p>represents the comma-separated, absolute path <i>strings</i> of the user's MathHub directories. If the <code>\mathhub</code> macro is set when the <code>stex</code> package is loaded, this one is used for finding archives. Otherwise it is determined from the <code>MATHHUB</code> environment variable, if set, or from the path in <code><HOME>/stex/mathhub.path</code>, if existent, or set to <code><HOME>/MathHub</code> if all else fails. In all cases, <code>\mathhub</code> will be defined.</p>
--	--

MathHub directories are scanned in order; i.e. earlier paths are prioritized over later ones.

A *math archive* is represented as a triple `{<archive id>}{<base uri>}{<file path string>}`, where an invariant is that `<file path string>` is of the form `<mh>/<archive id>` for some directory `<mh>` in `\c_stex_mathhub_files`.

Such a triple is stored in the macro `\c_stex_mathhub_<id>_archive` when the archives metadata is first loaded.

<div><code>\stex_archive_id:N</code></div>	<p>★ expands to the id of the given archive (a macro defined as a triple as described above)</p>
--	--

<div><code>\stex_archive_base:N</code></div> <div><code>\stex_archive_base:n</code></div>	<p>★ expands to the base URI of the given archive; either given as a macro defined as a triple as described above, or as the archive's id. The latter requires the archive to be loaded beforehand!</p>
---	---

<div><code>\stex_archive_path:N</code></div> <div><code>\stex_archive_path:n</code></div>	<p>★ expands to the file path <i>as a string</i> of the given archive; either given as a macro defined as a triple as described above, or as the archive's id. The latter requires the archive to be loaded beforehand!</p>
---	---

<div><code>\stex_in_archive:nn</code></div>	<div><code>\stex_in_archive:nn {<archive id>} {<code>}</code></div>
---	---

Temporarily sets the current archive `\l_stex_current_archive` to `<archive id>`, executes `<code>` and reverts back to the previous archive. Passes the id of the *now current* archive to `<code>` as **#1**: If the first argument is empty, this will be the current archive's id without changing it. Otherwise, the archive with the desired id will be loaded; throws an error, if it does not exist.

<code>\c_stex_no_archive_str</code>	The archive ID <code>\c_stex_no_archive_str=no/archive</code> is used for documents and other content that is not contained in a <i>math archive</i> . <code>\c_stex_no_archive</code> is the corresponding archive triple.
<code>\c_stex_no_archive</code>	

<code>\c_stex_main_archive</code>	Point to the main file's archive and the <i>current</i> archive, respectively, if existent; undefined otherwise.
<code>\l_stex_current_archive</code>	

<code>\stex_source_path:n</code> *	<code>\stex_source_path:n</code> { <i><relative path></i> }
<code>\stex_source_path:nn</code> *	expands to the absolute file path (string) of the given <i><relative path></i> relative to the source directory of the current archive – or relative to the current file, if we are not in an archive. In the latter case, the file path will <i>not</i> be canonicalized as to remain expandable.

The variant `\stex_source_path:nn` takes the ID of an archive as first argument (instead of the current archive).

Chapter 14

URIs

There are kinds of URIs used in IMMT/OMDoc:

- *Base URIs*, i.e. namespaces,
- *Archive URIs* of the form `<base uri>?a=<archive id>`,
- *Document URIs* of the form `<archive uri>[&p=<path>]&d=<name>&l=<language>`,
- *Document element URIs* of the form `<document uri>&e=<name>`,
- *Module URIs* of the form `<archive uri>[&p=<path>]&m=<name>`, and
- *Declaration URIs* of the form `<module uri>&s=<name>`.

We do not need all of these in `STeX` itself: the *base uri* of any URI is uniquely determined by the archive ID, and we conceptually merge document URIs and document element URIs.

We therefore represent document (element) URIs as 5-tuples `{<archive id>}{<path>}{<name>}{<language>}{<element name>}`, where `<path>` and `<element name>` may be empty; module URIs as triples `{<archive id>}{<path>}{<name>}`, and declaration URIs as 4-tuples `{<archive id>}{<path>}{<module name>}{<name>}`.

`\stex_use_archive_uri:n *` expands to the string representation of the archive URI of the archive with the given ID. Requires, that the archive has been loaded first

14.1 Document URIs

`\stex_use_document_uri:N *` expands to the string representation of the document uri (represented as a macro containing a 5-tuple as above). The variant `\stex_document_uri:n *` expects a 5-tuple directly.

`\stex_document_uri_archive:N *` expands to the archive ID of the document URI

`\stex_document_uri_path:N *`

expands to the path of the document URI (possibly empty)

`\stex_document_uri_name:N *`

expands to the document name of the document URI

`\stex_document_uri_language:N *`

expands to the language of the document URI

`\stex_document_uri_element:N *`

expands to the element name of the document (element) URI (possibly empty)

`\stex_document_uri_with_language:Nn *`

expands to the document URI with the given language

`\stex_new_document_element_uri:n *`

expands to a new document element URI in the current document

`\stex_document_uri_from_archive_file:Nn \stex_uri_from_archive_file:Nn \macro {<relative path>}`

Constructs the document URI of the file *<relative path>* in the current archive and stores it in `\macro`

`\c_stex_main_document_uri`
`\l_stex_current_document_uri`

store the document URIs of the top-level file being processed and the *current* file, respectively.

14.2 Module URIs

`\stex_use_module_uri:N *` expands to the string representation of the module uri (represented as a macro containing
`\stex_use_module_uri:n *` a triple as above). The variant `\stex_module_uri:n` expects a triple directly.

`\stex_module_uri_archive:N *`

expands to the archive ID of the module URI

`\stex_module_uri_path:N *` expands to the path of the module URI (possibly empty)
`\stex_module_uri_path:n *`

`\stex_module_uri_name:N` * expands to the name of the module URI
`\stex_module_uri_name:n` *

`\stex_module_uri_split_name:NNN`
`\stex_module_uri_split_name:NNn`

sets #1 as the parent module of the module URI and #2 as the last name

`\stex_module_uri_as_qm:n` *

`\stex_new_module_uri:n` * `\stex_new_module_uri:n` {*<name>*}

expands to a new module URI triple with the given *<name>* based on the current document URI or module URI, if existent

`\stex_uri_from_pair:Nnn` `\stex_uri_from_pair:Nnn` {*<archive id>*} {*<module path>*}

resolves the pair [*<archive id>*]{*<module path>*} to a module URI and stores the result in #1

`\stex_uri_from_pair:Nnn` like `\stex_uri_from_pair:Nnn`, but takes a token list as argument that may or may not be of the form [archive]module

14.3 Symbol URIs

`\stex_use_symbol_uri:N` * expands to the string representation of the module uri (represented as a macro containing a 4-tuple as above). The variant `\stex_use_symbol_uri:n` expects a 4-tuple directly.
`\stex_use_symbol_uri:n` *

`\stex_new_symbol_uri:n` * `\stex_new_symbol_uri:n` {*<name>*}

expands to a new symbol URI tuple with the given *<name>* based on the current module URI, which is assumed to exist(!).

`\stex_new_symbol_uri:nn` * expands to a new symbol URI tuple with the given *<name>* based on the *given* module URI (either as macro or as tuple), which is assumed to exist(!).

`\stex_symbol_uri_archive:N` *

expands to the archive ID of the symbol URI

`\stex_symbol_uri_path:N` * expands to the path of the symbol URI (possibly empty)

`\stex_symbol_uri_module:N *`
`\stex_symbol_uri_module:n *`

expands to the URI of the containing module of the symbol URI

`\stex_symbol_uri_name:N *` expands to the name of the symbol URI
`\stex_symbol_uri_name:n *`

Chapter 15

Documents

`\STEXftmlink`
`\STEXlinkftml`
`\STEXsetlinkftml`

`\STEXlinkftml` inserts the following link/disclaimer:

*The **FTML** version of this document can be found at*
[http://unknown.source?a=no/archive&p=/home/jazzpirate/work/
Software/sTeX/sTeX/doc&d=stex-doc&l=en](http://unknown.source?a=no/archive&p=/home/jazzpirate/work/Software/sTeX/sTeX/doc&d=stex-doc&l=en)

The precise text can be set using `\STEXsetlinkftml{<text>}`; The link on its own can be inserted using `\STEXftmlink`.

`\stexdoctitle` `\stexdoctitle {<title>}`

Sets `<title>` to be the title of the document. Only the first call of this command does something.

15.1 Sectioning

`\l_stex_current_section_level_int`
`\l_stex_current_section_level_str`

integer 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_current_section_level_int` to the corresponding integer value. `\l_stex_current_section_level_str` stores a string representation of the same value, but will initially contain `document`.

`\currentsectionlevel` will leave the current section level as text in the token input. The variant `\Currentsectionlevel` will produce the capitalized version.

If the `xspace` package is loaded, this will insert an `\xspace` afterwards.

In HTML mode, this will insert an annotation, so it can be adapted dynamically.

`sfragment` (*env.*) A section at the current section level; preferred over the primitives `\section`, `\subsection`, etc., because a) environments are better suited for such structuring, and b) it allows for adapting the inserted section header based on document context, rather than it being hardcoded – e.g. what’s a subsection in one document may be a section in another.

`titlefragment` (*env.*) like `sfragment`, but will use `\maketitle` instead of (one of) the sectioning macros, if no title has been used yet.

`blindfragment` (*env.*) skips one sectioning level in its body so that e.g. subsection 0.1 can be used before the first section

`\skipfragment` Skips one counter at the current sectioning level; e.g. increase the subsection counter to 2 if called immediately after `\section` (or, rather, after `\begin{sfragment}`).

`\setsectionlevel` `\setsectionlevel {<name>}`

Sets the current section level to the one specified (e.g. `\setsectionlevel{subsection}`). Should ideally be called at most once in the preamble of a document.

15.2 Inter-Document References

[id=foo] sets the current fragment's (e.g. Definition 3.1 (Foo)) name to foo, resulting in a DocumentElementURI ..&e=foo. It then delegates to \label{...&e=foo} (which defines \r@...&e=foo), stores ::&e=foo in \g_stex_sref_label_foo, and writes \STeXInternalSRefLabel{foo}{...&e=foo}{definition.3.1}{Foo} to the sref file.

15.3 Inputting From MathHub Resources

\inputref \inputref [*archive id*] {*filepath*}

Inputs the referenced file in the referenced archive (or current archive, if empty) in a tex group, while setting all the relevant macros for the current archive, document URI, etc.

In HTML, will instead just insert an annotation, so that the HTML of the referenced document can be dynamically inserted instead.

\mhinput \inputref [*archive id*] {*filepath*}

Like \inputref, but *always* actually inputs the referenced file (also in HTML mode!) and without opening a tex group.

\ifinputref L^AT_EX2 conditional; is true, if we currently are in an \inputref or \mhinput.

\IfInputref \IfInputref {*true*} {*false*}

Executes *true* if we currently are in an \inputref or \mhinput, otherwise executes *false*.

In HTML, *both* branches are executed(!) and inserted in the HTML with corresponding attributes, so that the relevant parts can be shown or hidden.

\libinput \libinput [*archive id*] {*file name*}

inputs all files *file name*.tex in the lib directories of the current archive. Will throw an error if we are not in an archive or no suitable file is found.

\libusepackage \libusepackage [*package options*] {*package name*}

inputs all packages *package name*.sty in the lib directories of the current archive. Will throw an error if we are not in an archive, or there is not *exactly one* file *package name*.sty somewhere in the lib directories.

Note that unlike \libinput, the optional argument does *not* refer to an archive ID, since this would conflict with package options.

Will give a spurious warning that package mathhub/directory/foo was requested, but package foo was found. This is a side-effect of using absolute file paths, which is necessary to make the package findable in the first place.

\libusetikzlibrary \libusetikzlibrary [*archive id*] {*name*}

like \libusepackage, but for \usetikzlibrary.

<hr/> \addmhbibresource <hr/>	\addmhbibresource [<i>archive id</i>] { <i>file name</i> }
	Calls \addbibresource on all <i>file name</i> .bib files in the lib directories of the current archive. Will throw an error if we are not in an archive or no suitable file is found.
<hr/> \mhgraphics <hr/> \cmhgraphics <hr/>	\mhgraphics adds the archive key to the optional arguments of \includegraphics to allow for using images in math archives (relative to its source directory). \cmhgraphics additionally wraps the image in a \begin{center} . Only defined if the graphicx package is loaded (but may be loaded after the stex package).
<hr/> \lstinputmhlisting <hr/> \clstinputmhlisting <hr/>	like \mhgraphics , but for \lstinputlisting . Only defined if the listings package is loaded (but may be loaded after the stex package).
<hr/> \mhtikzinput <hr/> \cmhtikzinput <hr/>	like \mhgraphics , but for \tikzinput . Only defined if the tikzinput package is loaded (but may be loaded after the stex package).

Chapter 16

Modules

`\l_stex_current_module_uri` stores the URI of the current module, if existent

`\l_stex_all_modules_seq` stores the URIs of all modules *currently in scope*.

`smodule (env.)` Parses the optional arguments, (if relevant) inserts HTML annotations, and then delegates to `\stex_module_setup:n`

`\stex_module_setup:n` Sets up a new module with the given name by checking whether it is a nested module, and if not, has a signature etc. Loads signature and metatheory, if necessary **TODO: deprecate in favor of every document being a module**

`\stex_module_setup_top_nosig:n` Sets up a new top-level module with the given name and no signature. Does not load a meta theory. **TODO: deprecate in favor of every document being a module**

`\stex_close_module:` closes the current module.

`\stex_every_module:n` adds code to be executed every time a new module is opened (e.g. activating certain macros).

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

`\stex_execute_in_module:n` Execute code in the current module (i.e. as if the `\begin{<smodule>}` was the current tex
`\stex_execute_in_module:e` group) and also adds it to the initialization code of the module; i.e. exports all macros defined.

`\stex_if_in_module_p:` * conditional whether we currently are in a module. **TODO: deprecate in favor of every document being a module**
`\stex_if_in_module:` *TF* *

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

conditional whether a module with the given URI exists

`\stex_activate_module:` *N*
`\stex_activate_module:` *n*

`\setmetatheory` `\setmetatheory` [*archive id*] {*module*}

sets the metatheory of all subsequent modules to the specified one

`\STEXexport` `\STEXexport` {*code*}

executes the provided code every time the current module is opened (including immediately). Processes its content in the `\ExplSyntaxOn` category code scheme.

`\stex_structural_feature_module:` *nn*
`\stex_structural_feature_module_end:`

16.1 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 T_EX’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_sms_allow:N` registers the provided macro to be allowed in sms mode.

This only works, if the macro takes no arguments and/or does not touch the subsequent tokens.

`\stex_sms_allow_escape:N` registers the provided macro to be allowed in sms mode.

If the macro is subsequently encountered in sms-mode, parsing is halted and it 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` registers the provided environment 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_sms_allow_import:Nn` `\stex_sms_allow_import:Nn \macro {<code>}`

registers the provided macro to be allowed in the *first pass* in sms mode. The provided `<code>` is executed at the start of the first pass, to define macros accordingly.

`\stex_sms_allow_import_env:nn` `\stex_sms_allow_import_env:nn \envname {<code>}`

registers the provided environment to be allowed in the *first pass* in sms mode. The provided `<code>` is executed at the start of the first pass, to define macros accordingly.

`\g_stex_sms_import_code` inserted between the first and second pass; macros/environments allowed via `\stex_sms_allow_import_*` should store their “results” in here.

`\stex_if_smsmode_p: *` tests for whether we are currently in sms-mode.
`\stex_if_smsmode:TF *`

`\stex_file_in_smsmode:Nn` `\stex_file_in_smsmode:Nn \document-URI {<file string>}`

Loads `{<file string>}` in sms mode, setting `\l_stex_current_document_uri` etc. accordingly and reverting them afterwards.

`\stex_smsmode_do:` should be called in escaped macros or environments allowed in sms mode; switches back to parsing file contents ignoring everything not explicitly allowed in sms mode.
 Does nothing outside of sms mode, so can be called safely.

16.2 Inheritance and Morphisms

```
\stex_require_module:N
\stex_require_module_noerr:N
\stex_require_module_noerr:n
```

makes sure that the module with the given URI is in scope. If not, it will attempt to load the module from disk. `\stex_require_module:N` throws an error if the module can't be found; `\stex_require_module_noerr:N` silently fails.

The `*:n`-variant takes the URI tuple directly rather than a macro.

```
\usemodule \usemodule [<archive ID>] {<module>}
```

loads the specified module without exporting it further.

```
\importmodule \importmodule [<archive ID>] {<module>}
```

loads the specified module and exports it further.

```
\requiremodule \requiremodule [<archive ID>] {<module>}
```

loads the specified module without exporting it further, but generates FTML equivalent to `\importmodule`.

```
\stex_add_morphism:nnnn \stex_add_morphism:nnnn {<name>} {<module URI>} {<kind>} {<contents>}
```

adds a morphism to the current module

16.3 Theory Morphisms

```
\stex_structural_feature_morphism:nnnnn
\stex_structural_feature_morphism_with_macros:nnnnn
\stex_structural_feature_morphism_end:
```

```

#1 : name
#2 : kind
archive ID
#3 : module spec
#4 : additional attributes
```

```
\stex_iterate_morphisms:nn
```

```
\stex_get_in_morphism:n
```

```
copymodule (env.) (and \copymod)
```

```
interpretmodule (env.) (and \interpretmod)
```

\assign

\renamedekl

\assignMorphism

Chapter 17

Symbols

17.1 Declarations

<code>\symdecl</code>	<code>\symdecl[*] {\langle iname \rangle} [\langle options \rangle]</code>
-----------------------	--

takes care of the optional arguments, styling, generates the symbol, defines the relevant macros and adds them to the current module.

<code>\symdef</code>	<code>\symdef {\langle iname \rangle} [\langle options \rangle] {\langle notation \rangle}</code>
----------------------	---

<code>\textsymdecl</code>	
---------------------------	--

<code>\stex_symdecl_do:</code>	does the actual work. Requires all the <code>\l_stex_key_*</code> macros are set.
--------------------------------	---

<code>_stex_symdecl_html:</code>	not namespaced so it can be reused in e.g. <code>sdefinitions</code> that generate new symbols. Requires that the relevant <code>\l_stex_key_*</code> macros and <code>\l_stex_macroname_str</code> are set.
-----------------------------------	--

<code>\stex_add_symbol:nnnnnnN</code>	<code>#1 : {\langle Macro name \rangle}</code> <code>#2 : {\langle Name \rangle}</code> <code>#3 : {\langle arity \rangle}</code> <code>#4 : {\{(\langle Arg num \rangle)\{\langle Arg str \rangle\}}^*}</code> <code>#5 : Definiens</code> <code>#6 : type</code> <code>#7 : Return</code> <code>#8 : Command</code> adds a new symbol to the current module.
---------------------------------------	--

<code>\stex_has_definiens_p:N</code>	<code>*</code>
<code>\stex_has_definiens:N\TF</code>	<code>*</code>
<code>\stex_has_definiens_p:n</code>	<code>*</code>
<code>\stex_has_definiens:n\TF</code>	<code>*</code>

17.2 Retrieval

<code>\stex_iterate_symbols:n</code>	iterates over all symbols currently in scope. The provided code will receive nine arguments: The URI of the containing module and the eight parameters as in <code>\stex_add_symbol:nnnnnnnN</code> .
<code>\stex_iterate_break:</code>	
<code>\stex_iterate_break:n</code>	iteration is stopped in the code using <code>\stex_iterate_break:</code> or <code>\stex_iterate_break:n</code> .

<code>\stex_iterate_symbols:nn</code>	<code>\stex_iterate_symbols:nn {<modules>} {<code>}</code>
---------------------------------------	--

iterates over all symbols in the comma-separated list of module URIs in `<modules>`. The provided code will receive nine arguments: The URI of the containing module and the eight parameters as in `\stex_add_symbol:nnnnnnnN`.

iteration is stopped in the code using `\stex_iterate_break:` or `\stex_iterate_break:n`.

<code>\stex_get_symbol:n</code>	attempts to find the symbol with the given name/id/URI suffix. If not successful, will execute the F code or throw an error. Otherwise, defines the following macros:
<code>\stex_get_symbol:nTF</code>	
<code>\stex_get_symbol:nF</code>	

- `\l_stex_get_symbol_uri,`
- `\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`

17.3 Variables

<code>\l_stex_variables_prop</code>	contains all variables currently in scope.
-------------------------------------	--

`\vardef`

`\newvar`

`\newseq`

`\stex_get_var:n`

17.3.1 Variable Sequences

`\varseq`

17.4 Expressions

`\symuse` retrieves a symbol by name/id and invokes it as if using a semantic macro.

`_stex_next_symbol:n` code to be executed the next time a symbol is invoked.

`_stex_invoke_symbol:NnnnnnN`
`_stex_invoke_symbol:NeooooN`
`_stex_invoke_symbol:nnnnnnN`

invokes the symbol. Takes as arguments the following data, and defines the corresponding macros accordingly:

- `\l_stex_current_symbol_uri`,
- `\l_stex_current_symbol_arity_int`,
- `\l_stex_current_symbol_args_tl`,
- `definiens` (currently not used),
- `\l_stex_current_symbol_type_tl`,
- `\l_stex_current_symbol_return_tl`,
- the invocation macro (is called after setup).

Also defines `\l_stex_current_full_tl` and `\l_stex_current_display_tl`

For a “normal” symbol defined via `\symdecl` or `\symdef`, `\l_stex_current_symbol_invoke_cs` is defined as `\stex_invoke_symbol:.`

Opens a new \TeX group that needs to be closed by `\l_stex_current_symbol_invoke_cs!`

`\stex_invoke_symbol:` Invokes `\l_stex_current_symbol_uri`; assumes all the `\l_stex_current_*`-macros have been defined prior.

`\stex_invoke_text_symbol:` Invokes `\l_stex_current_symbol_uri` as a symbol declared via `\textsymdecl`; assumes all the `\l_stex_current_*`-macros have been defined prior.

`\stex_invoke_sequence:` Invokes `\l_stex_current_symbol_uri` as a sequence variable declared via `\varseq`; assumes all the `\l_stex_current_*`-macros have been defined prior.

`\stex_invoke_structure:` Invokes a mathstructure symbol; assumes all the `\l_stex_current_*`-macros have been defined prior.

`_stex_invoke_variable:nnnnnn`

invokes the variable. Takes as arguments the following data, and defines the corresponding macros accordingly:

- variable name,
- `\l_stex_current_symbol_arity_int`,
- `\l_stex_current_symbol_args_tl`,
- definiens (currently not used),
- `\l_stex_current_symbol_type_tl`,
- `\l_stex_current_symbol_return_tl`,
- the invocation macro (is called after setup).

`\svar`

17.4.1 Term Annotations

`_stex_eat_exclamation_point:`

removes spurious ! characters

`_stex_term_oms:nn` `_stex_term_oms:nn {<notation id>} {<code>}`
 annotates `<code>` with `OMID="\l_stex_current_symbol_uri"`

`_stex_term_omv:nn` `_stex_term_omv:nn {<notation id>} {<code>}`
 annotates `<code>` with `OMV="\l_stex_current_symbol_uri"`
TODO: This shouldn't use `\l_stex_current_symbol_uri`!

`_stex_term_oma:nn` `_stex_term_oma:nn {<notation id>} {<code>}`
 annotates `<code>` with `OMA="\l_stex_current_symbol_uri"`

`_stex_term_omb:nn` `_stex_term_omb:nn {<notation id>} {<code>}`
 annotates `<code>` with `OMBIND="\l_stex_current_symbol_uri"`

17.4.2 Symbol Arguments

`\stex_term_arg:nnnnn` marks an argument of a symbol application, sets the precedence accordingly, sets `\l_stex_allow_semantic_bool` etc.

#1 : argument number
 #2 : argument mode
 #3 : precedence
 #4 : argument name (WiP)
 #5 : code / body

`\stex_term_arg:nnn` `\stex_term_arg:nnn` `{\mode}` `{\num}` `{\code}`

inserts the HTML annotations for the argument

`\stex_term_arg_aB:nnnnn` takes care of sequence arguments

`\seqmap`

17.4.3 Notation components

`\comp`
`\compemph@uri`
`\compemph`

`\varemp@uri`
`\varemp`

`\defemph@uri`
`\defemph`

`\symrefemph@uri`
`\symrefemph`

The following commands do the actual attributes:

`\do_comp:nnNn` `\do_comp:nnNn` `{\key-suffix}` `{\html-value}` `\comp-macro` `{\code}`

annotated `\code` with `data-ftml-{\key-suffix}` using the highlighting dictated by `\comp-macro`.

17.4.4 Symbol References

`\symref`
`\sr`

`\symname`
`\sn`
`\sns`
`\Symname`
`\Sn\Sns`

`\varref`
`\varname`
`\Varname`

`\definiendum`
`\definame`
`\Definame`
`\defnotation`

`\foldexpr`

17.5 Checking

`\stex_if_check_terms_p:` \star conditional whether terms (types and definientia of symbols) should get syntactically
`\stex_if_check_terms:` \underline{TF} \star checked. In HTML mode, this is always false, since they get written to the HTML
 anyway.

`\stex_check_term:` \underline{nn} typesets the given expression in a `\tiny\marginpar`, checking its syntactic validity. Does
 nothing, if term checking is not explicitly activated.

`_stex_check_terms:` checks the type and definiens components of a symbol declaration (i.e. `\stex_key_-`
`type_tl`, `\stex_key_def_tl` and `\stex_key_return_tl!`).

Chapter 18

Notations

`\notation`

`\varnotation`

`\stex_notation_parse:n` parses the body of a notations and saves it in `\l_stex_notation_macrocode_cs`

`_stex_notation_add:` adds the fully parsed notation to the current module

<code>\stex_add_notation:nnnnn</code>	#1 :	URI
<code>\stex_add_notation:ooeoo</code>	#2 :	variant
	#3 :	arity
	#4 :	macro body
	#5 :	op
		adds the notation to the current module

`_stex_map_args:N`
`_stex_map_notation_args:N`

`_stex_var_notation_macro:`

`\setnotation`
`_stex_notation_set_default:n`

<code>\stex_iterate_notations:nn</code>	<code>\stex_iterate_notations:nn {<modules>} {<code>}</code>
---	--

iterates over all notations in the comma-separated list of module URIs and executes `{<code>}` with #1,#2,#3,#4,#5 as the notation parameters.

<code>\stex_use_notation:nnTF</code>	<code>\stex_use_notation:nn {<uri>} {<id>} {<exists code>} {<missing code>}</code>
--------------------------------------	--

<code>\stex_use_op_notation:nnTF</code>	sets <code>\l_stex_notation_cs</code>
---	---------------------------------------

<code>_stex_notation_check:</code>

<code>_stex_notation_do_html:n</code>
--

<code>_stex_notation_make_args:</code>

<code>\stex_do_default_notation:</code>
<code>\stex_do_default_notation_op:</code>

<code>\STEXInternalNotation</code>	#1 : notation id
	#2 : operator precedence
	#3 : intent (WiP)
	#4 : arguments
	#5 : notation code
	#6 : continuation

<code>\argsep</code>

<code>\argmap</code>

<code>\argarraymap</code>

18.1 Automated Bracketing

<code>\infprec</code>	infinite and negative infinite precedences.
-----------------------	---

<code>\neginfprec</code>

<code>_stex_maybe_brackets:nn</code>	<code>_stex_maybe_brackets:nn {<prec>} {<body>}</code>
---------------------------------------	---

inserts parentheses around `<body>` iff precedences and context call for it.

`\dobrackets` actually inserts parentheses around its argument.

`\withbrackets` `\withbrackets` $\{\langle left \rangle\} \{\langle right \rangle\} \{\langle body \rangle\}$
sets $\langle left \rangle$ and $\langle right \rangle$ as the parentheses to use in $\langle body \rangle$.

`\dowithbrackets` `\dowithbrackets` $\{\langle left \rangle\} \{\langle right \rangle\} \{\langle body \rangle\}$
combines `\withbrackets` and `\dobrackets`.

Chapter 19

Structures

```
mathstructure (env.)  
  
extstructure (env.)
```

```
\this
```

```
\stex_get_mathstructure:n sets \l_stex_get_structure_module_uri or throws an error if no structure by the given  
name/id exists.
```

```
\usestructure
```

Chapter 20

Statements

`\stex_do_for_list:` resolves each id in the `for={...}` comma-separated list of ids `\l_stex_key_for_clist`. Stores the full resolved URIs in `\l_stex_fors_seq`.

`\stex_new_statement:nnn` `\stex_new_statement:nnn {<env name>} {<macro name>} {<code>}`
defines a new statement environment `s<env name>` and the optional macro-variant `\inline<macro name>` (may be empty). `{<code>}` does arbitrary additional setup and is called in the `\begin` part of the environment and the macro after optional arguments have been parsed. e.g. `\stex_new_statement:nnn{definition}{def}{...}` defines the `sdefinition` environment and the `\inlinedef` macro.

`sdefinition (env.)` (and `\inlinedef`)

`sassertion (env.)` (and `\inlineass`)

`sexample (env.)` (and `\inlineex`)

`sparagraph (env.)`

`definiens`

`\stex_add_definiens:nn` marks #2 as the definiens of the symbol with uri #1. Also marks the symbol as being defined, iff the symbol is declared in the current module.

`\varbind`

`\conclusion`

`\premise`

Chapter 21

Proofs

`sproof (env.)`

`subproof (env.)`

`spfsketchenv (env.)`

`\spfsketch`

`spfstepenv (env.)`

`spfblock (env.)`

`\spfstep`

`\assumption`

`\conclude`

`\eqstep`

`\yield`

`\spfjust`

`\spfby`

\spfarg

\sproofend

\stexcommentfont

Chapter 22

Metatheory

TODO

Chapter 23

Others

Chapter 24

Additional Packages

24.1 NotesSlides Documentation

TODO

24.2 Problem Documentation

TODO

24.3 HWExam Documentation

TODO

24.4 Tikzinput Documentation

TODO

Part IV

Implementation

Chapter 25

Setting up

Setup code for the document class

```
1 <*cls>
2 %%%%%%%%% stex.dtx %%%%%%%%%
3
4 \RequirePackage{expl3,l3keys2e}
5 \ProvidesExplClass{stex}{2026/06/24}{4.1.0}{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 \str_set:Nn \g_stex_document_kind_str{fragment}
15
16 \LoadClass{article}
17 </cls>
```

Setup code for the package

```
18 <*package>
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2026/06/24}{4.1.0}{sTeX package}
21 \IfFileExists{stex-expl-compat.sty}{
22   \usepackage{stex-expl-compat}
23 }{}
24 \RequirePackage{stex-logo} % externalized for backwards-compatibility reasons
25 \RequirePackage{standalone}
26
27 \bool_new:N \c_stex_use_sref_bool
28 \message{^^J*~This~is~sTeX~version~4.1.0~*^^J}
```

Package options:

```
29 \keys_define:nn { stex / package } {
30   debug      .str_set_x:N = \c_stex_debug_clist ,
31   lang       .clist_set:N = \c_stex_languages_clist ,
32   mathhub    .tl_set_x:N = \mathhub ,
33   usesms     .bool_set:N = \c_stex_persist_mode_bool ,
```

```

34  writesms .bool_set:N = \c_stex_persist_write_mode_bool ,
35  forcenousesms .bool_set:N = \c_stex_persist_force_bool ,
36  checkterms .bool_set:N = \c_stex_check_terms_bool ,
37  metadata .bool_set:N = \c_stex_metadata_bool ,
38  image .bool_set:N = \c_tikzinput_image_bool,
39  nofrontmatter .bool_set:N = \c_stex_no_frontmatter_bool,
40  unknown .code:n = {}
41 }
42 \exp_args:NNo \clist_set:Nn \c_stex_debug_clist \c_stex_debug_clist
43 \ProcessKeysOptions { stex / package }

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

```

Chapter 26

Utilities

`\stex_ignore_spaces_and_pars:`

```
45 \cs_new_protected:Nn \stex_ignore_spaces_and_pars:{
46   \begingroup\catcode13=10\relax
47   \@ifnextchar\par{
48     \endgroup\expandafter\stex_ignore_spaces_and_pars:\@gobble
49   }{
50     \endgroup
51   }
52 }
```

(End of definition for \stex_ignore_spaces_and_pars:. This function is documented on page 123.)
sfunction

`\stex_undefine:N`

`\stex_undefine:c`

```
53 \cs_new_protected:Nn \stex_undefine:N {
54   \cs_set_eq:NN #1 \tex_undefined:D
55 }
56 \cs_generate_variant:Nn \stex_undefine:N {c}
```

(End of definition for \stex_undefine:N. This function is documented on page 123.)
sfunction

`\stex_str_if_starts_with_p:nn`

`\stex_str_if_starts_with:nnTF`

```
57 \prg_new_conditional:Nnn \stex_str_if_starts_with:nn {p,T,F,TF} {
58   \exp_args:Ne \str_if_eq:nnTF {
59     \str_range:nnn{#1}{1}{\str_count:n{#2}}
60   }{#2}\prg_return_true: \prg_return_false:
61 }
```

(End of definition for \stex_str_if_starts_with:nnTF. This function is documented on page 123.)
sfunction

`\stex_str_if_ends_with_p:nn`

`\stex_str_if_ends_with:nnTF`

```
62 \prg_new_conditional:Nnn \stex_str_if_ends_with:nn {p,T,F,TF} {
63   \exp_args:Ne \str_if_eq:nnTF {
64     \str_range:nnn{#1}{- \str_count:n{#2}}{-1}
65   }{#2}\prg_return_true: \prg_return_false:
66 }
```

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

```
sfunction
```

```
\stex_deactivate_macro:Nn
\stex_reactivate_macro:N
67 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
68   \tl_set_eq:cN{\tl_to_str:n{#1}---orig}#1
69   \cs_set_protected:Npn#1{
70     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
71   }
72 }
73 \cs_new_protected:Nn \stex_reactivate_macro:N {
74   \cs_set_eq:Nc #1{\tl_to_str:n{#1}---orig}
75 }
```

(End of definition for `\stex_deactivate_macro:Nn` and `\stex_reactivate_macro:N`. These functions are documented on page 123.)

```
sfunction
```

26.1 kpsewhich and Environment Variables

```
76 <@@=stex_kpse>
```

```
\stex_kpsewhich:Nn
77 \cs_new_protected:Nn \stex_kpsewhich:Nn {
78   \group_begin:
79   \catcode'\ =12
80   \sys_get_shell:nnN { kpsewhich ~ #2 } { } \l_tmpa_tl
81   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
82   \group_end:
83   \exp_args:NNo\str_set:Nn #1 \l_tmpa_tl
84   \tl_trim_spaces:N #1
85 }
```

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

```
sfunction
```

```
\stex_get_env:Nn
86 \sys_if_platform_windows:TF{
87   \cs_new_protected:Nn \stex_get_env:Nn {\group_begin:
88     \escapechar=-1\catcode'\ =12
89     \exp_args:NNe \stex_kpsewhich:Nn #1 {-expand-var~\c_percent_str#2\c_percent_str}
90     \exp_args:NNx \str_if_eq:VnT #1 {\c_percent_str #2 \c_percent_str}{
91       \str_clear:N #1
92     }
93     \exp_args:NNx\use:nn\group_end:{
94       \str_set:Nn \exp_not:N #1 { #1 }
95     }
96   }
97 }{
98   \cs_new_protected:Nn \stex_get_env:Nn {
99     \stex_kpsewhich:Nn #1 {-var-value~#2}
100   }
101 }
```

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

sfunction

26.2 Logging

```

102 <@@=stex_debug>

\stex_debug:nn

103 \cs_new_protected:Nn \stex_debug:nn {
104   \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist { \tl_to_str:n{all} }{
105     \__stex_debug_:nn{#1}{#2}
106   }{
107     \exp_args:NNo \clist_if_in:NnT \c_stex_debug_clist { \tl_to_str:n{#1} }{
108       \__stex_debug_:nn{#1}{#2}
109     }
110   }
111 }

112
113 \cs_new_protected:Nn \__stex_debug_:nn {
114   \msg_set:nnn{stex}{debug / #1}{
115     \Debug~#1:~#2\
116   }
117   \msg_none:nn{stex}{debug / #1}
118 }

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

119 \stex_get_env:Nn\__stex_debug_env_str{STEX_DEBUG}
120 \str_if_empty:NTF\__stex_debug_env_str {
121   \clist_set_eq:NN \l__stex_debug_cl \c_stex_debug_clist
122 }{
123   \clist_set:No \l__stex_debug_cl {\__stex_debug_env_str}
124 }
125 \clist_clear:N \c_stex_debug_clist
126 \clist_map_inline:Nn \l__stex_debug_cl {
127   \exp_args:NNo \clist_put_right:Nn \c_stex_debug_clist
128   { \tl_to_str:n{#1} }
129 }
130
131 \exp_args:NNo \clist_if_in:NnTF \c_stex_debug_clist {\tl_to_str:n{all}} {
132   \msg_redirect_module:nnn{ stex }{ none }{ warning }
133   \stex_debug:nn{all}{Logging-everything!}
134 }{
135   \clist_map_inline:Nn \c_stex_debug_clist {
136     \msg_redirect_name:nnn{ stex }{ debug / #1 }{ warning }
137     \stex_debug:nn{#1}{Logging~#1}
138   }
139 }
```

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

sfunction

26.3 File Paths

```

140 <@@=stex_path>

\stex_file_set:Nn
\stex_file_set:No
\stex_file_set:Ne
141 \cs_new_protected:Nn \stex_file_set:Nn {
142   \str_if_empty:nTF {#2} { \seq_clear:N #1 }{
143     \exp_args:NNno \seq_set_split:Nnn #1 / { \tl_to_str:n{#2} }
144   }
145 }
146 \cs_generate_variant:Nn \stex_file_set:Nn {No, Ne}

(End of definition for \stex_file_set:Nn. This function is documented on page 124.)
sfunction

\stex_file_resolve:Nn
\stex_file_resolve:No
\stex_file_resolve:Ne
147 \sys_if_platform_windows:TF{
148   \cs_new_protected:Npn \__stex_path_win_take:w #1#2#3 \__stex_path_: {
149     \uppercase{ \str_set:Nn \l__stex_path_str{#1}}
150     \str_set:Ne \l__stex_path_win_drive {\l__stex_path_str #2}
151     \str_set:Nn \l__stex_path_str{#3}
152   }
153   \cs_new_protected:Nn \stex_file_resolve:Nn {
154     \str_set:Nn \l__stex_path_str {#2}
155     \str_clear:N \l__stex_path_win_drive
156     \exp_args:NNno \str_replace_all:Nnn \l__stex_path_str \c_backslash_str /
157     \exp_args:Ne \str_if_eq:nnT {\str_item:Nn \l__stex_path_str 2} : {
158       \exp_after:wN \__stex_path_win_take:w \l__stex_path_str \__stex_path_:
159     }
160     \stex_file_set:No #1 \l__stex_path_str
161     \__stex_path_canonicalize:N #1
162     \str_if_empty:NF \l__stex_path_win_drive {
163       \seq_pop_left:NN #1 \l__stex_path_str
164       \seq_put_left:No #1 \l__stex_path_win_drive
165     }
166     %\stex_debug:nn{files}{Set-\tl_to_str:n{#1}~to~\stex_file_use:N #1}
167   }
168 }{
169   \cs_new_protected:Nn \stex_file_resolve:Nn {
170     \str_set:Nn \l__stex_path_str {#2}
171     \stex_file_set:No #1 \l__stex_path_str
172     \__stex_path_canonicalize:N #1
173     % \stex_debug:nn{files}{Set-\tl_to_str:n{#1}~to~\stex_file_use:N #1}
174   }
175 }
176 \cs_generate_variant:Nn \stex_file_resolve:Nn {No, Ne}

Auxiliary methods:
177 \cs_new_protected:Nn \__stex_path_canonicalize:N {
178   \seq_if_empty:NF #1 {
179     \seq_pop:NN #1 \l__stex_path_can_str
180     \seq_clear:N \l__stex_path_can_seq
181     \str_if_empty:NTF \l__stex_path_can_str {
182       \seq_map_function:NN #1 \__stex_path_dodots:n
183       \seq_put_left:Nn \l__stex_path_can_seq {}

```

```

184     }{
185       \seq_push:No #1 \l__stex_path_can_str
186       \seq_map_function:NN #1 \__stex_path_dodots:n
187     }
188     \seq_set_eq:NN #1 \l__stex_path_can_seq
189   }
190 }
191
192 \cs_new_protected:Nn \__stex_path_dodots:n {
193   \str_if_empty:nF{#1}{
194     \str_if_eq:nnF {#1} {.} {
195       \str_if_eq:nnTF {#1} {...} {
196         \seq_if_empty:NF \l__stex_path_can_seq {
197           \seq_pop_right:NN \l__stex_path_can_seq \l__stex_path_can_str
198         }
199       }{
200         \seq_put_right:Nn \l__stex_path_can_seq {#1}
201       }
202     }
203   }
204 }

```

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

sfunction

`\stex_file_use:N`

```

205 \cs_new:Nn \stex_file_use:N {
206   \seq_use:Nn #1 /
207 }

```

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

sfunction

`\stex_file_split_off_ext:NN`

```

208 \cs_new_protected:Nn \stex_file_split_off_ext:NN {
209   \seq_set_eq:NN #1 #2
210   \seq_pop_right:NN #1 \l__stex_path_str
211   \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
212   \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
213   \seq_put_right:Ne #1 {\seq_use:Nn \l__stex_path_seq .}
214 }

```

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

sfunction

`\stex_file_split_off_lang:NN`

```

215 \cs_new_protected:Nn \stex_file_split_off_lang:NN {
216   \seq_set_eq:NN #1 #2
217   \seq_pop_right:NN #1 \l__stex_path_str
218   \seq_set_split:NnV \l__stex_path_seq . \l__stex_path_str
219   \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
220
221   \seq_pop_right:NN \l__stex_path_seq \l__stex_path_str
222   \exp_args:NNo \prop_if_in:NnF \c_stex_languages_prop \l__stex_path_str {

```

```

223   \seq_put_right:No \l__stex_path_seq \l__stex_path_str
224 }
225
226   \seq_put_right:Ne #1 {\seq_use:Nn \l__stex_path_seq .}
227 }

```

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

sfunction

```

\c_stex_pwd_file We determine the pwd
\c_stex_main_file
228 \sys_if_platform_windows:TF{
229   \stex_get_env:Nn \l__stex_path_str{CD}
230 }{
231   \stex_get_env:Nn \l__stex_path_str{PWD}
232 }
233 \stex_file_resolve:No \c_stex_pwd_file \l__stex_path_str
234 \seq_set_eq:NN \c_stex_main_file \c_stex_pwd_file
235 \seq_put_right:Ne \c_stex_main_file {\jobname\tl_to_str:n{.tex}}
236
237 \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 124.)

```

\c_stex_home_file
238 \sys_if_platform_windows:TF{
239   \stex_get_env:Nn \l__stex_path_str {homedrive\c_percent_str\c_percent_str homedrive}
240 }{
241   \stex_get_env:Nn \l__stex_path_str {HOME}
242 }
243 \stex_file_resolve:No \c_stex_home_file \l__stex_path_str

```

(End of definition for `\c_stex_home_file`. This variable is documented on page 125.)

```

\g_stex_current_file
244 \seq_gset_eq:NN \g_stex_current_file \c_stex_main_file

```

(End of definition for `\g_stex_current_file`. This variable is documented on page 125.)

```

\stex_input_with_hooks:Nn
\stex_input_with_hooks:Ne
245 \cs_new_protected:Nn \stex_input_with_hooks:Nn {
246   \stex_with_file_hooks:Nnn #1 {#2} {\input{#2}}
247 }
248 \cs_generate_variant:Nn \stex_input_with_hooks:Nn {Ne}

```

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

sfunction

```

\stex_with_file_hooks:Nnn
249 \cs_new_protected:Nn \stex_with_file_hooks:Nnn {
250   \stex_pseudogroup:nn {
251     \stex_file_resolve:Nn \l__stex_path_seq {#2}
252     \seq_gset_eq:NN \g_stex_current_file \l__stex_path_seq
253     \tl_set_eq:NN \l_stex_current_document_uri #1
254     \str_set:Ne \l_stex_current_language_str { \stex_document_uri_language:N \l_stex_current

```

```

255     #3
256   }{
257     \tl_gset:Nn \exp_not:N \g_stex_current_file { \exp_args:No \exp_not:n \g_stex_current_f
258     \stex_pseudogroup_restore:N \l_stex_current_language_str
259     \stex_pseudogroup_restore:N \l_stex_current_document_uri
260   }
261 }

```

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

sfunction

\stex_if_file_absolute_p:N
\stex_if_file_absolute:NTF

```

262 \sys_if_platform_windows:TF {
263   \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
264     \seq_if_empty:NTF #1 \prg_return_false: {
265       \exp_args:Ne \tl_if_empty:nTF {\seq_item:Nn #1 1} \prg_return_true: {
266         \exp_args:Ne \str_if_eq:nnTF { \exp_args:Ne \str_item:nn {\seq_item:Nn #1 1} 2} :
267         \prg_return_true: \prg_return_false:
268       }
269     }
270   }
271 }{
272   \prg_new_conditional:Nnn \stex_if_file_absolute:N {p, T, F, TF} {
273     \seq_if_empty:NTF #1 \prg_return_false: {
274       \exp_args:Ne \tl_if_empty:nTF {\seq_item:Nn #1 1}
275       \prg_return_true: \prg_return_false:
276     }
277   }
278 }

```

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

sfunction

\stex_if_file_starts_with:NNTF

```

279 \prg_new_protected_conditional:Nnn \stex_if_file_starts_with:NN {T,F,TF} {
280   \seq_set_eq:NN \l__stex_path_a_seq #1
281   \seq_set_eq:NN \l__stex_path_b_seq #2
282   \tl_clear:N \l__stex_path_return_tl
283   \bool_while_do:nn{
284     \bool_not_p:n{
285       \bool_lazy_any_p:n{
286         {\seq_if_empty_p:N \l__stex_path_a_seq}
287         {\seq_if_empty_p:N \l__stex_path_b_seq}
288         {\bool_not_p:n{\tl_if_empty_p:N \l__stex_path_return_tl}}
289       }
290     }
291   }{
292     \seq_pop_left:NN \l__stex_path_a_seq \l__stex_path_a_tl
293     \seq_pop_left:NN \l__stex_path_b_seq \l__stex_path_b_tl
294     \str_if_eq:NNF \l__stex_path_a_tl \l__stex_path_b_tl {
295       \tl_set:Nn \l__stex_path_return_tl {\prg_return_false:}
296     }
297   }
298   \tl_if_empty:NTF \l__stex_path_return_tl {

```

```

299     \seq_if_empty:NTF \l__stex_path_b_seq \prg_return_true: \prg_return_false:
300   } \l__stex_path_return_tl
301 }

```

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

```
sfunction
```

26.4 Group-like Behaviours

```

302 <@@=stex_groups>

```

`\stex_pseudogroup:nn`

```

303 \cs_new_protected:Npn \stex_pseudogroup:nn {
304   \exp_args:Nne \use:nn
305 }

```

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

```
sfunction
```

`\stex_pseudogroup_restore:N`

```

306 \cs_new:Nn \stex_pseudogroup_restore:N {
307   \tl_if_exist:NTF #1 {
308     \tl_set:Nn \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
309   }{
310     \stex_undefine:N \exp_not:N #1
311   }
312 }

```

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

```
sfunction
```

`\stex_pseudogroup_with:nn`

```

313 \cs_new_protected:Nn \stex_pseudogroup_with:nn {
314   \tl_map_inline:nn{#1}{
315     \cs_set_eq:cN{__stex_groups_\tl_to_str:n{##1}}##1
316   }
317   #2
318   \tl_map_inline:nn{#1}{
319     \cs_set_eq:Nc##1{__stex_groups_\tl_to_str:n{##1}}
320     \stex_undefine:c{__stex_groups_\tl_to_str:n{##1}}
321   }
322 }

```

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

```
sfunction
```

`\stex_metagroup_new:` Current metagroup group level

```

323 \int_new:N \l__stex_groups_lvl_int

```

start a new metagroup at the current group level

```

324 \cs_new_protected:Nn \stex_metagroup_new: {
325   \int_set_eq:NN \l__stex_groups_lvl_int \currentgrouplevel
326 }

```

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

sfunction

```

\stex_metagroup_do_in:n
\stex_metagroup_do_in:e
327 \cs_new_protected:Npn \stex_metagroup_do_in:n {
328   \int_compare:nNnTF \l__stex_groups_lvl_int = \currentgrouplevel
329     \use:n \__stex_groups_do_in:n
330 }
331 \cs_generate_variant:Nn \stex_metagroup_do_in:n {e}
332
333 \cs_new_protected:Nn \__stex_groups_do_in:n {
334   \tl_if_exist:cTF{g__stex_groups\_the\currentgrouplevel\_tl}{
335     \exp_args:Nno \tl_gput_right:cn{g__stex_groups\_the\currentgrouplevel\_tl}
336   }{
337     \exp_args:Nno \tl_gset:cn{g__stex_groups\_the\currentgrouplevel\_tl}
338   }{ #1 }
339   \bool_if_exist:cF {l__stex_groups\_the\currentgrouplevel\_bool} {
340     \group_insert_after:N \__stex_groups_do:
341     \bool_set_true:c {l__stex_groups\_the\currentgrouplevel\_bool}
342   }
343   #1
344 }
345
346 \cs_new_protected:Nn \__stex_groups_do: {
347   \tl_if_exist:cT{g__stex_groups\_int_eval:n{\currentgrouplevel+1}_tl}{
348     \exp_args:Nno \exp_args:No \stex_metagroup_do_in:n {
349       \csname g__stex_groups\_int_eval:n{\currentgrouplevel+1}_tl \endcsname
350     }
351     \cs_undefine:c{g__stex_groups\_int_eval:n{\currentgrouplevel+1}_tl}
352   }
353   \bool_if_exist:cF {l__stex_groups\_the\currentgrouplevel\_bool} {
354     \group_insert_after:N \__stex_groups_do:
355     \bool_set_true:c {l__stex_groups\_the\currentgrouplevel\_bool}
356   }
357 }

```

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

sfunction

26.5 Key Handling

358 <@@=stex_keys>

\stex_keys_define:nnnn

```

359 \cs_new_nopar:Nn \stex_keys_define:nnnn {
360   \tl_gset:cn {__stex_keys_keys_#1_pre_tl}{#2}
361   \tl_gset:cn {__stex_keys_keys_#1_def_tl}{#3}
362   \tl_if_empty:nF{#4}{
363     \clist_map_inline:nn{#4}{
364       \tl_set_eq:Nc \l__stex_keys_tl {__stex_keys_keys_##1_pre_tl}
365       \tl_gput_left:co{__stex_keys_keys_#1_pre_tl} \l__stex_keys_tl
366       \tl_set_eq:Nc \l__stex_keys_tl {__stex_keys_keys_##1_def_tl}
367       \tl_gput_left:cn{__stex_keys_keys_#1_def_tl} ,

```

```

368     \tl_gput_left:co{__stex_keys_keys_#1_def_tl} \l__stex_keys_tl
369   }
370 }
371 \tl_set_eq:Nc \l__stex_keys_tl {__stex_keys_keys_#1_def_tl}
372 \exp_args:Nno \keys_define:nn {stex / #1} {\l__stex_keys_tl}
373 }

```

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

sfunction

`\stex_keys_set:nn`

```

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

```

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

sfunction

Some ubiquitous key sets:

```

378 \stex_keys_define:nnnn{archive}{file}{
379   \str_clear:N \l_stex_key_archive_str
380   \str_clear:N \l_stex_key_file_str
381 }{
382   archive .str_set_x:N = \l_stex_key_archive_str ,
383   file .str_set_x:N = \l_stex_key_file_str
384 }{}
385
386 \stex_keys_define:nnnn{id}{
387   \str_clear:N \l_stex_key_id_str
388 }{
389   id .str_set_x:N = \l_stex_key_id_str
390 }{}
391
392 \stex_keys_define:nnnn{title}{
393   \tl_clear:N \l_stex_key_title_tl
394 }{
395   title .tl_set:N = \l_stex_key_title_tl
396 }{}
397
398 \stex_keys_define:nnnn{style}{
399   \clist_clear:N \l_stex_key_style_clist
400 }{
401   style .clist_set:N = \l_stex_key_style_clist
402 }{}
403
404 \stex_keys_define:nnnn{deprecate}{
405   \str_clear:N \l_stex_key_deprecate_str
406 }{
407   deprecate .str_set_x:N = \l_stex_key_deprecate_str
408 }{}
409
410 \stex_keys_define:nnnn{uses}{}{
411   uses .code:n = {
412     \clist_map_inline:nn{#1}{

```

```

413     \stex_str_if_starts_with:nnTF{##1}[]{
414       \__stex_keys_split_at_bracket:w ##1 \stex_end:
415     }{
416       \usemodule{##1}
417     }
418   }
419 }
420 }{}
421
422 \cs_new_protected:Npn \__stex_keys_split_at_bracket:w [ #1 ] #2 \stex_end: {
423   \usemodule[#1]{#2}
424 }

```

26.6 Languages

```

425 <@@=stex_lang>

```

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

```

426 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
427   en = english ,
428   de = ngerman ,
429   ar = arabic ,
430   bg = bulgarian ,
431   ru = russian ,
432   fi = finnish ,
433   ro = romanian ,
434   tr = turkish ,
435   fr = french ,
436   sl = slovenian
437 }}
438
439 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
440   english   = en ,
441   ngerman   = de ,
442   arabic    = ar ,
443   bulgarian = bg ,
444   russian   = ru ,
445   finnish   = fi ,
446   romanian  = ro ,
447   turkish   = tr ,
448   french    = fr ,
449   slovenian = sl ,
450 }}
451 % todo: chinese simplified (zhs)
452 %       chinese traditional (zht)

```

(End of definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 127.)

`\l_stex_current_language_str`

```

453 \str_new:N \l_stex_current_language_str

```

(End of definition for `\l_stex_current_language_str`. This variable is documented on page 127.)

Loading babel (if necessary), depending on the `lang` package option:


```

454 \clist_if_empty:NF \c_stex_languages_clist {
455   \bool_set_false:N \l__stex_lang_turkish_bool
456   \seq_clear:N \l_tmpa_seq
457   \clist_map_inline:Nn \c_stex_languages_clist {
458     \str_if_empty:NF \l_stex_current_language_str {
459       \str_set:Nn \l_stex_current_language_str {#1}
460     }
461     \str_set:Ne \l_tmpa_str {#1}
462     \str_if_eq:nnT {#1}{tr}{
463       \bool_set_true:N \l__stex_lang_turkish_bool
464     }
465     \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
466       \tl_set_rescan:Nno \l_tmpa_str {} \l_tmpa_str
467       \seq_put_right:No \l_tmpa_seq \l_tmpa_str
468     } {
469       \msg_error:nmx{stex}{error/unknownlanguage}{\l_tmpa_str}
470     }
471   }
472   \stex_debug:nn{lang} {Languages:~\seq_use:Nn \l_tmpa_seq {,~} }
473   \bool_if:NTF \l__stex_lang_turkish_bool {
474     \exp_args:NNe \use:nn \RequirePackage
475     {[main=\seq_use:Nn \l_tmpa_seq, ,shorthands=:!]}{babel}
476   }{
477     \exp_args:NNe \use:nn \RequirePackage
478     {[main=\seq_use:Nn \l_tmpa_seq, ]}{babel}
479   }
480 }

```

26.7 Styling

```

481 <@@=stex_styles>

```

```

\stex_new_stylable_cmd:nnnn
\stex_style_apply:

```

```

482 \cs_new_protected:Nn \stex_new_stylable_cmd:nnnn {
483   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[2] [] {
484     \__stex_styles_patch:nnn{#1}{##1}{##2}
485   }
486   \exp_after:wN \NewDocumentCommand\cs:w #1\cs_end:{#2}{
487     \cs_set:Npn \stex_style_apply: {
488       \__stex_styles_apply_patch:n{#1}
489     }
490     #3
491   }
492   \tl_set:cn {\__stex_styles_style_#1:} { #4 }
493 }
494
495 \cs_new_protected:Nn \__stex_styles_patch:nnn {
496   \str_if_empty:nTF {#2}{
497     \tl_set:cn{\__stex_styles_style_#1:}{#3}
498   }{
499     \tl_set:cn{\__stex_styles_style_#1_#2:}{#3}
500   }
501 }

```

```

502
503 \cs_new_protected:Nn \__stex_styles_apply_patch:n {
504   \clist_if_empty:NTF \l_stex_key_style_clist {
505     \tl_clear:N \thisstyle
506     \use:c{__stex_styles_style_#1:}
507   }{
508     \clist_get:NN \l_stex_key_style_clist \thisstyle
509     \tl_if_exist:cTF{__stex_styles_style_#1_\thisstyle :}{
510       \use:c{__stex_styles_style_#1_\thisstyle :}
511     }{
512       \use:c{__stex_styles_style_#1:}
513     }
514   }
515 }

```

(End of definition for `\stex_new_stylable_cmd:nnnn` and `\stex_style_apply:.` These functions are documented on page 127.)

sfunction

`\stex_new_stylable_env:nnnnnnn`

```

516 \cs_new_protected:Nn \stex_new_stylable_env:nnnnnnn {
517   \exp_after:wN \newcommand \cs:w stexstyle#1 \cs_end:[3] []{
518     \__stex_styles_patch:nnnn{#1}{##1}{##2}{##3} % <- defined after \stex_if_html_backend
519   }
520   \exp_after:wN \newcommand \cs:w stexcss#1 \cs_end:[1] []{
521     \__stex_styles_css_patch:nnnn{#1}{##1}
522   }
523   \NewDocumentEnvironment{#7#1}{#2}{
524     \cs_set:Npn \stex_style_apply: {
525       \stex_apply_patch_begin:n{#1}
526     }
527     #3
528   }{
529     \stex_if_html_backend:F{
530       \cs_set:Npn \stex_style_apply: {
531         \stex_apply_patch_end:n{#1}
532       }
533     }
534     #4
535   }
536   \tl_set:cn {__stex_styles_style_#1_start:} { #5 }
537   \tl_set:cn {__stex_styles_style_#1_end:} { #6 }
538 }
539
540
541 \cs_new_protected:Nn \stex_apply_patch_begin:n {
542   \clist_if_empty:NTF \l_stex_key_style_clist {
543     \tl_clear:N \thisstyle
544     \use:c{__stex_styles_style_#1_start:}
545   }{
546     \clist_get:NN \l_stex_key_style_clist \thisstyle
547     \stex_debug:nn{styling}{dominant~style:~\thisstyle}
548     \tl_if_exist:cTF{__stex_styles_style_#1_\thisstyle _start:}{
549       \use:c{__stex_styles_style_#1_\thisstyle _start:}

```

```

550     }{
551       \use:c{__stex_styles_style_#1_start:}
552     }
553   }
554 }
555
556 \cs_new_protected:Nn \_stex_apply_patch_end:n {
557   \tl_if_empty:NTF \thisstyle {
558     \use:c{__stex_styles_style_#1_end:}
559   }{
560     \tl_if_exist:cTF{__stex_styles_style_#1\_thisstyle _end:}{
561       \use:c{__stex_styles_style_#1\_thisstyle _end:}
562     }{
563       \use:c{__stex_styles_style_#1_end:}
564     }
565   }
566 }

```

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

sfunction

26.8 Persisting Dependencies

```

567 <@@=stex_persist>

```

We check the environment variables:

```

568 \stex_get_env:Nn\__stex_persist_env_str{STEX_USESMS}
569 \str_if_empty:NF\__stex_persist_env_str{
570   \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false}{
571     \bool_set_true:N \c_stex_persist_mode_bool
572   }
573 }
574 \stex_get_env:Nn\__stex_persist_env_str{STEX_WRITESMS}
575 \str_if_empty:NF\__stex_persist_env_str{
576   \exp_args:No \str_if_eq:nnF \__stex_persist_env_str{false}{
577     \bool_set_true:N \c_stex_persist_write_mode_bool
578   }
579 }
580
581 \bool_if:NT \c_stex_persist_force_bool {
582   \bool_set_false:N \c_stex_persist_mode_bool
583 }
584
585 \iow_new:N \c__stex_persist_sms_iow

```

`\stex_persist:n` defined later; requires `\stex_if_html_backend:TF`

`\stex_persist:e`
`\loadsms` (End of definition for `\stex_persist:n` and `\loadsms`. These functions are documented on page 127.)

sfunction

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

```

586
587 \cs_new_protected:Npn \loadsms #1 {
588   \stex_str_if_ends_with:nnTF{#1}{.sms}{
589     \__stex_persist_load_file:n{#1}

```

```

590   }{
591     \stex_str_if_ends_with:nnTF{#1}{.tex}{
592       \__stex_persist_load_file:n{#1}
593     }{
594       \__stex_persist_load_file:n{#1.sms}
595     }
596   }
597 }
598
599 \cs_new_protected:Nn \_stex_persist_read_now: {
600   \bool_if:NTF \c_stex_persist_mode_bool {
601     \bool_if:NTF \c_stex_persist_write_mode_bool
602     \__stex_persist_read_and_write:
603     {
604       \__stex_persist_load_file:n{\jobname.sms}
605     }
606   }{
607     \bool_if:NT \c_stex_persist_write_mode_bool \__stex_persist_write_only:
608   }
609 }
610
611 \cs_new_protected:Nn \__stex_persist_load_file:n {
612   \file_if_exist:nT{#1}{
613     \group_begin:
614     \cs_set:Npn \stex_persist:n ##1 {}
615     \cs_set:Npn \stex_persist:e ##1 {}
616     \stex_debug:nn{persist}{restoring~from~sms~file}
617     \catcode'\:=11\relax
618     \catcode'\_:=11\relax
619     \catcode'\ =10\relax
620     \cs:w @ @ input \cs_end:#1\relax
621   \group_end:
622 }
623 }
624
625 \cs_new_protected:Nn \__stex_persist_write_only: {
626   \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
627   \AtEndDocument{ \iow_close:N \c__stex_persist_sms_iow }
628 }
629
630 \cs_new_protected:Nn \__stex_persist_read_and_write: {
631   \file_if_exist:nTF{\jobname.sms}{
632     \ior_open:Nn \g_tmpa_ior {\jobname.sms}
633     \iow_open:Nn \g_tmpa_iow {\jobname.sms2}
634     \ior_str_map_inline:Nn \g_tmpa_ior {
635       \iow_now:Nn \g_tmpa_iow {##1}
636     }
637     \iow_close:N \g_tmpa_iow
638     \ior_close:N \g_tmpa_ior
639     \__stex_persist_write_only:
640     \ior_open:Nn \g_tmpa_ior {\jobname.sms2}
641     \ior_str_map_inline:Nn \g_tmpa_ior {
642       \iow_now:Nn \c__stex_persist_sms_iow {##1}
643     }

```

```

644 \ior_close:N \g_tmpa_ior
645 \__stex_persist_load_file:n{\jobname.sms2}
646 }\__stex_persist_write_only:
647 }

```

26.9 Auxiliary Methods

```

648 <@@=stex_aux>

```

`_stex_do_deprecation:n`

```

649 \cs_new:Nn \_stex_do_deprecation:n {
650   \str_if_empty:NF \l_stex_key_deprecate_str {
651     \msg_warning:nxxx{stex}{warning/deprecated}{#1}{\l_stex_key_deprecate_str}
652   }
653 }

```

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

`_stex_do_id:`

```

654 \cs_new_protected:Nn \_stex_do_id: {
655   \stex_if_smsmode:F {
656     \str_if_empty:NF \l_stex_key_id_str {
657       \exp_args:No \stex_ref_new_doc_target:n \l_stex_key_id_str
658     }
659   }
660 }

```

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

Chapter 27

HTML Output

```
661 <@=stex_annotate>
\stex@backend We determine and \input the backend config file:
662 \ifcstype if@rustex\endcstype\else
663   \expandafter\newif\cstype if@rustex\endcstype
664   \@rustexfalse
665 \fi
666
667 \stex_get_env:Nn\__stex_annotate_env_str{STEX_FORCE_PDF}
668 \exp_args:No \str_if_eq:nnTF \__stex_annotate_env_str {true} {
669   \def\stex@backend{pdflatex}
670 }{
671   \tl_if_exist:NF\stex@backend{
672     \if@rustex
673       \def\stex@backend{rustex}
674     \else
675       \cs_if_exist:NTF\HCode{
676         \def\stex@backend{tex4ht}
677       }{
678         \def\stex@backend{pdflatex}
679       }
680     \fi
681   }
682 }
683
684 \input{stex-backend-\stex@backend.cfg}

(End of definition for \stex@backend. This variable is documented on page 128.)

\stex_if_html_backend_p: is provided by the backend config file.
\stex_if_html_backend:TF (End of definition for \stex_if_html_backend:TF. This function is documented on page 128.)
sfunction

\ifstexhtml
685 \bool_new:N \l__stex_annotate_do_output_bool
686
687 \newif\ifstexhtml
688 \stex_if_html_backend:TF {
```

```

689 \stexhtmltrue
690 \bool_set_true:N \l__stex_annotate_do_output_bool
691 }{
692 \stexhtmlfalse
693 \bool_set_false:N \l__stex_annotate_do_output_bool
694 }

```

(End of definition for `\ifstexhtml`. This function is documented on page 128.)
sfunction

```

\stex_if_do_html_p:
\stex_if_do_html:TF
695 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
696 \bool_if:NTF \l__stex_annotate_do_output_bool
697 \prg_return_true: \prg_return_false:
698 }

```

(End of definition for `\stex_if_do_html:TF`. This function is documented on page 128.)
sfunction

```

\stex_suppress_html:n
699 \cs_new_protected:Nn \stex_suppress_html:n {
700 \stex_pseudogroup:nn{
701 \bool_set_false:N \l__stex_annotate_do_output_bool
702 #1
703 }{
704 \stex_if_do_html:T {
705 \bool_set_true:N \l__stex_annotate_do_output_bool
706 }
707 }
708 }

```

(End of definition for `\stex_suppress_html:n`. This function is documented on page 128.)
sfunction

`\stex_annotate:nn` is provided by the backend config file.

(End of definition for `\stex_annotate:nn`. This function is documented on page 128.)
sfunction

`stex_annotate_env (env.)` is provided by the backend config file.

`\stex_html_node:nnn` is provided by the backend config file.

(End of definition for `\stex_html_node:nnn`. This function is documented on page 128.)
sfunction

`stex_env_node (env.)` is provided by the backend config file.

`\stex_annotate_invisible:nn` is provided by the backend config file.

```

\stex_annotate_invisible:n

```

(End of definition for `\stex_annotate_invisible:nn` and `\stex_annotate_invisible:n`. These functions are documented on page 129.)
sfunction

`\STEXinvisible`

```
709 \cs_new_protected:Npn \STEXinvisible #1 {  
710   \stex_annotate_invisible:n { #1 }  
711 }
```

(End of definition for \STEXinvisible. This function is documented on page 129.)
sfunction

`\stex_css_link:n` is provided by the backend config file.

(End of definition for \stex_css_link:n. This function is documented on page 129.)
sfunction

`\stex_css_literal:n` is provided by the backend config file.

(End of definition for \stex_css_literal:n. This function is documented on page 129.)
sfunction

`_stex_annotate_force_break:n` is provided by the backend config file.

(End of definition for _stex_annotate_force_break:n. This function is documented on page 129.)
sfunction

`\mmlintent`

```
\mmlarg 712 \stex_if_html_backend:TF {  
713   \cs_new_protected:Npn \mmlintent #1 #2 {  
714     \stex_annotate:nn{\mml:intent={#1}}{#2}  
715   }  
716   \cs_new_protected:Npn \mmlarg #1 #2 {  
717     \stex_annotate:nn{\mml:arg={#1}}{#2}  
718   }  
719 }{  
720   \cs_new_protected:Npn \mmlintent #1 #2 { #2 }  
721   \cs_new_protected:Npn \mmlarg #1 #2 { #2 }  
722 }
```

(End of definition for \mmlintent and \mmlarg. These functions are documented on page 129.)
sfunction

We can now define `\stex_persist:n`:

```
723 <@@=stex_persist>  
724 \bool_if:NTF \c_stex_persist_write_mode_bool {  
725   \stex_if_html_backend:TF{  
726     \cs_new:Npn \stex_persist:n #1 {}  
727     \cs_new:Npn \stex_persist:e #1 {}  
728   }{  
729     \cs_new_protected:Nn \stex_persist:n {  
730       \iow_now:Nn \c__stex_persist_sms_iow {#1}  
731     }  
732     \cs_generate_variant:Nn \stex_persist:n {e}  
733   }  
734 }{  
735   \cs_new:Npn \stex_persist:n #1 {}  
736   \cs_new:Npn \stex_persist:e #1 {}  
737 }
```


and style patching:

```

738 <@@=stex_styles>
739 \stex_if_html_backend:TF{
740   \cs_new_protected:Nn \stex_style_apply: {}
741   \cs_new_protected:Nn \__stex_styles_patch:nnnn {}
742   \stex_keys_define:nnnn{ csspatch }{
743     \str_clear:N \l_stex_keys_cls_id
744     \str_clear:N \l_stex_keys_counter_id
745     \str_clear:N \l_stex_keys_counter_parent
746   }{
747     counter      .str_set_x:N = \l_stex_keys_counter_id,
748     parent       .str_set_x:N = \l_stex_keys_counter_parent,
749     unknown      .code:n      = {
750       \exp_args:NNo \str_set:Nn \l_stex_keys_cls_id {\l_keys_key_tl}
751     }
752   }{}
753   \cs_new_protected:Npn \__stex_styles_css_patch:nnnn #1 #2 {
754     \stex_keys_set:nn{ csspatch }{ #2 }
755     \group_begin:
756     \catcode'\ =12\relax
757     \catcode'\^J=12\relax
758     \__stex_styles_patch_i:nnn {#1}
759   }
760
761   \seq_new:N \g__stex_styles_counters_seq
762
763   \cs_new:Nn \__stex_styles_patch_html_c: {
764     \stex_html_literal:n{
765       <span~data-ftml-counter="\l_stex_keys_counter_id"~style="display:none;"~
766       data-ftml-counter-parent="\l_stex_keys_counter_parent"
767     ></span>
768   }
769 }
770
771 \cs_new:Nn \__stex_styles_patch_html: {
772   \stex_html_literal:n{
773     <span~data-ftml-style="\l_stex_keys_cls_id"~style="display:none;"~
774     data-ftml-counter="\l_stex_keys_counter_id"
775   ></span>
776 }
777 }
778
779 \cs_new_protected:Nn \__stex_styles_patch_i:nnn {
780   \str_if_empty:NTF \l_stex_keys_cls_id {
781     \str_set:Nn \l_stex_keys_cls_id { #1 }
782   }{
783     \str_set:Ne \l_stex_keys_cls_id { #1-\l_stex_keys_cls_id }
784   }
785   \exp_args:No \str_case:nnF \l_stex_keys_counter_parent {
786     {}{}
787     {part}{\str_set:Nn \l_stex_keys_counter_parent { 0 }}
788     {chapter}{\str_set:Nn \l_stex_keys_counter_parent { 1 }}
789     {section}{\str_set:Nn \l_stex_keys_counter_parent { 2 }}
790     {subsection}{\str_set:Nn \l_stex_keys_counter_parent { 3 }}

```

```

791     {subsubsection}{\str_set:Nn \l_stex_keys_counter_parent { 4 }}
792     {paragraph}{\str_set:Nn \l_stex_keys_counter_parent { 5 }}
793     {subparagraph}{\str_set:Nn \l_stex_keys_counter_parent { 6 }}
794   }{
795     \msg_error:nnx{stex}{error/notasection}\l_stex_keys_counter_parent
796   }
797   \str_set:Ne \l__stex_styles_first_str { &~.ftml-title-paragraph~ { display:inline-block
798   \str_if_empty:NF \l_stex_keys_counter_id {
799     %\str_put_left:Ne \l__stex_styles_first_str { counter-increment:~ ftml-\l_stex_keys_c
800     \exp_args:NNo \seq_if_in:NnF \g__stex_styles_counters_seq \l_stex_keys_counter_id {
801       \seq_gpush:No \g__stex_styles_counters_seq \l_stex_keys_counter_id
802       \cs_if_eq:ccTF{@onlypreamble}{@notprerr}{
803         \__stex_styles_patch_html_c:
804         % in body
805       }{
806         \exp_args:Ne \AtBeginDocument \__stex_styles_patch_html_c:
807         % in preamble
808       }
809     }
810     \cs_if_eq:ccTF{@onlypreamble}{@notprerr}{
811       \__stex_styles_patch_html:
812       % in body
813     }{
814       \exp_args:Ne \AtBeginDocument \__stex_styles_patch_html:
815       % in preamble
816     }
817   }
818   \exp_args:Ne \stex_css_literal:n { .ftml-\l_stex_keys_cls_id { \l__stex_styles_first_st
819
820   % TODO export counter reset somehow
821
822   \group_end:
823 }
824 }{
825   \cs_new_protected:Nn \__stex_styles_patch:nnnn {
826     \str_if_empty:NTF {#2}{
827       \tl_set:cn{\__stex_styles_style_#1_start:}{#3}
828       \tl_set:cn{\__stex_styles_style_#1_end:}{#4}
829     }{
830       \tl_set:cn{\__stex_styles_style_#1_#2_start:}{#3}
831       \tl_set:cn{\__stex_styles_style_#1_#2_end:}{#4}
832     }
833   }
834   \cs_new_protected:Nn \__stex_styles_css_patch:nnnn {}
835 }

```

Chapter 28

Math Archives

```
\c_stex_mathhub_files
\mathhub
836 <@@=stex_mathhub>
837 \str_if_empty:NTF\mathhub{
838   \stex_get_env:Nn \l__stex_mathhub_str {MATHHUB}
839   \str_if_empty:NTF \l__stex_mathhub_str {
840     \ior_open:NnTF \g_tmpa_ior{\stex_file_use:N \c_stex_home_file/.stex/mathhub.path}{
841       \group_begin:
842         \escapechar=-1\catcode'\=12
843         \ior_str_get:NN \g_tmpa_ior \l__stex_mathhub_str
844         \str_gset_eq:NN \l__stex_mathhub_str \l__stex_mathhub_str
845       \group_end:
846       \ior_close:N \g_tmpa_ior
847       \stex_debug:nn{mathhub}{MathHub-directory~determined~from~home~directory}
848     }{
849       \str_clear:N \l__stex_mathhub_str
850     }
851   }{
852     \stex_debug:nn{mathhub}{MathHub-directory~determined~from~environment~variable}
853   }
854 }{
855   \str_set_eq:NN \l__stex_mathhub_str \mathhub
856 }
857
858 \str_if_empty:NTF \l__stex_mathhub_str {
859   \msg_warning:nn{stex}{warning/nomathhub}
860   \stex_file_set:Ne \l_tmpa_seq {\stex_file_use:N \c_stex_home_file \tl_to_str:n{/MathHub}}
861   \seq_clear:N \c_stex_mathhub_files
862   \seq_push:Ne \c_stex_mathhub_files {\stex_file_use:N \l_tmpa_seq}
863 }{
864   \seq_clear:N \c_stex_mathhub_files
865   \seq_set_split:NnV \l_tmpa_seq , \l__stex_mathhub_str
866   \seq_map_inline:Nn \l_tmpa_seq {
867     \stex_file_resolve:Nn \l__stex_mathhub_str {#1}
868     \stex_if_file_absolute:NF \l__stex_mathhub_str {
869       \stex_file_resolve:Ne \l__stex_mathhub_str {
870         \stex_file_use:N \c_stex_pwd_file / #1
871       }

```

```

872     }
873     \seq_put_right:Ne \c_stex_mathhub_files {
874         \stex_file_use:N \l__stex_mathhub_str
875     }
876 }
877 }
878
879 \exp_args:NNe \str_set:Nn \mathhub {\seq_use:Nn \c_stex_mathhub_files ,}
880 \stex_debug:nn{mathhub}{MATHHUBS:~\mathhub}

```

(End of definition for `\c_stex_mathhub_files` and `\mathhub`. These variables are documented on page 132.)

`\stex_archive_id:N`

```

881 \cs_new:Npn \stex_archive_id:N {
882     \exp_after:wN \use_i:nnn
883 }

```

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

sfunction

`\stex_archive_base:N`

`\stex_archive_base:n`

```

884 \cs_new:Npn \stex_archive_base:N {
885     \exp_after:wN \use_ii:nnn
886 }
887
888 \cs_new:Nn \stex_archive_base:n {
889     \exp_args:NNe \use:nn \use_ii:nnn { \use:c {c_stex_mathhub_#1_archive} }
890 }

```

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

sfunction

`\stex_archive_path:N`

`\stex_archive_path:n`

```

891 \cs_new:Npn \stex_archive_path:N {
892     \exp_after:wN \use_iii:nnn
893 }
894
895 \cs_new:Nn \stex_archive_path:n {
896     \exp_args:NNe \use:nn \use_iii:nnn { \use:c {c_stex_mathhub_#1_archive} }
897 }

```

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

sfunction

`\stex_in_archive:nn`

```

898 \cs_new_protected:Nn \stex_in_archive:nn {
899     \stex_pseudogroup:nn{
900         \cs_set:Npn \l__stex_mathhub_cs ##1 {#2}
901         \tl_if_empty:nTF{#1}{
902             \tl_if_exist:NTF \l_stex_current_archive {
903                 \exp_args:Ne \l__stex_mathhub_cs {\stex_archive_id:N \l_stex_current_archive }
904             }{

```

```

905     \l__stex_mathhub_cs {}
906   }
907   }{
908     \__stex_mathhub_set_current:n{#1}
909     \l__stex_mathhub_cs {#1}
910   }
911   }{
912     \stex_pseudogroup_restore:N \l_stex_current_archive
913     \cs_set:Npn \exp_not:N \l__stex_mathhub_cs ##1 {
914       \exp_args:No \exp_not:n {\l__stex_mathhub_cs {##1}}
915     }
916   }
917 }
918
919 \cs_set:Npn \l__stex_mathhub_cs #1 {}

```

Auxiliary macros for loading archives:

```

920 \cs_new_protected:Nn \__stex_mathhub_set_current:n {
921   \__stex_mathhub_require:n { #1 }
922   \stex_debug:nn{mathhub}{switching-to~archive~#1}
923   \tl_set_eq:Nc \l_stex_current_archive {
924     c_stex_mathhub_#1_archive
925   }
926 }
927
928 \cs_new_protected:Nn \_stex_set_meta_archive: {
929   \prop_if_exist:cF { c_stex_mathhub_FTML/meta_archive } {
930     \seq_if_empty:NF \c_stex_mathhub_files {
931       \stex_debug:nn{mathhub}{Opening~archive:~FTML/meta}
932       \__stex_mathhub_do_manifest:nn { FTML/meta }{
933         \tl_gset:ce {c_stex_mathhub_FTML/meta_archive}{
934           {\tl_to_str:n{FTML/meta}}
935           {\tl_to_str:n{http://mathhub.info}}
936         }
937       }
938     }
939   }
940 }
941
942 }
943
944 \cs_new_protected:Nn \__stex_mathhub_require:n {
945   \prop_if_exist:cF { c_stex_mathhub_#1_archive } {
946     \seq_if_empty:NTF \c_stex_mathhub_files {
947       \msg_fatal:nn{stex}{warning/nomathhub}
948     }{
949       \stex_debug:nn{mathhub}{Opening~archive:~#1}
950       \__stex_mathhub_do_manifest:nn { #1 }{
951         \msg_fatal:nnee{stex}{error/noarchive}
952         {#1}{\seq_use:Nn \c_stex_mathhub_files ,}
953       }
954     }
955   }
956 }

```

Attempts to find the archive's manifest file:

```

957 \cs_new_protected:Nn \__stex_mathhub_do_manifest:nn {
958   \seq_map_inline:Nn \c_stex_mathhub_files {
959     \__stex_mathhub_find_manifest:nn {##1}{#1}
960     \str_if_empty:NF \l__stex_mathhub_manifest_str \seq_map_break:
961   }
962   %\exp_args:Ne \__stex_mathhub_find_manifest:n {\stex_file_use:N \c_stex_mathhub_file / #1}
963   \str_if_empty:NTF \l__stex_mathhub_manifest_str { #2 }{
964     \__stex_mathhub_parse_manifest:n {#1}
965   }
966 }
967
968 \cs_new_protected:Nn \__stex_mathhub_find_manifest:nn {
969   \str_clear:N \l__stex_mathhub_manifest_str
970   \seq_set_split:Nnn \l_tmpa_seq / { #1 }
971   \seq_set_split:Nnn \l__stex_mathhub_seq / {#1 / #2}
972   \bool_set_true:N \l__stex_mathhub_bool
973   \bool_while_do:Nn \l__stex_mathhub_bool {
974     \tl_if_eq:NNTF \l__stex_mathhub_seq \l_tmpa_seq {
975       \bool_set_false:N \l__stex_mathhub_bool
976     }{
977       \__stex_mathhub_check_manifest:
978       \bool_if:NT \l__stex_mathhub_bool {
979         \seq_pop_right:NN \l__stex_mathhub_seq \l__stex_mathhub_tl
980       }
981     }
982   }
983 }
984
985 \cs_new_protected:Nn \__stex_mathhub_check_manifest: {
986   \__stex_mathhub_check_manifest:n {MANIFEST.MF}
987   \bool_if:NT \l__stex_mathhub_bool {
988     \__stex_mathhub_check_manifest:n {META-INF/MANIFEST.MF}
989     \bool_if:NT \l__stex_mathhub_bool {
990       \__stex_mathhub_check_manifest:n {meta-inf/MANIFEST.MF}
991     }
992   }
993 }
994
995 \cs_new_protected:Nn \__stex_mathhub_check_manifest:n {
996   \stex_debug:nn{mathhub}{Checking~\stex_file_use:N \l__stex_mathhub_seq / #1}
997   \file_if_exist:nT {\stex_file_use:N \l__stex_mathhub_seq / #1} {
998     \bool_set_false:N \l__stex_mathhub_bool
999     \str_set:Ne \l__stex_mathhub_manifest_str {\stex_file_use:N \l__stex_mathhub_seq / #1}
1000   }
1001 }

```

Parses the archive's manifest file:

```

1002 \ior_new:N \c__stex_mathhub_manifest_ior
1003
1004 \str_set:Nn \l_tmpa_str {https://}
1005
1006 \exp_after:wN \cs_new:Npn \exp_after:wN \__stex_mathhub_replace_https:w \l_tmpa_str #1 \__s
1007   http:// #1

```

```

1008 }
1009
1010 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
1011   \ior_open:Nn \c__stex_mathhub_manifest_ior \l__stex_mathhub_manifest_str
1012   \str_set:Ne \l__stex_mathhub_file_str {\stex_file_use:N \l__stex_mathhub_seq}
1013   \str_set:Nn \l__stex_mathhub_id_str {#1}
1014   \str_set:Nn \l__stex_mathhub_base_str {http://mathhub.info}
1015
1016   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
1017     \exp_args:NNNo \exp_args:NNNx
1018     \seq_set_split:Nnn \l__stex_mathhub_seq \c_colon_str {\tl_to_str:n{##1}}
1019     \seq_pop_left:NNT \l__stex_mathhub_seq \l__stex_mathhub_key {
1020       \exp_args:NNo \str_set:Nn \l__stex_mathhub_key \l__stex_mathhub_key
1021       \str_set:Ne \l__stex_mathhub_val {\seq_use:Nn \l__stex_mathhub_seq :}
1022       \str_case:Nn \l__stex_mathhub_key {
1023         {id} { \str_set_eq:NN \l__stex_mathhub_id_str \l__stex_mathhub_val }
1024         {url-base} { \str_set_eq:NN \l__stex_mathhub_base_str \l__stex_mathhub_val }
1025       }
1026     }
1027   }
1028   \ior_close:N \c__stex_mathhub_manifest_ior
1029   \exp_args:No \stex_str_if_starts_with:nnT \l__stex_mathhub_base_str {https://} {
1030     \str_set:Ne \l__stex_mathhub_base_str { \exp_after:wN \__stex_mathhub_replace_https:w \
1031   }
1032   \tl_gset:ce { c_stex_mathhub_#1_archive } { {\l__stex_mathhub_id_str} {\l__stex_mathhub_b
1033   \stex_debug:nn{mathhub}{Result:~\use:c{c_stex_mathhub_#1_archive}}
1034   \str_if_eq:nnF {#1}{main}{
1035     \stex_persist:e {
1036       \__stex_mathhub_restore:nn{\l__stex_mathhub_id_str}{\l__stex_mathhub_base_str}
1037     }
1038   }
1039 }

```

The `.sms` file persisting content should not store the file path of the archive, since that depends on the directory layout of the system that generated it. We therefore, when restoring archive macros, check if we find the archive on the local system, and if not, leave the directory path empty. This will throw an error iff some functionality at some point *requires* the archive to actually exist locally.

```

1040 \cs_set_protected:Nn \__stex_mathhub_restore:nn {
1041   \__stex_mathhub_do_manifest:nn { #1 }{}
1042   \tl_if_exist:cF { c_stex_mathhub_#1_archive }{
1043     \tl_set:ce{ c_stex_mathhub_#1_archive }{
1044       {\tl_to_str:n{#1}}
1045       {\tl_to_str:n{#2}}
1046     }
1047   }
1048 }
1049 }
1050

```

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

sfunction

```

\c_stex_no_archive_str
\c_stex_no_archive

```

```

1051 \str_const:Nn \c_stex_no_archive_str { no/archive }
1052
1053 \tl_const:Ne \c_stex_no_archive_uri {
1054   {\c_stex_no_archive_str}
1055   {\tl_to_str:n{http://unknown.source}}
1056   {}
1057 }
1058
1059 \tl_set_eq:cN {c_stex_mathhub_ no/archive _ archive}\c_stex_no_archive_uri

```

(End of definition for `\c_stex_no_archive_str` and `\c_stex_no_archive`. These variables are documented on page 133.)

`\c_stex_main_archive`
`\l_stex_current_archive`

```

1060 \seq_map_inline:Nn \c_stex_mathhub_files {
1061   \seq_set_split:Nnn \l_tmpa_seq / {#1}
1062   \stex_if_file_starts_with:NNT \c_stex_pwd_file \l_tmpa_seq {
1063     \seq_set_eq:NN \l_tmpb_seq \c_stex_pwd_file
1064     \seq_map_inline:Nn \l_tmpa_seq { \seq_pop_left:NN \l_tmpb_seq \l_tmpa_tl }
1065     \exp_args:Nne \_stex_mathhub_find_manifest:nn {#1} { \stex_file_use:N \l_tmpb_seq }
1066     \str_if_empty:NTF \l__stex_mathhub_manifest_str {
1067       \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~archive}
1068     }{
1069       \_stex_mathhub_parse_manifest:n { main }
1070       \tl_set_eq:NN \c_stex_main_archive \c_stex_mathhub_main_archive
1071       \cs_undefine:N \c_stex_mathhub_main_archive
1072       \tl_set_eq:cN { c_stex_mathhub_ \stex_archive_id:N \c_stex_main_archive _archive }
1073       \c_stex_main_archive
1074       \prop_set_eq:NN \l_stex_current_archive \c_stex_main_archive
1075       \stex_debug:nn{mathhub}{Current~archive:~
1076         \stex_archive_id:N \c_stex_main_archive
1077       }
1078       \stex_persist:e {
1079         \_stex_mathhub_restore:nn{\stex_archive_id:N \c_stex_main_archive}{\stex_archive_b
1080         \tl_gset_eq:Nc \exp_not:N \c_stex_main_archive {c_stex_mathhub_ \stex_archive_id:N
1081         \tl_gset_eq:NN \exp_not:N \l_stex_current_archive \exp_not:N \c_stex_main_archive
1082       }
1083       %\bool_if:NT \c_stex_persist_write_mode_bool {
1084       %   \tl_put_right:Ne \_stex_persist_read_now: {
1085       %     \stex_persist:n {{c_stex_mathhub_ \stex_archive_id:N \c_stex_main_archive _archi
1086       %       \tl_gset_eq:cN{c_stex_mathhub_ \stex_archive_id:N \c_stex_main_archive _manife
1087       %       \tl_gset_eq:NN \exp_not:N \l_stex_current_archive \exp_not:N \c_stex_main_archi
1088       %     }
1089       %   }
1090       %}
1091   }
1092   \seq_map_break:
1093 }
1094 }
1095 \tl_if_exist:NF \c_stex_main_archive {
1096   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~directory}
1097 }

```

(End of definition for `\c_stex_main_archive` and `\l_stex_current_archive`. These variables are documented on page 133.)


```

\stex_source_path:n
\stex_source_path:nn
1098 \cs_new:Nn \stex_source_path:n {
1099   \tl_if_exist:NTF \l_stex_current_archive {
1100     \stex_archive_path:N \l_stex_current_archive / source \tl_if_empty:nF{#1}{/ #1}
1101   }{
1102     \stex_file_use:N \g_stex_current_file / .. \tl_if_empty:nF{#1}{/ #1}
1103   }
1104 }
1105 \cs_new:Nn \stex_source_path:nn {
1106   \tl_if_empty:nTF{#1}{
1107     \stex_source_path:n {#2}
1108   }{
1109     \tl_if_exist:cTF {c_stex_mathhub_ #1 _archive } {
1110       \stex_archive_path:n {#1} / source \tl_if_empty:nF{#2}{/ #2}
1111     }{
1112       \stex_file_use:N \g_stex_current_file / .. \tl_if_empty:nF{#2}{/ #2}
1113     }
1114   }
1115 }

```

(End of definition for \stex_source_path:n and \stex_source_path:nn. These functions are documented on page 133.)

sfunction

Chapter 29

URIs

```
1116 <@@=stex_uris>
```

```
\stex_use_archive_uri:n
```

```
1117 \cs_new:Nc \stex_use_archive_uri:n {
1118   \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1
1119 }
```

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

```
sfunction
```

29.1 Document URIs

```
\stex_use_document_uri:N
```

```
\stex_use_document_uri:n
```

```
1120 \cs_new:Nc \stex_use_document_uri:N {
1121   \exp_after:wN \__stex_uris_doc:nnnnn #1
1122 }
1123 \cs_new:Nc \stex_use_document_uri:n {
1124   \__stex_uris_doc:nnnnn #1
1125 }
1126 \cs_new:Nc \__stex_uris_doc:nnnnn {
1127   \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1
1128   \exp_not:N \tl_if_empty:nF{#2}{
1129     \c_ampersand_str\tl_to_str:n{p=}#2
1130   }
1131   \c_ampersand_str\tl_to_str:n{d=}#3
1132   \c_ampersand_str\tl_to_str:n{l=}#4
1133   \exp_not:N \tl_if_empty:nF{#5}{
1134     \c_ampersand_str\tl_to_str:n{e=}#5
1135   }
1136 }
```

(End of definition for \stex_use_document_uri:N and \stex_use_document_uri:n. These functions are documented on page 134.)

```
sfunction
```

```
\stex_document_uri_archive:N
```

```
1137 \cs_new:Nc \stex_document_uri_archive:N {
1138   \exp_after:wN \use_i:nnnnn #1
```

```

1139 }
(End of definition for \stex_document_uri_archive:N. This function is documented on page 134.)
sfunction

```

`\stex_document_uri_path:N`

```

1140 \cs_new:Nn \stex_document_uri_path:N {
1141   \exp_after:wN \use_ii:nnnnn #1
1142 }
(End of definition for \stex_document_uri_path:N. This function is documented on page 135.)
sfunction

```

`\stex_document_uri_name:N`

```

1143 \cs_new:Nn \stex_document_uri_name:N {
1144   \exp_after:wN \use_iii:nnnnn #1
1145 }
(End of definition for \stex_document_uri_name:N. This function is documented on page 135.)
sfunction

```

`\stex_document_uri_language:N`

```

1146 \cs_new:Nn \stex_document_uri_language:N {
1147   \exp_after:wN \use_iv:nnnnn #1
1148 }
(End of definition for \stex_document_uri_language:N. This function is documented on page 135.)
sfunction

```

`\stex_document_uri_element:N`

```

1149 \cs_new:Nn \stex_document_uri_element:N {
1150   \exp_after:wN \use_v:nnnnn #1
1151 }
(End of definition for \stex_document_uri_element:N. This function is documented on page 135.)
sfunction

```

`\stex_document_uri_with_language:Nn`

```

1152 \cs_new:Npn \stex_document_uri_with_language:Nn {
1153   \exp_after:wN \__stex_uris_with_language:nnnnnn
1154 }
1155 \cs_new:Nn \__stex_uris_with_language:nnnnnn {
1156   {#1}{#2}{#3}{#6}{}
1157 }
(End of definition for \stex_document_uri_with_language:Nn. This function is documented on page 135.)
sfunction

```

`\stex_new_document_element_uri:n`

```

1158 \cs_new:Npn \stex_new_document_element_uri:n {
1159   \exp_after:wN \__stex_uris_with_elem:nnnnnn \l_stex_current_document_uri
1160 }
1161 \cs_new:Nn \__stex_uris_with_elem:nnnnnn {
1162   {#1}{#2}{#3}{#4}{ \tl_if_empty:nTF{#5}{#6}{#5/#6}}
1163 }

```

(End of definition for \stex_new_document_element_uri:n. This function is documented on page 135.)

sfunction

\stex_document_uri_from_archive_file:Nn

```

1164 \cs_new_protected:Nn \stex_document_uri_from_archive_file:Nn {
1165   \stex_file_set:Nn \l__stex_uris_file {#2}
1166   \stex_debug:nn{URI}{relative~path:~\stex_file_use:N \l__stex_uris_file}
1167   \__stex_uris_doc_from_archive_file:NN #1 \l__stex_uris_file
1168 }
1169
1170 \cs_new_protected:Nn \__stex_uris_doc_from_archive_file:NN {
1171   \__stex_uris_split_file:N #2
1172   \prop_if_exist:NTF \l_stex_current_archive {
1173     \str_set:Ne \l__stex_uris_archive {
1174       \stex_archive_id:N \l_stex_current_archive
1175     }
1176   }{
1177     \str_set_eq:NN \l__stex_uris_archive \c_stex_no_archive_str
1178   }
1179   \tl_set:Ne #1 {
1180     { \l__stex_uris_archive }
1181     { \stex_file_use:N \l__stex_uris_path_seq }
1182     { \l__stex_uris_name_str }
1183     { \l__stex_uris_lang_str }
1184   }
1185 }
1186 }
1187
1188 \cs_new_protected:Nn \__stex_uris_split_file:N {
1189   \seq_set_eq:NN \l__stex_uris_path_seq #1
1190   \seq_pop_right:NN \l__stex_uris_path_seq \l__stex_uris_name_str
1191   \seq_set_split:NnV \l_tmpa_seq . \l__stex_uris_name_str
1192   \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_lang_str % .tex?
1193   \seq_if_empty:NTF \l_tmpa_seq {
1194     \str_set_eq:NN \l__stex_uris_name_str \l__stex_uris_lang_str
1195     \str_set_eq:NN \l__stex_uris_lang_str \l_stex_current_language_str
1196   }\__stex_uris_split_file_i:
1197 }
1198
1199 \cs_new_protected:Nn \__stex_uris_split_file_i: {
1200   \cs_if_eq:NNTF \l__stex_uris_lang_str \q_no_value {
1201     \str_set_eq:NN \l__stex_uris_lang_str \l_stex_current_language_str
1202   }{
1203     \prop_if_in:NoF \c_stex_languages_prop \l__stex_uris_lang_str {
1204       \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_lang_str % actual language maybe?
1205       \cs_if_eq:NNTF \l__stex_uris_lang_str \q_no_value {
1206         \str_set_eq:NN \l__stex_uris_lang_str \l_stex_current_language_str
1207       }{
1208         \prop_if_in:NoF \c_stex_languages_prop \l__stex_uris_lang_str {
1209           \seq_put_right:No \l_tmpa_seq \l__stex_uris_lang_str
1210           \str_set:Nn \l__stex_uris_lang_str {en}
1211         }
1212       }
1213     }

```

```

1214 }
1215 \str_set:Ne \l__stex_uris_name_str {\seq_use:Nn \l_tmpa_seq .}
1216 }
1217
1218 \cs_new_protected:Nn \__stex_uris_split_file_ii: {
1219
1220 }

```

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

sfunction

```

\c_stex_main_document_uri
\l_stex_current_document_uri
1221 \tl_if_exist:NTF \l_stex_current_archive {
1222   \str_set:Ne \l_tmpa_str { \stex_archive_path:N \l_stex_current_archive }
1223   \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
1224   \seq_set_eq:NN \l_tmpb_seq \c_stex_main_file
1225   \seq_map_inline:Nn \l_tmpa_seq {
1226     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_tl
1227   }
1228   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_tl
1229   \__stex_uris_doc_from_archive_file:NN \l_stex_current_document_uri \l_tmpb_seq
1230 }{
1231   \__stex_uris_doc_from_archive_file:NN \l_stex_current_document_uri \c_stex_main_file
1232 }
1233
1234 \tl_set_eq:NN \c_stex_main_document_uri \l_stex_current_document_uri
1235 \str_set:Ne \l_stex_current_language_str { \stex_document_uri_language:N \l_stex_current_do
1236
1237 \stex_debug:nn{URIs}{Current~document::~ \stex_use_document_uri:N \c_stex_main_document_uri}

```

(End of definition for `\c_stex_main_document_uri` and `\l_stex_current_document_uri`. These variables are documented on page 135.)

29.2 Module URIs

```

\stex_use_module_uri:N
\stex_use_module_uri:n
1238 \cs_new:Nn \stex_use_module_uri:N {
1239   \exp_after:wN \__stex_uris_module:nnn #1
1240 }
1241 \cs_new:Nn \stex_use_module_uri:n {
1242   \__stex_uris_module:nnn #1
1243 }
1244 \cs_new:Ne \__stex_uris_module:nnn {
1245   \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1
1246   \exp_not:N \tl_if_empty:nF{#2}{
1247     \c_ampersand_str\tl_to_str:n{p=}#2
1248   }
1249   \c_ampersand_str\tl_to_str:n{m=}#3
1250 }

```

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

sfunction

`\stex_module_uri_archive:N`

```
1251 \cs_new:Nn \stex_module_uri_archive:N {
1252   \exp_after:wN \use_i:nnn #1
1253 }
```

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

sfunction

`\stex_module_uri_path:N`

`\stex_module_uri_path:n`

```
1254 \cs_new:Nn \stex_module_uri_path:N {
1255   \exp_after:wN \use_ii:nnn #1
1256 }
1257 \cs_new:Nn \stex_module_uri_path:n {
1258   \use_ii:nnn #1
1259 }
```

(End of definition for \stex_module_uri_path:N and \stex_module_uri_path:n. These functions are documented on page 135.)

sfunction

`\stex_module_uri_name:N`

`\stex_module_uri_name:n`

```
1260 \cs_new:Nn \stex_module_uri_name:N {
1261   \exp_after:wN \use_iii:nnn #1
1262 }
1263
1264 \cs_new:Nn \stex_module_uri_name:n {
1265   \use_iii:nnn #1
1266 }
```

(End of definition for \stex_module_uri_name:N and \stex_module_uri_name:n. These functions are documented on page 136.)

sfunction

`\stex_module_uri_split_name:NNN`

`\stex_module_uri_split_name:NNn`

```
1267 \cs_new_protected:Npn \stex_module_uri_split_name:NNN #1 {
1268   \exp_after:wN \__stex_uris_parent:NNnnn \exp_after:wN #1 \exp_after:wN
1269 }
1270
1271 \cs_new_protected:Nn \stex_module_uri_split_name:NNn {
1272   \__stex_uris_parent:NNnnn #1 #2 #3
1273 }
1274
1275 \cs_new_protected:Nn \__stex_uris_parent:NNnnn {
1276   \seq_set_split:Nnn #1 / {#5}
1277   \seq_pop_right:NN #1 #2
1278   \seq_if_empty:NTF #1 {
1279     \tl_set:Ne #1 { {#3} {#4} {#3}}
1280     \tl_clear:N #2
1281   }{
1282     \tl_set:Ne #1 { {#3} {#4} {\seq_use:Nn #1 /} }
1283   }
1284 }
```

(End of definition for `\stex_module_uri_split_name:NNN` and `\stex_module_uri_split_name:NNn`. These functions are documented on page 136.)

sfunction

`\stex_module_uri_as_qm:n`

```
1285 \cs_new:Nn \stex_module_uri_as_qm:n {
1286   \__stex_uris_as_qm:nnn #1
1287 }
1288 \cs_new:Nn \__stex_uris_as_qm:nnn {
1289   #1 / #2 ? #3
1290 }
```

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

sfunction

`\stex_new_module_uri:n`

```
1291 \cs_new:Npn \stex_new_module_uri:n {
1292   \stex_if_in_module:TF {
1293     \exp_after:wN \__stex_uris_new_nested_mod:nnnn \l_stex_current_module_uri
1294   }{
1295     \exp_after:wN \__stex_uris_new_mod:nnnnnn \l_stex_current_document_uri
1296   }
1297 }
1298
1299 \cs_new:Nn \__stex_uris_new_mod:nnnnnn {
1300   {#1}
1301   \str_if_eq:nnTF{#3}{#6}{
1302     {#2}{#3}
1303   }{
1304     {#2 \tl_if_empty:nF{#2}/ #3}{ \tl_to_str:n{ #6 } }
1305   }
1306 }
1307 \cs_new:Nn \__stex_uris_new_nested_mod:nnnn {
1308   {#1}
1309   {#2}
1310   {#3 / \tl_to_str:n{#4}}
1311 }
```

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

sfunction

`\stex_uri_from_pair:Nnn`

```
1312 \bool_new:N \l_stex_relative_import_bool
1313 \cs_new_protected:Nn \stex_uri_from_pair:Nnn {
1314   \bool_set_false:N \l_stex_relative_import_bool
1315   \stex_debug:nn{uri}{resolving~<#2,#3>~in~\stex_use_document_uri:N \l_stex_current_document_uri}
1316   \seq_set_split:Nne \l_tmpa_seq ? { \tl_to_str:n {#3} }
1317   \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_name_str
1318   \str_clear:N \l__stex_uris_path_str
1319   \seq_if_empty:NF \l_tmpa_seq {
1320     \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_path_str
1321   }
1322   \tl_if_empty:nTF { #2 }{
1323     \str_if_empty:NTF \l__stex_uris_path_str
```

```

1324     \__stex_uris_maybe_relative:N \__stex_uris_absolute:N
1325     #1
1326   }{
1327     \__stex_uris_pair_in_archive:Nn #1 { #2 }
1328   }
1329 }
1330 %\cs_generate_variant:Nn \stex_uri_from_pair:Nnn {Nee}
1331
1332 \cs_new_protected:Nn \__stex_uris_maybe_relative:N {
1333   \tl_set:Ne \l_tmpb_tl {
1334     \stex_document_uri_path:N \l_stex_current_document_uri
1335   }
1336   \tl_set:Ne \l_tmpa_tl {
1337     {
1338       \stex_document_uri_archive:N \l_stex_current_document_uri
1339     }
1340     {
1341       \l_tmpb_tl
1342       \exp_args:NNo \exp_args:Nne \str_if_eq:nnF \l__stex_uris_name_str {
1343         \stex_document_uri_name:N \l_stex_current_document_uri
1344       }{
1345         \str_if_empty:NF \l_tmpb_tl / \stex_document_uri_name:N \l_stex_current_document_uri
1346       }
1347     }
1348     { \l__stex_uris_name_str }
1349   }
1350   \stex_if_module_exists:NTF \l_tmpa_tl {
1351     \tl_set_eq:NN #1 \l_tmpa_tl
1352     \bool_set_true:N \l_stex_relative_import_bool
1353     \stex_debug:nn{uri}{returning~relative~\stex_use_module_uri:N #1}
1354   }{
1355     \__stex_uris_absolute:N #1
1356   }
1357 }
1358
1359 \cs_new_protected:Nn \__stex_uris_absolute:N {
1360   \tl_if_exist:NTF \l_stex_current_archive {
1361     \tl_set:Ne #1 {
1362       { \stex_archive_id:N \l_stex_current_archive }
1363       { \l__stex_uris_path_str }
1364       { \l__stex_uris_name_str }
1365     }
1366     \stex_debug:nn{uri}{returning~\stex_use_module_uri:N #1}
1367   }{
1368     \__stex_uris_pair_no_archive:N #1
1369   }
1370 }
1371
1372 \cs_new_protected:Nn \__stex_uris_pair_no_archive:N {
1373   \stex_file_resolve:Ne \l_tmpa_seq {
1374     \stex_file_use:N \g_stex_current_file / .. / \l__stex_uris_path_str
1375   }
1376   \tl_set:Ne #1 {
1377     { \c_stex_no_archive_str }

```



```

1378     { \stex_file_use:N \l_tmpa_seq }
1379     { \l__stex_uris_name_str }
1380   }
1381   \stex_debug:nn{uri}{returning~\stex_use_module_uri:N #1}
1382 }
1383
1384 \cs_new_protected:Nn \__stex_uris_pair_in_archive:Nn {
1385   \tl_set:Nc #1 {
1386     { \tl_to_str:n{ #2 } }
1387     { \l__stex_uris_path_str }
1388     { \l__stex_uris_name_str }
1389   }
1390   \stex_debug:nn{uri}{returning~\stex_use_module_uri:N #1}
1391 }

```

(End of definition for \stex_uri_from_pair:Nnn. This function is documented on page 136.)

sfunction

\stex_uri_from_pair:Nn

```

1392 \cs_new_protected:Nn \stex_uri_from_pair:Nn {
1393   \peek_charcode:NTF [ {
1394     \__stex_uris_parse_i:Nw #1
1395   }{
1396     \__stex_uris_parse_ii:Nw #1
1397   } #2 \__stex_uris_end:
1398 }
1399
1400 \cs_new_protected:Npn \__stex_uris_parse_i:Nw #1 [#2] #3 \__stex_uris_end: {
1401   \stex_uri_from_pair:Nnn #1 {#2} {#3}
1402 }
1403
1404 \cs_new_protected:Npn \__stex_uris_parse_ii:Nw #1 #2 \__stex_uris_end: {
1405   \stex_uri_from_pair:Nnn #1 {} {#2}
1406 }

```

(End of definition for \stex_uri_from_pair:Nn. This function is documented on page 136.)

sfunction

29.3 Symbol URIs

\stex_use_symbol_uri:N

\stex_use_symbol_uri:n

```

1407 \cs_new:Nn \stex_use_symbol_uri:N {
1408   \exp_after:wN \__stex_uris_symbol:nnnn #1
1409 }
1410 \cs_new:Nn \stex_use_symbol_uri:n {
1411   \__stex_uris_symbol:nnnn #1
1412 }
1413 \cs_new:Nc \__stex_uris_symbol:nnnn {
1414   \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1
1415   \exp_not:N \tl_if_empty:nF{#2}{
1416     \c_ampersand_str\tl_to_str:n{p=}#2
1417   }
1418   \c_ampersand_str\tl_to_str:n{m=}#3

```

```

1419 \c_ampersand_str\tl_to_str:n{s=}#4
1420 }

```

(End of definition for \stex_use_symbol_uri:N and \stex_use_symbol_uri:n. These functions are documented on page 136.)

```
sfunction
```

```
\stex_new_symbol_uri:n
```

```

1421 \cs_new:Nn \stex_new_symbol_uri:n {
1422   \l_stex_current_module_uri { \tl_to_str:n{ #1 } }
1423 }

```

(End of definition for \stex_new_symbol_uri:n. This function is documented on page 136.)

```
sfunction
```

```
\stex_new_symbol_uri:nn
```

```

1424 \cs_new:Nn \stex_new_symbol_uri:nn {
1425   #1 { #2 }
1426 }

```

(End of definition for \stex_new_symbol_uri:nn. This function is documented on page 136.)

```
sfunction
```

```
\stex_symbol_uri_archive:N
```

```

1427 \cs_new:Nn \stex_symbol_uri_archive:N {
1428   \exp_after:wN \use_i:nnnn #1
1429 }

```

(End of definition for \stex_symbol_uri_archive:N. This function is documented on page 136.)

```
sfunction
```

```
\stex_symbol_uri_path:N
```

```

1430 \cs_new:Nn \stex_symbol_uri_path:N {
1431   \exp_after:wN \use_ii:nnnn #1
1432 }

```

(End of definition for \stex_symbol_uri_path:N. This function is documented on page 136.)

```
sfunction
```

```
\stex_symbol_uri_module:N
```

```
\stex_symbol_uri_module:n
```

```

1433 \cs_new:Nn \stex_symbol_uri_module:N {
1434   \exp_after:wN \__stex_uris_first_three:nnnn #1
1435 }
1436 \cs_new:Nn \stex_symbol_uri_module:n {
1437   \__stex_uris_first_three:nnnn #1
1438 }
1439
1440 \cs_new:Nn \__stex_uris_first_three:nnnn {
1441   {#1}{#2}{#3}
1442 }

```

(End of definition for \stex_symbol_uri_module:N and \stex_symbol_uri_module:n. These functions are documented on page 137.)

```
sfunction
```

`\stex_symbol_uri_name:N`

`\stex_symbol_uri_name:n`

```
1443 \cs_new:Nn \stex_symbol_uri_name:N {  
1444   \exp_after:wN \use_iv:nnnn #1  
1445 }  
1446 \cs_new:Nn \stex_symbol_uri_name:n {  
1447   \use_iv:nnnn #1  
1448 }
```

(End of definition for `\stex_symbol_uri_name:N` and `\stex_symbol_uri_name:n`. These functions are documented on page 137.)

sfunction

Chapter 30

Documents

```
1449 (@@=stex_doc)

\STEXftmllink
\STEXlinkftml
\STEXsetlinkftml

1450 \cs_new_protected:Npn \STEXftmllink {
1451   \exp_args:Nn \url{
1452     \stex_use_document_uri:N \l_stex_current_document_uri
1453   }
1454 }
1455
1456 \cs_new_protected:Npn \STEXsetlinkftml #1 {
1457   \tl_gset:Nn \g__stex_doc_ftml_link_text_tl { #1 }
1458 }
1459
1460 \tl_gset:Nn \g__stex_doc_ftml_link_text_tl{
1461   The~
1462   \stex_get_symbol:nF{FTML}{~}
1463   \tl_if_empty:NTF \l_stex_get_symbol_uri {FTML}{\sn{FTML}}
1464   {~ version ~ of ~ this ~ document ~ can ~ be ~ found ~ at\\
1465   \STEXftmllink
1466 }
1467
1468 \NewDocumentCommand \STEXlinkftml { 0{ } } {
1469   \ifstexhtml\else
1470     \begin{center}
1471       \emph{
1472         \tl_if_empty:NTF{#1}\g__stex_doc_ftml_link_text_tl{#1}
1473       }
1474     \end{center}
1475   \fi
1476 }
```

(End of definition for \STEXftmllink, \STEXlinkftml, and \STEXsetlinkftml. These functions are documented on page 138.)

sffunction

\stexdoctitle Initial definition (before \begin{document}):

```
1477 \tl_new:N \g__stex_doc_title_tl
1478
1479 \str_new:N \g_stex_document_kind_str
```

```

1480 \tl_new:N \g_stex_document_kind_parameters_tl
1481 \str_set:Nn \g_stex_document_kind_str{article}
1482 \stex_if_html_backend:T{
1483   \AtBeginDocument{
1484     \exp_args:Nne\stex_html_node:nnn{span}{
1485       data-ftml-document-kind=\g_stex_document_kind_str
1486       \g_stex_document_kind_parameters_tl
1487     }{}
1488   }
1489 }
1490
1491 \cs_new_protected:Npn \stexdoctitle #1 {
1492   \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1493   \global\def\stexdoctitle##1{}
1494 }

```

At begin document, we switch to:

```

1495 \cs_new_protected:Nn \__stex_doc_set_title:n {
1496   \stex_if_smsmode:F{
1497     \gdef\stexdoctitle##1{}
1498     \stex_debug:nn{title}{Setting~title~to:~\tl_to_str:n{#1}}
1499     \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1500     \__stex_doc_title_html:
1501   }
1502 }
1503
1504 \cs_new_protected:Nn \__stex_doc_title_html: {
1505   \stex_if_do_html:T{
1506     \stex_annotate_invisible:nn{data-ftml-doctitle={}}{ \hbox{\g__stex_doc_title_tl} }
1507   }
1508 }
1509
1510 \AtBeginDocument {
1511   \tl_if_empty:NTF \g__stex_doc_title_tl {
1512     \cs_set_eq:NN \stexdoctitle \__stex_doc_set_title:n
1513   }{
1514     \gdef\stexdoctitle#1{}
1515     \__stex_doc_title_html:
1516   }
1517
1518   \cs_set_eq:NN \__stex_doc_maketitle: \maketitle
1519   \protected\gdef\maketitle{
1520     \tl_if_empty:NF \@title {
1521       \exp_args:No \stexdoctitle \@title
1522     }
1523     \__stex_doc_maketitle:
1524   }
1525 }

```

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

sfunction

30.1 Sectioning

`\l_stex_current_section_level_int`
`\l_stex_current_section_level_str`

```
1526 \int_new:N \l_stex_current_section_level_int
1527 \str_set:Nn \l_stex_current_section_level_str {document}
```

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

`\currentsectionlevel`
`\Currentsectionlevel`

```
1528 \cs_set_protected:Npn \currentsectionlevel {
1529   \stex_if_do_html:TF{
1530     \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
1531     \stex_annotate:nn{data-ftml-currentsectionlevel={},data-ftml-capitalize=false}{}
1532   }{
1533     \l_stex_current_section_level_str
1534   }
1535   \tl_if_exist:NT\xspace\xspace
1536 }
1537
1538 \cs_set_protected:Npn \Currentsectionlevel {
1539   \stex_if_do_html:TF{
1540     \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
1541     \stex_annotate:nn{data-ftml-currentsectionlevel={},data-ftml-capitalize=true}{}
1542   }{
1543     \exp_after:wN \str_uppercase:n \l_stex_current_section_level_str
1544   }
1545   \tl_if_exist:NT\xspace\xspace
1546 }
```

(End of definition for `\currentsectionlevel` and `\Currentsectionlevel`. These functions are documented on page 139.)

sfunction

`sfragment (env.)`

```
1547 \stex_keys_define:nnnn{ sfragment }{
1548   \tl_clear:N \l_stex_key_short_tl
1549 }{
1550   short .tl_set:N = \l_stex_key_short_tl
1551 }{id}
1552
1553 \NewDocumentEnvironment{sfragment}{0{} m}{
1554   \stex_keys_set:nn{sfragment}{#1}
1555   \__stex_doc_do_section:n{#2}
1556   \stex_do_id: % TODO
1557 }{
1558   \_sfragment_end:
1559 }
1560
1561 \cs_new_protected:Npn \__stex_doc_do_section:n {
1562   \int_case:nnF \l_stex_current_section_level_int {
1563     {0}{\cs_if_exist:NTF \thepart {\_sfragment_do_level:nn{part}}{
1564       \int_incr:N \l_stex_current_section_level_int
1565       \__stex_doc_do_section:n
```

```

1566     }}
1567     {1}{\cs_if_exist:NTF \thechapter {\_sfragment_do_level:nn{chapter}}{
1568         \int_incr:N \l_stex_current_section_level_int
1569         \__stex_doc_do_section:n
1570     }}
1571     {2}{\_sfragment_do_level:nn{section}}
1572     {3}{\_sfragment_do_level:nn{subsection}}
1573     {4}{\_sfragment_do_level:nn{subsubsection}}
1574     {5}{\_sfragment_do_level:nn{paragraph}}
1575 }{\_sfragment_do_level:nn{subparagraph}}
1576 }
1577
1578 \stex_if_html_backend:TF {
1579     \cs_new_protected:Nn \_sfragment_do_level:nn {
1580         \stexdoctitle{#2}
1581         \par
1582         \begin{stex_env_node}{section}{data-ftml-section={\int_use:N \l_stex_current_section_level_int}
1583             \noindent\stex_html_node:nnn{h1}{data-ftml-title={}}{
1584                 \stex_annotate_force_break:n{#2}
1585             }\par
1586         }
1587         \cs_new_protected:Nn \_sfragment_end: {
1588             \end{stex_env_node}
1589         }
1590     }{
1591         \cs_new_protected:Nn \_sfragment_do_level:nn {
1592             \stexdoctitle{#2}
1593             \tl_if_empty:NTF \l_stex_key_short_tl {
1594                 \use:c{#1}
1595             }{
1596                 \exp_args:Nne \use:nn{\use:c{#1}}{[\exp_args:No \exp_not:n \l_stex_key_short_tl]}
1597             }{#2}
1598
1599             \int_compare:nNnF \l_stex_current_section_level_int = 6{
1600                 \int_incr:N \l_stex_current_section_level_int
1601             }
1602             \tl_set:Nn\l_stex_current_section_level_str{#1}
1603         }
1604         \cs_new_protected:Nn \_sfragment_end: {
1605

```

`titlefragment (env.)`

```

1606 \bool_new:N \l__stex_doc_titlefragment_bool
1607
1608 \NewDocumentEnvironment{titlefragment}{0}{m}{
1609     \stex_keys_set:nn{sfragment}{#1}
1610     \cs_if_exist:NTF \maketitle {
1611         \bool_set_true:N \l__stex_doc_titlefragment_bool
1612         \title{#2}
1613     }{\tl_if_empty:NF \l_stex_key_short_tl {
1614         \cs_if_exist:NT \shorttitle {
1615             \exp_args:No \shorttitle \l_stex_key_short_tl
1616         }
1617     }

```

```

1618     \stexdoctitle{#2}
1619     \maketitle
1620   }{
1621     \bool_set_false:N \l__stex_doc_titlefragment_bool
1622     \__stex_doc_do_section:n{#2}
1623     \_stex_do_id:
1624   }
1625 }{
1626   \bool_if:NF \l__stex_doc_titlefragment_bool \_sfragment_end:
1627 }

```

blindfragment (*env.*)

```

1628 \cs_new_protected:Nn \__stex_doc_skip_section_i: {
1629   \int_case:nn \l_stex_current_section_level_int {
1630     {0}{\cs_if_exist:NF \thepart {
1631       \int_incr:N \l_stex_current_section_level_int \__stex_doc_skip_section_i:
1632     }}
1633     {1}{\cs_if_exist:NF \thechapter {
1634       \int_incr:N \l_stex_current_section_level_int \__stex_doc_skip_section_i:
1635     }}
1636   }
1637   \int_compare:nNnF \l_stex_current_section_level_int = 6{
1638     \int_incr:N \l_stex_current_section_level_int
1639   }
1640 }
1641
1642 \stex_if_html_backend:TF {
1643   \cs_new_protected:Nn \__stex_doc_skip_section: {
1644     \__stex_doc_skip_section_i:
1645     \begin{stex_annotate_env}{data-ftml-skipsection={\int_use:N \l_stex_current_section_level_int}}
1646     \_stex_annotate_force_break:n{}
1647   }
1648 }{
1649   \cs_set_eq:NN \__stex_doc_skip_section: \__stex_doc_skip_section_i:
1650 }
1651
1652 \NewDocumentEnvironment{blindfragment}{}{
1653   \__stex_doc_skip_section:
1654 }{
1655   \stex_if_html_backend:T{
1656     \stex_annotate_invisible:n{~}
1657     \end{stex_annotate_env}
1658   }
1659 }

```

\skipfragment

```

1660 \cs_new_protected:Npn \skipfragment {
1661   \int_case:nnF \l_stex_current_section_level_int {
1662     {0}{\cs_if_exist:NTF \thepart {\stepcounter{part}}{
1663       \int_incr:N \l_stex_current_section_level_int
1664       \skipfragment
1665     }}
1666     {1}{\cs_if_exist:NTF \thechapter {\stepcounter{chapter}}{
1667       \int_incr:N \l_stex_current_section_level_int

```



```

1668     \skipfragment
1669   }}
1670   {2}{\stepcounter{section}}
1671   {3}{\stepcounter{subsection}}
1672   {4}{\stepcounter{subsubsection}}
1673   {5}{\stepcounter{paragraph}}
1674   }{\stepcounter{subparagraph}}
1675 }

```

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

sfunction

\setsectionlevel

```

1676 \cs_new_protected:Npn \setsectionlevel #1 {
1677   \str_case:nnF{#1}{
1678     {part}{\int_set:Nn \l_stex_current_section_level_int 0}
1679     {chapter}{\int_set:Nn \l_stex_current_section_level_int 1}
1680     {section}{\int_set:Nn \l_stex_current_section_level_int 2}
1681     {subsection}{\int_set:Nn \l_stex_current_section_level_int 3}
1682     {subsubsection}{\int_set:Nn \l_stex_current_section_level_int 4}
1683     {paragraph}{\int_set:Nn \l_stex_current_section_level_int 5}
1684   }{
1685     \int_set:Nn \l_stex_current_section_level_int 6
1686   }
1687   \stex_if_do_html:T{
1688     \cs_if_eq:NNTF\@onlypreamble\@notprerr{
1689       \stex_html_literal:n{<span~ data-ftml-sectionlevel="\int_use:N\l_stex_current_section_level_int}
1690     }{
1691       \AtBeginDocument{
1692         \stex_html_literal:n{<span~ data-ftml-sectionlevel="\int_use:N\l_stex_current_section_level_int}
1693       }
1694     }
1695   }
1696 }

```

In HTML mode, we make sure to record the top section level:

```

1697 \stex_if_html_backend:T{
1698   \cs_new_protected:Nn \__stex_doc_check_topsect: {
1699     \int_case:nnF \l_stex_current_section_level_int {
1700       {0}{\cs_if_exist:NTF \thepart {
1701         \stex_annotate_invisible:nn{data-ftml-sectionlevel=0}{}}
1702       }{
1703         \int_incr:N \l_stex_current_section_level_int
1704         \__stex_doc_check_topsect:
1705       }}
1706     {1}{\cs_if_exist:NTF \thechapter {
1707       \stex_annotate_invisible:nn{data-ftml-sectionlevel=1}{}}
1708     }{
1709       \int_incr:N \l_stex_current_section_level_int
1710       \__stex_doc_check_topsect:
1711     }}
1712   }{
1713     \stex_annotate_invisible:nn{data-ftml-sectionlevel={\int_use:N\l_stex_current_section_level_int}}
1714   }
1715 }

```

```

1716 \AtBeginDocument{\__stex_doc_check_topsect:
1717 \cs_if_exist:NT \thechapter {
1718 \int_case:nnT \l_stex_current_section_level_int{
1719 {0}{}}
1720 {1}{}}
1721 }{
1722 \stex_css_literal:n {
1723 .ftml-section {
1724 &~.ftml-title-section { &::before {
1725 content:~counter(ftml-chapter)~"."~counter(ftml-section)~"";
1726 } }
1727 }
1728 .ftml-subsection {
1729 &~.ftml-title-subsection { &::before {
1730 content:~counter(ftml-chapter)~"."~counter(ftml-section)~"."~counter(ftml-sub
1731 } }
1732 }
1733 .ftml-subsubsection {
1734 &~.ftml-title-subsubsection { &::before {
1735 content:~counter(ftml-chapter)~"."~counter(ftml-section)~"."~counter(ftml-sub
1736 } }
1737 }
1738 }
1739 }
1740 }
1741 }
1742 }

```

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

sfunction

We make sure `\frontmatter` and `\backmatter` are always defined. **TODO: this seems like a hack and probably shouldn't be here**

```

1743 \bool_if:NF \c_stex_no_frontmatter_bool {
1744 \cs_new_protected:Nn \__stex_doc_x_matter: {
1745 \cs_if_exist:NTF\frontmatter{
1746 \let\__stex_doc_orig_frontmatter\frontmatter
1747 \let\__stex_doc_orig_endfrontmatter\endfrontmatter
1748 \let\endfrontmatter\relax
1749 \let\frontmatter\relax
1750 }{
1751 \tl_set:Nn\__stex_doc_orig_frontmatter{
1752 \clearpage
1753 %\@mainmatterfalse
1754 \pagenumbering{roman}
1755 }
1756 \tl_set:Nn\__stex_doc_orig_endfrontmatter{}
1757 }
1758 \cs_if_exist:NTF\backmatter{
1759 \let\__stex_doc_orig_backmatter\backmatter
1760 \let\__stex_doc_orig_endbackmatter\endbackmatter
1761 \let\backmatter\relax
1762 \let\endbackmatter\relax
1763 }{
1764 \tl_set:Nn\__stex_doc_orig_backmatter{

```

```

1765     \clearpage
1766     %\@mainmatterfalse
1767     \pagenumbering{roman}
1768   }
1769 }
1770 \newenvironment{frontmatter}{
1771   \__stex_doc_orig_frontmatter
1772 }{
1773   \cs_if_exist:NTF\mainmatter{
1774     \mainmatter
1775   }{
1776     \clearpage
1777     %\@mainmattertrue
1778     \pagenumbering{arabic}
1779   }
1780 }
1781 \newenvironment{backmatter}{
1782   \__stex_doc_orig_backmatter
1783 }{
1784   \cs_if_exist:NTF\mainmatter{
1785     \mainmatter
1786   }{
1787     \clearpage
1788     %\@mainmattertrue
1789     \pagenumbering{arabic}
1790   }
1791 }
1792 }
1793 \AddToHook{begindocument/end}\__stex_doc_x_matter:
1794 }

```

30.2 Inter-Document References

1795 <@@=stex_refs>

The .sref-file:

```

1796 \iow_new:N \c__stex_refs_iow
1797 \AtBeginDocument{ \iow_open:Nn \c__stex_refs_iow {\jobname.sref} }
1798 \AtEndDocument{\iow_close:N \c__stex_refs_iow}
1799 \str_set:Ne \c__stex_refs_iow_str { \stex_file_use:N \c_stex_pwd_file / }

```

Keys:

```

1800 \stex_keys_define:nnnn{sref / 1}{
1801   \tl_clear:N \l_stex_key_pre_tl
1802   \tl_clear:N \l_stex_key_post_tl
1803   \tl_clear:N \l_stex_key_fallback_tl
1804 }{
1805   % TODO get rid of this
1806   fallback .tl_set:N = \l_stex_key_fallback_tl,
1807   pre      .code:n = {},
1808   post     .code:n = {}
1809 }{archive file}
1810 \stex_keys_define:nnnn{sref / 2}{}{}{archive file, title}

```

Target declarations:

\sreflabel

\stex_ref_new_doc_target:n

```

1811 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1812   \stex_if_smsmode:F{
1813     \tl_set:Nc \l__stex_refs_new_tl { \stex_new_document_element_uri:n { #1 } }
1814     \exp_args:Nc \label { \stex_use_document_uri:N \l__stex_refs_new_tl }
1815     \exp_args:Nne \STeXInternalNewSRefLabel{#1}{ \stex_use_document_uri:N \l__stex_refs_new_tl }
1816     \iow_now:Nc \auxout {
1817       \STeXInternalNewSRefLabel{#1}{ \stex_use_document_uri:N \l__stex_refs_new_tl }
1818     }
1819     \iow_now:Nc \c__stex_refs_iow {
1820       \exp_not:N \STeXInternalSRefLabel
1821       { #1 }
1822       { \stex_use_document_uri:N \l__stex_refs_new_tl }
1823       { \exp_not:o \@currentHref }
1824       { \exp_not:o \@currentlabelname }
1825     }
1826   }
1827 }
1828
1829 \NewDocumentCommand \sreflabel {m} { \stex_ref_new_doc_target:n{#1} }

```

(End of definition for \sreflabel and \stex_ref_new_doc_target:n. These functions are documented on page 86.)

sfunction

\stex_ref_new_sym_target:n

```

1830 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1831   \stex_if_smsmode:F{
1832     \cs_if_exist:cTF{r@\tl_to_str:n{#1}}{
1833       \cs_if_exist:cTF{__stex_sref_aux_sym_ \tl_to_str:n{#1}}{
1834         \cs_undefine:c{__stex_sref_aux_sym_ \tl_to_str:n{#1}}
1835         \exp_args:Nc \label { \tl_to_str:n{#1} }
1836       }
1837     }{ \exp_args:Nc \label { \tl_to_str:n{#1} } }
1838   }
1839 }

```

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

sfunction

Inserting References:

\sref

```

1840 \NewDocumentCommand \sref { 0{} m 0{} }{
1841   \stex_keys_set:nn { sref / 1 }{ #1 }
1842   \tl_clear:N \l__stex_refs_tmp
1843   \str_if_empty:NTF \l_stex_key_file_str {
1844     \__stex_refs_local:n
1845   }{
1846     \__stex_refs_remote:nn{#3}
1847   }{#2}
1848 }
1849
1850 \cs_new_protected:Nn \__stex_refs_local:n {
1851   \stex_debug:nn{sref}{Checking~#1}

```

```

1852 \tl_if_exist:cTF{ g_stex_sref_label_ #1 }{
1853   \%exp_args:Ne \stex_annotate:nn{data-ftml-sref={\use:c{ g_stex_sref_label_ #1 }}}{
1854     \exp_args:Ne \__stex_refs_local_ref:n {\use:c{ g_stex_sref_label_ #1 }}
1855   }%}
1856 }{
1857   \tl_if_empty:NTF \l_stex_key_fallback_tl { ??? }{ \l_stex_key_fallback_tl }
1858 }
1859 }
1860
1861 \cs_new_protected:Nn \__stex_refs_local_ref:n {
1862   \cs_if_exist:cTF{autoref}{
1863     \l_stex_key_pre_tl \autoref{#1} \l_stex_key_post_tl
1864   }{
1865     \message{^^J^^J
1866       sTeX~Warning: \c_backslash_str sref ~ with ~ local ~ target ~ used ~ without ~ hyperr
1867       ^^J^^J}
1868     \l_stex_key_pre_tl \ref{#1} \l_stex_key_post_tl
1869   }
1870 }
1871
1872 \cs_new_protected:Nn \__stex_refs_remote:nn {
1873   \str_set:Nn \l__stex_refs_key { #2 }
1874   \tl_set:Nn \l__stex_refs_in { #1 }
1875   \exp_args:No \stex_in_archive:nn \l_stex_key_archive_str {
1876     \exp_args:NNo \stex_document_uri_from_archive_file:Nn \l__stex_refs_uri {\l_stex_key_fi
1877     \stex_debug:nn{sref}{Checking~#2~from~\stex_use_document_uri:N \l__stex_refs_uri}
1878     \tl_if_eq:NNTF \l__stex_refs_uri \c_stex_main_document_uri {
1879       \stex_debug:nn{sref}{From~current~document;~delegating~to~\string\autoref}
1880     }
1881     \__stex_refs_local:n{ #2 }
1882   }{
1883     \str_set:Ne \l__stex_refs_uri { \stex_use_document_uri:N \l__stex_refs_uri }
1884     \str_set:Ne \l__stex_refs_file { \exp_args:No \stex_source_path:n {\l_stex_key_file_s
1885     \exp_args:No \IfFileExists \l__stex_refs_file {
1886       \__stex_refs_find:
1887     }{}
1888     \tl_if_empty:NTF \l__stex_refs_tmp {
1889       \stex_debug:nn{sref}{Not~found}
1890       \tl_if_empty:NTF \l_stex_key_fallback_tl { ??? }{ \l_stex_key_fallback_tl }
1891     }{
1892       \str_set_eq:NN \l__stex_refs_uri \l__stex_refs_tmp
1893       \stex_debug:nn{sref}{Found~ \l__stex_refs_uri}
1894       \__stex_refs_maybe_in:
1895     }
1896   }
1897 }
1898
1899 \cs_new_protected:Nn \__stex_refs_find: {
1900   \setbox0\vbox\bgroup
1901   \cs_set_eq:NN \STeXInternalSRefLabel \__stex_refs_check_i:nnnn
1902   \use:c{@ @ input}{\l__stex_refs_file}
1903   \exp_args:NNe \use:nn
1904   \egroup {
1905     \tl_set:Nn \exp_not:N \l__stex_refs_tmp { \l__stex_refs_tmp }

```

```

1906 }
1907 }
1908
1909 \cs_new_protected:Nn \__stex_refs_check_i:nnnn {
1910   \exp_args:No \str_if_eq:nnT{ \l__stex_refs_key }{#1}{
1911     \exp_args:Nno \stex_str_if_starts_with:nnT{#2}{\l__stex_refs_uri}{
1912       \str_set:Nn \l__stex_refs_tmp {#2}
1913     }
1914   }
1915 }
1916 }
1917
1918 \cs_new_protected:Nn \__stex_refs_maybe_in: {
1919   \tl_if_empty:NTF\l__stex_refs_in {
1920     \cs_if_exist:cTF{r@\l__stex_refs_uri}{
1921       \exp_args:No \__stex_refs_local_ref:n \l__stex_refs_uri
1922     }{
1923       \tl_if_empty:NTF \l__stex_refs_default_file {
1924         \exp_args:No \__stex_refs_local_ref:n \l__stex_refs_uri
1925       }{
1926         \tl_set_eq:NN \l_stex_key_title_tl \l__stex_refs_default_title
1927         \str_set_eq:NN \l__stex_refs_file \l__stex_refs_default_file
1928         \str_set_eq:NN \l_stex_key_archive_str \l__stex_refs_default_archive
1929         \str_set_eq:NN \l_stex_key_file_str \l__stex_refs_default_relpath
1930         \str_if_eq:NNTF \l__stex_refs_default_file \c__stex_refs_iow_str {
1931           \exp_args:No \__stex_refs_local_ref:n \l__stex_refs_uri
1932         }{
1933           \stex_debug:nn{sref}{in~title=\exp_args:No\exp_not:n\l_stex_key_title_tl,file=\l__stex_refs_in:
1934           }
1935         }
1936       }
1937     }
1938   }{
1939     \stex_debug:nn{sref}{in~\exp_args:No\exp_not:n\l__stex_refs_in,archive=\l_stex_key_archive_str
1940     \exp_args:No \stex_in_archive:nn \l_stex_key_archive_str {
1941       \str_set:Ne \l__stex_refs_file { \exp_args:No \stex_source_path:n {\l_stex_key_file_str
1942     }
1943     \exp_args:Nno \stex_keys_set:nn{ sref / 2 }{ \l__stex_refs_in }
1944     \__stex_refs_in:
1945   }
1946 }
1947
1948 \cs_new_protected:Nn \__stex_refs_in: {
1949   \tl_clear:N \l__stex_refs_tmp
1950   \exp_args:No \IfFileExists \l__stex_refs_file {
1951     \__stex_refs_find_in:
1952   }{
1953     \stex_debug:nn{sref}{not~found:~\l__stex_refs_file}
1954   }
1955   \tl_if_empty:NTF\l__stex_refs_tmp{
1956     \stex_debug:nn{sref}{Not~found}
1957     \tl_if_empty:NTF \l_stex_key_fallback_tl { ??? }{ \l_stex_key_fallback_tl }
1958   }{
1959     \stex_debug:nn{sref}{found:~\meaning\l__stex_refs_tmp}

```

```

1960 \tl_set:N\l__stex_refs_tmp {
1961   \exp_after:wN \__stex_refs_in_text:nn \l__stex_refs_tmp
1962 }
1963 \stex_if_html_backend:TF{
1964   \exp_args:No \stex_in_archive:nn \l_stex_key_archive_str {
1965     \exp_args:NNo \stex_document_uri_from_archive_file:Nn \l__stex_refs_in \l_stex_key_
1966     \exp_args:Ne \stex_annotate:nn{data-ftml-sref={\l__stex_refs_uri},data-ftml-sref-in
1967       \l__stex_refs_tmp
1968   }
1969 }
1970 }{
1971   \cs_if_exist:NT \href {\exp_args:No \href \l__stex_refs_uri }{\l__stex_refs_tmp
1972 }
1973 }
1974 }
1975
1976 \cs_new:Nn \__stex_refs_in_text:nn {
1977   \__stex_refs_in_text:w #1 \__stex_refs_stop:
1978   \tl_if_empty:NF{#2}{~(\exp_not:n{#2})}
1979   \tl_if_empty:NF \l_stex_key_title_tl {
1980     {}~in~\exp_args:No \exp_not:n \l_stex_key_title_tl
1981   }
1982 }
1983
1984 \cs_new:Npn \__stex_refs_in_text:w #1.#2 \__stex_refs_stop: {
1985   \__stex_refs_uppercase:n #1~#2~
1986 }
1987
1988 \cs_new:Nn \__stex_refs_uppercase:n { \uppercase{#1}}
1989
1990 \cs_new_protected:Nn \__stex_refs_find_in: {
1991   \stex_debug:nn{sref}{finding~\l__stex_refs_uri;~in~\l__stex_refs_file...?}
1992   \setbox0\vbox\bgroup
1993   \catcode'\J=9\relax
1994   \cs_set_eq:NN \STeXInternalSRefLabel \__stex_refs_check_in:nnnn
1995   \use:c{@ @ input}{\l__stex_refs_file}
1996   \exp_args:NNe \use:nn
1997   \egroup {
1998     \tl_set:Nn \exp_not:N \l__stex_refs_tmp { \exp_args:No \exp_not:n \l__stex_refs_tmp }
1999   }
2000 }
2001
2002 \cs_new_protected:Nn \__stex_refs_check_in:nnnn {
2003   \exp_args:No \str_if_eq:nnT{ \l__stex_refs_uri }{#2}{
2004     \tl_set:Nn \l__stex_refs_tmp { {#3}{#4} }
2005     \endinput
2006   }
2007 }

```

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

sfunction
TODO

```

2008 %\int_new:N \l__stex_refs_unnamed_counter_int
2009

```

```

2010
2011
2012
2013 \cs_new:Npn \STeXInternalSRefLabel #1 #2 #3 #4 {}
2014
2015 \cs_new_protected:Npn \STeXInternalNewSRefLabel #1 #2 {
2016   \tl_gset:ce{ g_stex_sref_label_ #1 }{ \tl_to_str:n{ #2 } }
2017 }
2018
2019 \tl_new:N \l__stex_refs_default_title
2020 \str_new:N \l__stex_refs_default_file
2021 \str_new:N \l__stex_refs_default_archive
2022 \str_new:N \l__stex_refs_default_relpath
2023
2024 \newcommand\srefsetin[3][]{
2025   \str_set:Ne \l__stex_refs_default_relpath { #2 }
2026   \tl_if_empty:nTF{#1}{
2027     \str_set:Ne \l__stex_refs_default_archive { \stex_archive_id:N \c_stex_main_archive }
2028   }{
2029     \str_set:Nn \l__stex_refs_default_archive { #1 }
2030   }
2031   \exp_args:No \stex_in_archive:nn \l__stex_refs_default_archive {
2032     \str_set:Ne \l__stex_refs_default_file { \exp_args:No \stex_source_path:n {\l__stex_ref
2033   }
2034   \str_set:Nn \l__stex_refs_default_title { #3 }
2035 }
2036 \NewDocumentCommand \extref { 0{} m m }{???}
2037
2038 \cs_new_protected:Nn \stex_ref_new_symbol:n {}
2039
2040
2041 \NewDocumentCommand \srefsym { m m }{
2042   \stex_get_symbol:n { #1 }
2043   \exp_args:Ne \srefsymuri { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }{ #2 }
2044 }
2045
2046 \cs_new_protected:Npn \srefsymuri #1 {
2047   \cs_if_exist:cTF{r@\tl_to_str:n{#1}}{
2048     \cs_if_exist:NT \hyperlink { \hyperlink{#1} }
2049   }{
2050     \cs_if_exist:NT \href {\href{#1} }
2051   }
2052 }

```

30.3 Inputting From MathHub Resources

```

2053 <@@=stex_inputs>

\inputref

2054 \NewDocumentCommand \inputref { 0{} m }{
2055   \stex_in_archive:nn{#1}{
2056     \stex_document_uri_from_archive_file:Nn \l__stex_inputs_uri {#2}
2057     \__stex_inputs_ref:n{#2}

```



```

2058 }
2059 }
2060
2061 \stex_if_html_backend:TF{
2062   \str_set:Nn \c__stex_inputs_ref_pre_str {<span~data-ftml-inputref="}
2063   \str_set:Nn \c__stex_inputs_ref_post_str {"~><!--<!--></span>}
2064   \cs_new_protected:Nn \__stex_inputs_ref_i:n {
2065     \IfFileExists{\stex_source_path:n {#1}}{
2066       %\relax
2067       %\noindent\vbox{
2068         \exp_args:N\stex_html_literal:n{
2069           \c__stex_inputs_ref_pre_str
2070           \stex_use_document_uri:N\l__stex_inputs_uri
2071           \c__stex_inputs_ref_post_str
2072         }
2073       %}
2074     }{
2075       \stex_input_with_hooks:Ne\l__stex_inputs_uri {\stex_source_path:n {#1}}
2076     }
2077   }
2078   \cs_new_protected:Nn \__stex_inputs_ref:n {
2079     \ifnum\@listdepth>0
2080       \noindent\vbox{\vskip-26pt\relax\__stex_inputs_ref_i:n{#1}}\par
2081       %\begin{stex_env_node}{div}{style:foo=bar}\vskip-700pt\end{stex_env_node}
2082     \else
2083       \__stex_inputs_ref_i:n{#1}
2084     \fi
2085   }
2086
2087   {}{
2088     \cs_new_protected:Nn \__stex_inputs_ref:n {
2089       \begingroup
2090         \inputreftrue
2091         \let \l_stex_metatheory_uri \c_stex_default_metatheory
2092         %\seq_clear:N \l_stex_all_modules_seq
2093         \stex_undefine:N \l_stex_current_module_uri
2094         \stex_debug:nn{inputref}{Inputrefing~document~\stex_use_document_uri:N \l__stex_input
2095         \stex_input_with_hooks:Ne\l__stex_inputs_uri{\stex_source_path:n {#1}}
2096       \endgroup\medskip
2097     }
2098   }

```

(End of definition for \inputref. This function is documented on page 140.)

sfunction

\mhinput

```

2099 \NewDocumentCommand \mhinput { 0{} m}{
2100   \stex_in_archive:nn {#1} {
2101     \stex_document_uri_from_archive_file:Nn \l__stex_inputs_uri {#2}
2102     \ifinputref
2103       \stex_input_with_hooks:Ne\l__stex_inputs_uri{\stex_source_path:n {#2}}
2104     \else
2105       \inputreftrue
2106       \stex_input_with_hooks:Ne\l__stex_inputs_uri{\stex_source_path:n {#2}}

```

```

2107     \inputreffalse
2108   \fi
2109 }
2110 }

```

(End of definition for \mhinput. This function is documented on page 140.)

sfunction

\ifinputref

```

2111 \newif \ifinputref \inputreffalse

```

(End of definition for \ifinputref. This function is documented on page 140.)

sfunction

\IfInputref

```

2112 \stex_if_html_backend:TF{
2113   \newcommand \IfInputref[2]{
2114     \stex_annotate:nn{data-ftml-ifinputref=true}{#1}
2115     \stex_annotate:nn{data-ftml-ifinputref=false}{#2}
2116   }
2117 }{
2118   \newcommand \IfInputref[2]{
2119     \ifinputref #1 \else #2 \fi
2120   }
2121 }

```

(End of definition for \IfInputref. This function is documented on page 140.)

sfunction

\libinput

```

2122 \newcommand \libinput [2] [] {
2123   \stex_in_archive:nn{#1}{
2124     \__stex_inputs_up_archive:nn{#2}{tex}
2125     \stex_debug:nn{lib}{Result:~\seq_use:Nn \l__stex_inputs_libinput_files_seq ,}
2126     \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
2127       \msg_error:nnnn{stex}{error/nofile}{\libinput}{#2.tex}
2128     }{
2129       \seq_map_function:NN \l__stex_inputs_libinput_files_seq \input
2130     }
2131   }
2132 }

```

(End of definition for \libinput. This function is documented on page 140.)

sfunction

\libusepackage

```

2133 \newcommand\libusepackage[2] []{
2134   \__stex_inputs_up_archive:nn{#2}{sty}
2135   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2136     \str_set:Ne \l__stex_inputs_tmp_str {\seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
2137     \exp_args:Nne \use:n {\usepackage[#1]} {
2138       \str_range:Nnn\l__stex_inputs_tmp_str 1 {-5}
2139     }
2140   }{
2141     \msg_fatal:nnnn{stex}{error/nofile}{\libusepackage}{#2.sty}

```

```

2142 }
2143 }

```

(End of definition for \libusepackage. This function is documented on page 140.)

sfunction

\libusetikzlibrary

```

2144 \newcommand \libusetikzlibrary [2] [] {
2145   \cs_if_exist:NF \usetikzlibrary {
2146     \msg_error:nnx{stex}{error/notikz}{\tl_to_str:n{\libusetikzlibrary}}
2147   }
2148   \tl_if_empty:NTF{#1}{
2149     \__stex_inputs_usetikzlibrary:n{#2}
2150   }{
2151     \stex_in_archive:nn{#1}{\__stex_inputs_usetikzlibrary:n{#2}}
2152   }
2153 }
2154
2155 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary:n{
2156   \__stex_inputs_up_archive:nn{tikzlibrary#1}{code.tex}
2157   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2158     \exp_args:Nne \__stex_inputs_usetikzlibrary_i:nn{#1}{ \seq_item:Nn \l__stex_inputs_libi
2159   }{
2160     \msg_fatal:nnnn{stex}{error/nofile}{\libusetikzlibrary}{tikzlibrary#1.code.tex}
2161   }
2162 }
2163
2164 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary_i:nn {
2165   \pgfkeys@spdef\pgf@temp{#1}
2166   \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
2167   \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgfu
2168   \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
2169   \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
2170   \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
2171   \catcode'\@=11
2172   \catcode'\|=12
2173   \catcode'\$=3
2174   \pgfutil@InputIfFileExists{#2}{\}{}
2175   \catcode'\@=\csname tikz@library@#1@atcode\endcsname
2176   \catcode'\|=\csname tikz@library@#1@barcode\endcsname
2177   \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
2178 }

```

(End of definition for \libusetikzlibrary. This function is documented on page 140.)

sfunction

\addmhbibresource

```

2179 \newcommand \addmhbibresource [2] [] {
2180   \stex_in_archive:nn{#1}{
2181     \__stex_inputs_up_archive:nn{#2}{bib}
2182     \stex_debug:nn{lib}{Result:-\seq_use:Nn \l__stex_inputs_libinput_files_seq ,}
2183     \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
2184       \msg_error:nnnn{stex}{error/nofile}{\addmhbibresource}{#2.bib}
2185     }{

```

```

2186     \seq_map_function:NN \l__stex_inputs_libinput_files_seq \addbibresource
2187   }
2188 }
2189 }

```

(End of definition for \addmhbibresource. This function is documented on page 141.)

sfunction

__stex_inputs_up_archive:nn Iterates up the current archive's directory in search for lib files with name #1.#2

```

2190 \cs_new_protected:Nn \__stex_inputs_up_archive:nn {
2191   \tl_if_exist:NF \l_stex_current_archive {
2192     \msg_error:nnn{stex}{error/notinarchive}\libinput
2193   }
2194   \seq_clear:N \l__stex_inputs_libinput_files_seq
2195   \seq_set_split:Nne \l__stex_inputs_path_seq / {\stex_archive_path:N \l_stex_current_archi
2196   \seq_set_split:Nne \l__stex_inputs_id_seq / {\stex_archive_id:N \l_stex_current_archive}
2197   \seq_map_inline:Nn \l__stex_inputs_id_seq {
2198     \seq_pop_right:NN \l__stex_inputs_path_seq \l_tmpa_tl
2199   }
2200   \stex_debug:nn{lib}{Checking-#1~in~\seq_use:Nn \l__stex_inputs_path_seq /}
2201
2202   \bool_while_do:nn { ! \seq_if_empty_p:N \l__stex_inputs_id_seq }{
2203     \str_set:Ne \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / meta-
2204     \stex_debug:nn{lib}{Checking~\l__stex_inputs_path_str}
2205     \IfFileExists{ \l__stex_inputs_path_str }{
2206       \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_
2207       \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
2208     }
2209     }{}
2210     \seq_pop_left:NN \l__stex_inputs_id_seq \l__stex_inputs_path_str
2211     \seq_put_right:No \l__stex_inputs_path_seq \l__stex_inputs_path_str
2212   }
2213
2214   \str_set:Ne \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / lib / #
2215   \stex_debug:nn{lib}{Checking~\l__stex_inputs_path_str}
2216   \IfFileExists{ \l__stex_inputs_path_str }{
2217     \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_st
2218     \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
2219   }
2220   }{}
2221 }

```

(End of definition for __stex_inputs_up_archive:nn.)

\mhgraphics

\cmhgraphics

```

2222 \ltx@ifpackageloaded{graphicx}{\use:n}{\AtEndOfPackageFile{graphicx}}{
2223   \define@key{Gin}{archive}{
2224     \stex_in_archive:nn{#1}{}
2225     \str_set:Ne \Gin@mhrepos {
2226       \stex_source_path:nn{#1}{}
2227     }
2228   }
2229   \providecommand\mhgraphics[2][]{
2230     \str_set:Ne \Gin@mhrepos {

```

```

2231     \stex_source_path:n{}
2232   }
2233   \setkeys{Gin}{#1}
2234   \includegraphics[#1]{ \Gin@mhrepos / #2 }
2235 }
2236 \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
2237 }

```

(End of definition for \mhgraphics and \cmhgraphics. These functions are documented on page 141.)

sfunction

\lstinputmhlisting
\clstinputmhlisting

```

2238 \ltx@ifpackageloaded{listings}{\use:n}{\AtEndOfPackageFile{listings}}{
2239   \define@key{lst}{archive}{
2240     \stex_in_archive:nn{#1}{}
2241     \str_set:Ne \lst@mhrepos{
2242       \stex_source_path:nn{#1}{}
2243     }
2244   }
2245   \newcommand\lstinputmhlisting[2] [] {%
2246     \str_set:Ne \lst@mhrepos{
2247       \stex_source_path:n{}
2248     }
2249     \setkeys{lst}{#1}%
2250     \lstinputlisting[#1]{\lst@mhrepos / #2}}
2251   \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
2252 }

```

(End of definition for \lstinputmhlisting and \clstinputmhlisting. These functions are documented on page 141.)

sfunction

\mhtikzinput
\cmhtikzinput

```

2253 \ltx@ifpackageloaded{tikzinput}{\use:n}{\AtEndOfPackageFile{tikzinput}}{
2254   \define@key{Gin}{archive}{
2255     \stex_in_archive:nn{#1}{}
2256     \str_set:Ne \Gin@mhrepos{
2257       \stex_source_path:nn{#1}{}
2258     }
2259     \str_set:Ne \l__stex_inputs_gin_repo_str {#1}
2260   }
2261   \newcommand\mhtikzinput[2] [] {%
2262     \str_clear:N \l__stex_inputs_gin_repo_str
2263     \str_set:Ne \Gin@mhrepos{
2264       \stex_source_path:n{}
2265     }
2266     \setkeys{Gin}{#1}%
2267     \exp_args:No \stex_in_archive:nn \l__stex_inputs_gin_repo_str {
2268       \tikzinput[#1]{\Gin@mhrepos / #2}
2269     }
2270   }
2271   \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
2272 }

```

(End of definition for \mhtikzinput and \cmhtikzinput. These functions are documented on page 141.)

sfunction

Chapter 31

Modules

```
2273 (@@=stex_modules)
```

A module consists of four separate macros, storing its *symbols*, *incoming morphisms* (e.g. `\importmodules`), *notations*, and “initialization code”, respectively:

```
2274 \cs_new:Nn \__stex_modules_macro_short:nnn {
2275   #1 ? #2 ? #3
2276 }
2277 \cs_new:Nn \_stex_symbol_macro:N {
2278   c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _symbols_prop
2279 }
2280 \cs_new:Nn \_stex_symbol_macro:n {
2281   c_stex_module_ \__stex_modules_macro_short:nnn #1 _symbols_prop
2282 }
2283 \cs_new:Nn \_stex_morphisms_macro:N {
2284   c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _morphisms_prop
2285 }
2286 \cs_new:Nn \_stex_morphisms_macro:n {
2287   c_stex_module_ \__stex_modules_macro_short:nnn #1 _morphisms_prop
2288 }
2289 \cs_new:Nn \_stex_notations_macro:N {
2290   c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _notations_prop
2291 }
2292 \cs_new:Nn \_stex_notations_macro:n {
2293   c_stex_module_ \__stex_modules_macro_short:nnn #1 _notations_prop
2294 }
2295 \cs_new:Nn \_stex_module_code_macro:N {
2296   c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _code
2297 }
2298 \cs_new:Nn \_stex_module_code_macro:n {
2299   c_stex_module_ \__stex_modules_macro_short:nnn #1 _code
2300 }
```

`\l_stex_current_module_uri` initially undefined.

(End of definition for `\l_stex_current_module_uri`. This variable is documented on page 142.)

`\l_stex_all_modules_seq`

```
2301 \seq_new:N \l_stex_all_modules_seq
```

(End of definition for `\l_stex_all_modules_seq`. This variable is documented on page 142.)

smodule (*env*.)

```

2302 \tl_new:N \l_stex_metatheory_uri
2303
2304 \stex_keys_define:nnnn{smodule}{
2305   \str_clear:N \l_stex_key_sig_str
2306 }{
2307   meta .code:n = {
2308     \tl_if_empty:nTF {#1}{
2309       \tl_clear:N \l_stex_metatheory_uri
2310     }{
2311       \stex_uri_from_pair:Nn \l_stex_metatheory_uri { #1 }
2312     }
2313   },
2314   %ns .code:n = {
2315     % \stex_uri_resolve:Ne \l_stex_current_ns_uri { #1 }
2316     %} ,
2317   %lang .code:n = {
2318     % \stex_set_language:n { #1 }
2319     %} ,
2320   sig .str_set_x:N = \l_stex_key_sig_str ,
2321   %creators .code:n = {} , % todo ?
2322   %contributors .code:n = {} , % todo ?
2323 }{id, title, style, deprecate}
2324
2325 \stex_if_html_backend:T {
2326   \cs_new_protected:Nn \__stex_modules_html_annots: {
2327     \exp_args:Ne \stex_annotate_env {
2328       data-ftml-module={\stex_module_uri_name:N \l_stex_current_module_uri}
2329       %data-ftml-language={ \l_stex_current_language_str},
2330       \str_if_empty:NF\l_stex_key_sig_str {, data-ftml-signature={\l_stex_key_sig_str}}
2331       \tl_if_empty:NF \l_stex_metatheory_uri {,
2332         data-ftml-metatheory={\stex_use_module_uri:N \l_stex_metatheory_uri}
2333       }
2334     }
2335     \stex_annotate_invisible:n{}
2336   }
2337 }
2338
2339 \stex_new_stylable_env:nnnnnn {module} {0{} m}{
2340   \stex_keys_set:nn { smodule }{ #1 }
2341   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
2342   \tl_if_empty:NF \thistitle {
2343     \exp_args:No \stexdoctitle \thistitle
2344   }
2345   \stex_module_setup:n { #2 }
2346
2347   \stex_if_do_html:T \__stex_modules_html_annots:
2348   \stex_if_smsmode:F {
2349     \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_current_module_uri }
2350     \str_set:Nn \thismodulename {#2}
2351     \stex_style_apply:
2352   }
2353   \stex_smsmode_do:
2354 }{

```

```

2355 \stex_close_module:
2356 \stex_if_smsmode:F \stex_style_apply:
2357 \stex_if_do_html:T{ \endstex_annotate_env }
2358 }{}{}{s}

```

\stex_module_setup:n

```

2359 \cs_new_protected:Npn \stex_module_setup:n {
2360   \stex_if_in_module:TF \__stex_modules_nested:n \__stex_modules_top:n
2361 }
2362
2363 \cs_new_protected:Nn \__stex_modules_top:n {
2364   \str_if_empty:NTF \l_stex_key_sig_str
2365     \stex_module_setup_top_nosig:n \__stex_modules_top_sig:n {#1}
2366   \g_stex_every_module_tl
2367   \tl_if_empty:NF \l_stex_metatheory_uri {
2368     \stex_debug:nn{modules}{loading~metatheory~\stex_use_module_uri:N \l_stex_metatheory_uri}
2369     \__stex_modules_load_meta:
2370   }
2371 }

```

(End of definition for \stex_module_setup:n. This function is documented on page 142.)

sfunction

\stex_module_setup_top_nosig:n

```

2372 \cs_new_protected:Nn \stex_module_setup_top_nosig:n {
2373   \stex_metagroup_new:
2374   \tl_set:Ne \l__stex_modules_uri { \stex_new_module_uri:n {#1} }
2375   \stex_debug:nn{module}{URI:\stex_use_module_uri:N \l__stex_modules_uri}
2376   \exp_args:No \stex_if_module_exists:NTF \l__stex_modules_uri {
2377     \stex_debug:nn{modules}{(already exists)}
2378   }{
2379     \tl_gclear:c{ \stex_module_code_macro:N \l__stex_modules_uri }
2380     \prop_gclear:c{ \stex_morphisms_macro:N \l__stex_modules_uri }
2381     \prop_gclear:c{ \stex_symbol_macro:N \l__stex_modules_uri }
2382     \prop_gclear:c{ \stex_notations_macro:N \l__stex_modules_uri }
2383   }
2384   \tl_set_eq:NN \l_stex_current_module_uri \l__stex_modules_uri
2385   \seq_put_right:No \l_stex_all_modules_seq \l__stex_modules_uri
2386 }

```

(End of definition for \stex_module_setup_top_nosig:n. This function is documented on page 142.)

sfunction

__stex_modules_load_meta: loads the metatheory:

```

2387 \bool_new:N \l_stex_in_meta_bool
2388
2389 \cs_new_protected:Nn \__stex_modules_load_meta: {
2390   \exp_after:wN \stex_execute_in_module:n \exp_after:wN {
2391     \exp_after:wN \__stex_modules_load_meta_i:n \exp_after:wN { \l_stex_metatheory_uri }
2392   }
2393 }
2394
2395 \cs_new_protected:Nn \__stex_modules_load_meta_i:n {
2396   \bool_if:NTF \l_stex_in_meta_bool {

```



```

2397 \stex_activate_module:n {#1}
2398 }{
2399 \bool_set_true:N \l_stex_in_meta_bool
2400 \stex_activate_module:n {#1}
2401 \bool_set_false:N \l_stex_in_meta_bool
2402 }
2403 }

```

(End of definition for _stex_modules_load_meta:.)

_stex_modules_top_sig:n sets up a new module with a signature:

```

2404 \cs_new_protected:Nn \_stex_modules_top_sig:n {
2405 \stex_debug:nn{modules}{Signature:\l_stex_key_sig_str}
2406 \stex_metagroup_new:
2407 \tl_set:Nc \l__stex_modules_uri { \stex_new_module_uri:n {#1} }
2408 \stex_debug:nn{module}{URI:\stex_use_module_uri:N \l__stex_modules_uri}
2409 \stex_if_module_exists:NTF \l__stex_modules_uri {
2410 \stex_debug:nn{modules}{(already exists)}
2411 \stex_activate_module:N \l__stex_modules_uri
2412 }{
2413 \stex_debug:nn{modules}{(needs loading)}
2414 \_stex_modules_load_sig:
2415 }
2416 \tl_set_eq:NN \l_stex_current_module_uri \l__stex_modules_uri
2417 }
2418
2419 \cs_new_protected:Nn \_stex_modules_load_sig: {
2420 \stex_file_split_off_lang:NN \l__stex_modules_sigfile \g_stex_current_file
2421 \tl_set:Nc \l__stex_modules_sig {
2422 \exp_args:NNo \stex_document_uri_with_language:Nn
2423 \l_stex_current_document_uri \l_stex_key_sig_str
2424 }
2425 \stex_debug:nn{modules}{Loading~signature~\stex_use_document_uri:N \l__stex_modules_sig;~
2426 \exp_args:NNe \stex_file_in_smsmode:Nn \l__stex_modules_sig {
2427 \stex_file_use:N \l__stex_modules_sigfile . \l_stex_key_sig_str . tex
2428 }
2429 \stex_activate_module:N \l__stex_modules_uri
2430 }

```

(End of definition for _stex_modules_top_sig:n.)

_stex_modules_nested:n sets up a new nested module:

```

2431 \cs_new_protected:Nn \_stex_modules_nested:n {
2432 \stex_metagroup_new:
2433 \stex_debug:nn{module}{Nested~module~#1~in~\stex_use_module_uri:N \l_stex_current_module_
2434 \tl_set:Nc \l__stex_modules_uri { \stex_new_module_uri:n{#1} }
2435 \stex_debug:nn{module}{URI:\stex_use_module_uri:N \l__stex_modules_uri}
2436 \exp_args:No \stex_if_module_exists:NTF \l__stex_modules_uri {
2437 \stex_debug:nn{modules}{(already exists)}
2438 }{
2439 \tl_gclear:c{ \_stex_module_code_macro:N \l__stex_modules_uri }
2440 \prop_gclear:c{ \_stex_morphisms_macro:N \l__stex_modules_uri }
2441 \prop_gclear:c{ \_stex_symbol_macro:N \l__stex_modules_uri }
2442 \prop_gclear:c{ \_stex_notations_macro:N \l__stex_modules_uri }
2443 }

```

```

2444 \tl_set_eq:NN \l_stex_current_module_uri \l__stex_modules_uri
2445 \seq_put_left:No \l_stex_all_modules_seq \l__stex_modules_uri
2446 }

```

(End of definition for __stex_modules_nested:n.)

\stex_close_module:

```

2447 \cs_new_protected:Nn \stex_close_module: {
2448   \bool_if:NT \c_stex_persist_write_mode_bool \__stex_modules_persist_module:
2449   \stex_debug:nn{module}{
2450     Closing~module~\stex_use_module_uri:N \l_stex_current_module_uri^^J
2451     Code:~\cs_meaning:c{ \stex_module_code_macro:N \l_stex_current_module_uri }^^J
2452     Imports:~ \exp_after:wN \prop_to_keyval:N \cs:w \stex_morphisms_macro:N \l_stex_current_module_uri
2453     Declarations:~ \exp_after:wN \prop_to_keyval:N \cs:w \stex_symbol_macro:N \l_stex_current_module_uri
2454     Notations:~ \exp_after:wN \prop_to_keyval:N \cs:w \stex_notations_macro:N \l_stex_current_module_uri
2455   }
2456 }

```

Persist and restore modules in the .sms-file:

```

2457 \cs_new_protected:Nn \__stex_modules_persist_module: {
2458   \stex_persist:e {
2459     \__stex_modules_restore_module:nnnn {\l_stex_current_module_uri}{
2460       \exp_after:wN \prop_to_keyval:N \cs:w
2461       \stex_morphisms_macro:N \l_stex_current_module_uri
2462       \cs_end:
2463     }{
2464       \exp_after:wN \prop_to_keyval:N \cs:w
2465       \stex_symbol_macro:N \l_stex_current_module_uri
2466       \cs_end:
2467     }{
2468       \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
2469       \exp_after:wN \exp_after:wN \exp_after:wN
2470       { \cs:w \stex_module_code_macro:N \l_stex_current_module_uri \cs_end: }
2471     }{}
2472     \prop_map_function:cN{\stex_notations_macro:N \l_stex_current_module_uri}
2473     \__stex_modules_persist_not_i:nn
2474     \exp_not:N \STEXRestoreNotEnd {}
2475   }
2476 }
2477
2478 \cs_new:Nn \__stex_modules_persist_not_i:nn {
2479   \exp_not:n{#2}
2480 }
2481
2482
2483 \cs_new:Nn \__stex_modules_stringify_uri:nnn {
2484   {\tl_to_str:n{#1}}
2485   {\tl_to_str:n{#2}}
2486   {\tl_to_str:n{#3}}
2487 }
2488 \cs_new:Nn \__stex_modules_stringify_uri:nnnn {
2489   {\tl_to_str:n{#1}}
2490   {\tl_to_str:n{#2}}
2491   {\tl_to_str:n{#3}}
2492   {\tl_to_str:n{#4}}

```

```

2493 }
2494
2495 \cs_new_protected:Nn \__stex_modules_restore_module:nnnn {
2496   \tl_set:Nc \l__stex_modules_uri { \__stex_modules_stringify_uri:nnn #1 }
2497   \stex_debug:nn{persist}{restoring~\stex_use_module_uri:N \l__stex_modules_uri}
2498   \prop_gset_from_keyval:cn{ \stex_morphisms_macro:N \l__stex_modules_uri }{#2}
2499   \prop_gset_from_keyval:cn{ \stex_symbol_macro:N \l__stex_modules_uri }{#3}
2500   \prop_map_inline:cn{ \stex_symbol_macro:N \l__stex_modules_uri }{
2501     \stex_ref_new_symbol:n{#1?##1}
2502   }
2503   \def \l_tmpa_tl { #4 }
2504   \exp_args:Nno \tl_gset:cn{ \stex_module_code_macro:N \l__stex_modules_uri } \l_tmpa_tl
2505   \prop_gc_clear:c{ \stex_notations_macro:N \l__stex_modules_uri }
2506   \group_begin:
2507     \catcode' =8\relax
2508     \catcode' =12\relax
2509     \__stex_modules_restore_notcs:n
2510   }
2511
2512 \quark_new:N \STEXRestoreNotcsEnd
2513
2514 \cs_new_protected:Nn \__stex_modules_restore_notcs:n {
2515   \__stex_modules_restore_notcs_i:n
2516 }
2517
2518 \cs_new_protected:Nn \__stex_modules_restore_notcs_i:n {
2519   \tl_if_eq:nnTF{#1}{\STEXRestoreNotcsEnd}{
2520     \group_end:
2521   }{
2522     \__stex_modules_restore_notcs_ii:nnnn {#1}
2523   }
2524 }
2525
2526 \cs_new_protected:Nn \__stex_modules_restore_notcs_ii:nnnn {
2527   \tl_set:Nn \l__stex_modules_tl {{#4}{#5}}
2528   % eliminate ## => #
2529   \exp_after:wN \def \exp_after:wN \l__stex_modules_tl \exp_after:wN {\l__stex_modules_tl}
2530   \exp_args:NNe\use:nn\prop_gput:cnn{
2531     { \stex_notations_macro:N \l__stex_modules_uri }
2532     { \stex_use_symbol_uri:n{#1}\tl_to_str:n{?#2}}{
2533       { \__stex_modules_stringify_uri:nnnn #1 }{ \tl_to_str:n{#2}}{#3}
2534     }
2535   }
2536 }
2537 \__stex_modules_restore_notcs_i:n
2538 }

```

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

sfunction

\stex_every_module:n

```

2539 \tl_new:N \g_stex_every_module_tl
2540 \cs_new_protected:Nn \stex_every_module:n {
2541   \tl_gput_right:Nn \g_stex_every_module_tl { #1 }
2542 }

```

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

sfunction

`\stex_do_up_to_module:n`

`\stex_do_up_to_module:e`

```
2543 \cs_new_protected:Nn \stex_do_up_to_module:n {
2544   \stex_metagroup_do_in:n {#1}
2545 }
2546 \cs_generate_variant:Nn \stex_do_up_to_module:n {e}
```

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

sfunction

`\stex_execute_in_module:n`

`\stex_execute_in_module:e`

```
2547 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:TF {
2548   \tl_gput_right:cn { \_stex_module_code_macro:N \l_stex_current_module_uri } { #1 }
2549   \stex_debug:nn{modules}{extending~activation~for~\stex_use_module_uri:N \l_stex_current_m
2550   \stex_do_up_to_module:n { #1 }
2551 }{ #1 }}
2552 \cs_generate_variant:Nn \stex_execute_in_module:n {e}
```

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

sfunction

`\stex_if_in_module_p:`

`\stex_if_in_module:TF`

```
2553 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
2554   \tl_if_exist:NTF \l_stex_current_module_uri
2555   \prg_return_true: \prg_return_false:
2556 }
```

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

sfunction

`\stex_if_module_exists_p:N`

`\stex_if_module_exists:NTF`

`\stex_if_module_exists_p:n`

`\stex_if_module_exists:nTF`

```
2557 \prg_new_conditional:Nnn \stex_if_module_exists:N {p, T, F, TF} {
2558   \tl_if_exist:cTF { \_stex_module_code_macro:N #1 }
2559   \prg_return_true: \prg_return_false:
2560 }
2561 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
2562   \tl_if_exist:cTF { \_stex_module_code_macro:n {#1} }
2563   \prg_return_true: \prg_return_false:
2564 }
```

(End of definition for `\stex_if_module_exists:N` and `\stex_if_module_exists:n`. These functions are documented on page 143.)

sfunction

`\stex_activate_module:N`

`\stex_activate_module:n`

```
2565 \cs_new_protected:Nn \stex_activate_module:N {
2566   \exp_args:No \stex_activate_module:n #1
2567 }
2568
2569 \cs_new_protected:Nn \stex_activate_module:n {
2570   \seq_if_in:NnF \l_stex_all_modules_seq {#1} {
2571     \stex_debug:nn{modules}{Activating~module~\stex_use_module_uri:n{#1}}
```

```

2572 \seq_put_right:Nn \l_stex_all_modules_seq {#1}
2573 \stex_pseudogroup:nn{
2574   \tl_set:Nn \l_stex_current_module_uri {#1}
2575   \tl_if_exist:cF { \_stex_module_code_macro:N \l_stex_current_module_uri }{
2576     \msg_error:nnx{stex}{error/unknownmodule}{\stex_use_module_uri:n {#1} }
2577   }
2578   \_stex_activate_symbols:
2579   \_stex_activate_notations:
2580   \stex_debug:nn{modules}{Executing:~\expandafter\meaning\csname\_stex_module_code_macro:N \l_stex_current_module_uri }
2581   \use:c{ \_stex_module_code_macro:N \l_stex_current_module_uri }
2582 }{
2583   \stex_pseudogroup_restore:N \l_stex_current_module_uri
2584 }
2585 \stex_debug:nn{modules}{Activated~module~\stex_use_module_uri:n {#1} }
2586 }
2587 }

```

(End of definition for `\stex_activate_module:N` and `\stex_activate_module:n`. These functions are documented on page 143.)

sfunction

`\setmetatheory`

```

2588 \NewDocumentCommand \setmetatheory {0{} m}{
2589   \stex_debug:nn{metatheory}{Setting~metatheory~[#1]#2}
2590   \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
2591   \stex_debug:nn{metatheory}{Here:\stex_use_module_uri:N \l_stex_import_uri
2592 }
2593   \tl_set_eq:NN \l_stex_metatheory_uri \l_stex_import_uri
2594   \begingroup
2595     \stex_require_module:N \l_stex_metatheory_uri
2596   \endgroup
2597   \stex_smsmode_do:
2598 }

```

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

sfunction

`\STEXexport`

```

2599 \cs_new_protected:Npn \STEXexport {
2600   \ExplSyntaxOn
2601   \__stex_modules_export:n
2602 }
2603 \cs_new_protected:Nn \__stex_modules_export:n {
2604   \stex_ignore_spaces_and_pars:#1\ExplSyntaxOff
2605   \tl_gput_right:cn { \_stex_module_code_macro:N \l_stex_current_module_uri } {
2606     \stex_ignore_spaces_and_pars: #1
2607   }
2608   \stex_smsmode_do:
2609 }
2610
2611 \stex_deactivate_macro:Nn \STEXexport {module~environments}
2612 \stex_every_module:n {\stex_reactivate_macro:N \STEXexport}

```

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

sfunction

```

\stex_structural_feature_module:nn
\stex_structural_feature_module_end:
2613 \cs_new_protected:Nn \stex_structural_feature_module:nn {
2614   \stex_if_do_html:TF {
2615     \exp_args:Nne \begin{stex_annotate_env} {
2616       data-ftml-feature-#2={#1}
2617       \str_if_empty:NF \l_stex_macroname_str {,
2618         data-ftml-macroname={\l_stex_macroname_str}
2619     }
2620   }
2621   \stex_annotate_invisible:n{ }
2622 } \group_begin:
2623 \stex_module_setup:n {#1}
2624 }
2625
2626 \cs_new_protected:Nn \stex_structural_feature_module_end: {
2627   \tl_gset_eq:NN \g_stex_last_feature_str \l_stex_current_module_str
2628   \stex_close_module:
2629   \stex_if_do_html:TF{
2630     \end{stex_annotate_env}
2631   } \group_end:
2632 }

(End of definition for \stex_structural_feature_module:nn and \stex_structural_feature_module_end:.)
These functions are documented on page 143.)
sfunction

```

31.1 SMS Mode

```

2633 <@@=stex_smsmode>

\stex_sms_allow:N

2634 \tl_new:N \g__stex_smsmode_allowed_tl
2635
2636 \cs_new_protected:Nn \stex_sms_allow:N {
2637   \tl_gput_right:Nn \g__stex_smsmode_allowed_tl {#1}
2638 }

(End of definition for \stex_sms_allow:N. This function is documented on page 144.)
sfunction

\stex_sms_allow_escape:N

2639 \tl_new:N \g__stex_smsmode_allowed_escape_tl
2640
2641 \cs_new_protected:Nn \stex_sms_allow_escape:N {
2642   \tl_gput_right:Nn \g__stex_smsmode_allowed_escape_tl {#1}
2643 }

(End of definition for \stex_sms_allow_escape:N. This function is documented on page 144.)
sfunction

\stex_sms_allow_env:n

2644 \seq_new:N \g__stex_smsmode_allowedenvs_seq
2645
2646 \cs_new_protected:Nn \stex_sms_allow_env:n {

```

```

2647 \seq_gput_right:Ne \g__stex_smsmode_allowedenvs_seq {\tl_to_str:n{#1}}
2648 }

(End of definition for \stex_sms_allow_env:n. This function is documented on page 144.)
sfunction
Some initial allowed macros:
2649 \stex_sms_allow:N \makeatletter
2650 \stex_sms_allow:N \makeatother
2651 \stex_sms_allow:N \ExplSyntaxOn
2652 \stex_sms_allow:N \ExplSyntaxOff
2653 \stex_sms_allow:N \rustexBREAK
2654 \stex_sms_allow_env:n{smodule}
2655 \stex_sms_allow_escape:N \setmetatheory
2656 \stex_sms_allow_escape:N \STEXexport

\stex_sms_allow_import:Nn

2657 \tl_new:N \g__stex_smsmode_allowed_import_tl
2658 \tl_new:N \g__stex_smsmode_import_setup_tl
2659
2660 \cs_new_protected:Nn \stex_sms_allow_import:Nn {
2661   \tl_gput_right:Nn \g__stex_smsmode_allowed_import_tl {#1}
2662   \tl_gput_right:Nn \g__stex_smsmode_import_setup_tl {#2}
2663 }

(End of definition for \stex_sms_allow_import:Nn. This function is documented on page 144.)
sfunction

\stex_sms_allow_import_env:nn

2664 \seq_new:N \g__stex_smsmode_allowed_import_env_seq
2665
2666 \cs_new_protected:Nn \stex_sms_allow_import_env:nn {
2667   \seq_gput_right:Ne \g__stex_smsmode_allowed_import_env_seq {\tl_to_str:n{#1}}
2668   \tl_gput_right:Nn \g__stex_smsmode_import_setup_tl {#2}
2669 }
2670
2671 %\stex_sms_allow_import_env:nn{smodule}{}

(End of definition for \stex_sms_allow_import_env:nn. This function is documented on page 144.)
sfunction

\g_stex_sms_import_code

2672 \tl_new:N \g_stex_sms_import_code

(End of definition for \g_stex_sms_import_code. This variable is documented on page 144.)

\stex_if_smsmode_p:
\stex_if_smsmode:TF
2673 \bool_new:N \g__stex_smsmode_bool
2674
2675 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2676   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2677 }

(End of definition for \stex_if_smsmode:TF. This function is documented on page 144.)
sfunction

```

\stex_file_in_smsmode:Nn

```
2678
2679 \seq_new:N \l__stex_smsmode_cycles_seq
2680
2681 \cs_new_protected:Nn \stex_file_in_smsmode:Nn {
2682   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2683   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2684   \tl_clear:N \g_stex_sms_import_code
2685   \seq_if_in:NnF \l__stex_smsmode_cycles_seq {#2} {
2686     \group_begin:
2687       \seq_push:Nn \l__stex_smsmode_cycles_seq {#2}
2688       \let \l_stex_metatheory_uri \c_stex_default_metatheory
2689       \seq_clear:N \l_stex_all_modules_seq
2690       \stex_undefine:N \l_stex_current_module_uri
2691       \cs_set:Npn \stex_check_term:nn ##1 ##2 {}
2692       \stex_debug:nn{sms}{Loading~#2~in~\stex_use_document_uri:N #1}
2693       \stex_with_file_hooks:Nnn #1 {#2}{
2694         \__stex_smsmode_in_smsmode:n {
2695           \group_begin:
2696             \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_imports:N
2697             \g__stex_smsmode_import_setup_tl
2698             \__stex_smsmode_start_smsmode:n{#2}
2699           \group_end:
2700           %{
2701             \g_stex_sms_import_code
2702           %}
2703           %\group_begin:
2704             \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_normal:N
2705             \__stex_smsmode_start_smsmode:n{#2}
2706           %\group_end:
2707         }
2708       }
2709     \group_end:
2710   }
2711 }
2712
2713 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:n {
2714   \stex_suppress_html:n {
2715     \vbox_set:Nn \l_tmpa_box {
2716       \bool_set_true:N \g__stex_smsmode_bool
2717       #1
2718     }
2719   }
2720 }
2721
2722 \quark_new:N \q__stex_smsmode_break
2723
2724 \cs_new_protected:Nn \__stex_smsmode_start_smsmode:n {
2725   \everyeof{\q__stex_smsmode_break\exp_not:N}
2726   \let \stex_smsmode_do:\__stex_smsmode_smsmode_do:
2727   \exp_after:wN \exp_after:wN \exp_after:wN
2728   \stex_smsmode_do:
2729   \cs:w @ @ input\cs_end: "#1" \relax
2730 }
```


(End of definition for \stex_file_in_smsmode:Nn. This function is documented on page 144.)

sfunction

\stex_smsmode_do:

```

2731 \let\stex_smsmode_do:\relax
2732
2733 \cs_new_protected:Nn \__stex_smsmode_smsmode_do: { \__stex_smsmode_do:w }
2734
2735 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2736   \exp_args:Ne \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2737     %\__stex_smsmode_check_cs:NNn \__stex_smsmode_do_aux:N \__stex_smsmode_do:w { #1 }
2738     \__stex_smsmode_check_cs:N {#1}
2739   }{
2740     \__stex_smsmode_do:w #1
2741   }
2742 }
2743
2744 \cs_new_protected:Nn \__stex_smsmode_check_cs:N {
2745   %\stex_debug:nn{temp}{Checking \exp_not:N#1 \relax}
2746   \exp_after:wN\if\exp_after:wN\relax\noexpand #1 ~
2747   \exp_after:wN \__stex_smsmode_do_aux:N \exp_after:wN#1\else
2748   \exp_after:wN \__stex_smsmode_do:w \fi
2749 }
2750
2751 %\cs_new_protected:Nn \__stex_smsmode_check_cs:NNn {
2752 %  \message{^^JHERE: \tl_to_str:n{#1~and~#2~and~#3}}
2753 %  \exp_after:wN\if\exp_after:wN\relax\exp_not:N#3
2754 %  \exp_after:wN#1\exp_after:wN#3\else
2755 %  \exp_after:wN#2\fi
2756 %}
2757
2758 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2759   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
2760     \__stex_smsmode_do_aux_curr:N #1
2761   }
2762 }
2763
2764 \cs_new_protected:Nn \__stex_smsmode_do_aux_normal:N {
2765   \tl_if_in:NnTF \g__stex_smsmode_allowed_tl {#1} {
2766     \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}}
2767     #1\__stex_smsmode_do:w
2768   }{
2769     \tl_if_in:NnTF \g__stex_smsmode_allowed_escape_tl {#1} {
2770       \stex_debug:nn{sms}{Executing~escaped~\tl_to_str:n{#1}}
2771       #1
2772     }{
2773       \cs_if_eq:NNTF \begin #1 {
2774         \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowedenvs_seq
2775       }{
2776         \cs_if_eq:NNTF \end #1 {
2777           \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowedenvs_seq
2778         }{
2779           \__stex_smsmode_do:w
2780         }

```

```

2781     }
2782   }
2783 }
2784 }
2785
2786 \cs_new_protected:Nn \__stex_smsmode_do_aux_imports:N {
2787   \tl_if_in:NnTF \g__stex_smsmode_allowed_import_tl {#1} {
2788     \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}~in~import}
2789     #1
2790   }{
2791     \cs_if_eq:NNTF \begin #1 {
2792       \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowed_import_env_seq
2793     }{
2794       \cs_if_eq:NNTF \end #1 {
2795         \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowed_import_env_seq
2796       }{
2797         \__stex_smsmode_do:w
2798       }
2799     }
2800   }
2801 }
2802
2803 \cs_new_protected:Nn \__stex_smsmode_check_begin:Nn {
2804   \seq_if_in:NeTF #1 { \tl_to_str:n{#2} }{
2805     \stex_debug:nn{sms}{Environment~#2}
2806     \begin{#2}
2807   }{
2808     \__stex_smsmode_do:w
2809   }
2810 }
2811 \cs_new_protected:Nn \__stex_smsmode_check_end:Nn {
2812   \seq_if_in:NeTF #1 { \tl_to_str:n{#2} }{
2813     \stex_debug:nn{sms}{End~Environment~#2}
2814     \end{#2}\__stex_smsmode_do:w
2815   }{
2816     \__stex_smsmode_do:w
2817   }
2818 }

```

(End of definition for `\stex_smsmode_do:.` This function is documented on page 144.)

sfunction

31.2 Inheritance and Morphisms

```

2819 <@@=stex_import>
\stex_require_module:N
\stex_require_module_noerr:N
\stex_require_module_noerr:n
2820 \cs_new_protected:Nn \stex_require_module:N {
2821   \stex_if_module_exists:NF #1 {
2822     \__stex_import_load_module:NF #1 {
2823       \msg_error:nnx{stex}{error/unknownmodule}{\stex_use_module_uri:N #1}
2824     }
2825   }
2826   \stex_activate_module:N #1

```

```

2827 }
2828
2829 \cs_new_protected:Nn \stex_require_module_noerr:N {
2830   \stex_if_module_exists:NF #1 {
2831     \__stex_import_load_module:NF #1 {}
2832     \str_if_empty:NF \l__stex_import_file_str {
2833       \stex_activate_module:N #1
2834     }
2835   }
2836 }
2837
2838 \cs_new_protected:Nn \stex_require_module_noerr:n {
2839   \tl_set:Nn \l__stex_import_uri { #1 }
2840   \stex_require_module_noerr:N \l__stex_import_uri
2841 }
2842
2843 \cs_new_protected:Npn \__stex_import_load_module:NF #1 #2 {
2844   \tl_set_eq:NN \l__stex_import_uri #1
2845   \tl_set:Ne \l__stex_import_archive_str { \stex_module_uri_archive:N #1 }
2846   \tl_set:Ne \l__stex_import_path_str { \stex_module_uri_path:N #1 }
2847   \tl_set:Ne \l__stex_import_name_str { \stex_module_uri_name:N #1 }
2848   \str_if_eq:NNTF \l__stex_import_archive_str \c_stex_no_archive_str {
2849     \str_set_eq:NN \l__stex_import_pre_str \l__stex_import_path_str
2850     \__stex_import_load_check:n {#2}
2851   }{
2852     \exp_args:No \stex_in_archive:nn \l__stex_import_archive_str {
2853       \str_set:Ne \l__stex_import_pre_str {
2854         \exp_args:Ne \stex_source_path:n{ \l__stex_import_path_str }
2855       }
2856       \exp_args:No \__stex_import_load_check:n {#2}
2857     }
2858   }
2859 }
2860
2861 \cs_new_protected:Nn \__stex_import_load_check:n {
2862   \str_clear:N \l__stex_import_file_str
2863   \str_set_eq:NN \l__stex_import_language_str \l_stex_current_language_str
2864   \__stex_import_check_file:nnn{ / \l__stex_import_name_str .tex }{}{
2865     \__stex_import_check_file:nnn{/ \l__stex_import_name_str . \l_stex_current_language_str
2866       \__stex_import_check_file:nnn{/ \l__stex_import_name_str .en.tex}{
2867         \str_set:Nn \l__stex_import_language_str {en}
2868       }{
2869         \__stex_import_load_check_i:n{#1}
2870       }
2871     }
2872   }
2873   \__stex_import_load_check_ii:
2874 }
2875
2876 \cs_new_protected:Nn \__stex_import_load_check_i:n {
2877   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq / \l__stex_import_path_str
2878   \seq_pop_right:NN \l_tmpa_seq \l__stex_import_name_str
2879   \str_set:Ne \l__stex_import_path_str { \seq_use:Nn \l_tmpa_seq /}
2880   \__stex_import_check_file:nnn{.tex}{}{

```

```

2881 \__stex_import_check_file:nnn{. \l_stex_current_language_str .tex}{ }{
2882 \__stex_import_check_file:nnn{.en.tex}{
2883 \str_set:Nn \l__stex_import_language_str {en}
2884 }{
2885 #1
2886 }
2887 }
2888 }
2889 }
2890
2891 \cs_new_protected:Nn \__stex_import_load_check_ii: {
2892 \str_if_empty:NF \l__stex_import_file_str {
2893 \stex_if_smsmode:TF{
2894 \exp_args:NNo \exp_args:Nne \str_if_eq:nnTF{\l__stex_import_file_str}{\stex_file_use:
2895 \stex_debug:nn{imports}{Skipping~current~file}
2896 }{
2897 \IfFileExists{ \l__stex_import_file_str }{
2898 \__stex_import_load_file:
2899 }{ }
2900 }
2901 }{
2902 \IfFileExists{ \l__stex_import_file_str }{
2903 \__stex_import_load_file:
2904 }{ }
2905 }
2906 }
2907 }
2908
2909 \cs_new_protected:Npn \__stex_import_check_file:nnn #1 #2 {
2910 \stex_debug:nn{imports}{Checking~ \l__stex_import_pre_str #1}
2911 \IfFileExists{ \l__stex_import_pre_str #1 }{
2912 \stex_debug:nn{imports}{Success}
2913 \str_set:Ne \l__stex_import_file_str { \l__stex_import_pre_str #1 }
2914 #2
2915 }
2916 }
2917
2918 \cs_new_protected:Nn \__stex_import_load_file: {
2919 \tl_set:Ne \l__stex_import_doc_uri {
2920 { \l__stex_import_archive_str }
2921 { \l__stex_import_path_str }
2922 { \l__stex_import_name_str }
2923 { \l__stex_import_language_str }
2924 {}
2925 }
2926 \exp_args:NNo \stex_file_in_smsmode:Nn \l__stex_import_doc_uri \l__stex_import_file_str
2927 \stex_if_module_exists:NF \l__stex_import_uri {
2928 \msg_error:nnx{stex}{error/unknownmodule}{\stex_use_module_uri:N \l__stex_import_uri}
2929 }
2930 }

```

(End of definition for `\stex_require_module:N`, `\stex_require_module_noerr:N`, and `\stex_require_module_noerr:n`. These functions are documented on page 145.)

sfunction

\usemodule

```

2931 \stex_new_stylable_cmd:nnnn {usemodule} { 0{ } m } {
2932   \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
2933   \stex_require_module:N \l_stex_import_uri
2934   \__stex_import_usemodule:
2935 }{ }%{\aftergroup\ignorespaces}
2936
2937 \cs_new_protected:Nn \__stex_import_usemodule: {
2938   \stex_debug:nn{usemodule}{Done.}
2939   \stex_if_do_html:T {
2940     \hbox{\stex_annotate_invisible:nn
2941       {data-ftml-usemodule=\stex_use_module_uri:N \l_stex_import_uri } {}}
2942   }
2943   \stex_if_smsmode:F{
2944     \group_begin:
2945     \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_import_uri }
2946     \str_set:Ne \thismodulename {\stex_module_uri_name:N \l_stex_import_uri}
2947     \tl_clear:N \thisstyle
2948     \stex_style_apply:
2949     \group_end:
2950   }
2951 }
2952 \stex_deactivate_macro:Nn \usemodule {document~environment}
2953 \AtBeginDocument{\stex_reactivate_macro:N \usemodule}

```

(End of definition for \usemodule. This function is documented on page 145.)

sfunction

\importmodule

```

2954 \stex_new_stylable_cmd:nnnn{importmodule} { 0{ } m } {
2955   \__stex_import_import_module:nn {#1}{#2}
2956   \stex_smsmode_do:
2957 }{ }\aftergroup\ignorespaces}
2958
2959 \stex_deactivate_macro:Nn \importmodule {module~environments}
2960 \stex_every_module:n {\stex_reactivate_macro:N \importmodule}
2961 \stex_sms_allow_escape:N \importmodule
2962
2963 \cs_new_protected:Nn \__stex_import_import_module:nn {
2964   \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
2965   \stex_require_module:N \l_stex_import_uri
2966   \__stex_import_import_module_i:n {#1}
2967 }
2968
2969 \cs_new_protected:Nn \__stex_import_import_module_i:n {
2970   \stex_execute_in_module:e{
2971     \stex_activate_module:n { \l_stex_import_uri }
2972   }
2973   \stex_add_morphism:nonn
2974     {}{\l_stex_import_uri}{import}{ }{ }
2975   \stex_if_do_html:T {
2976     \stex_annotate_invisible:nn
2977       {data-ftml-import=\stex_use_module_uri:N \l_stex_import_uri } { }
2978   }

```

```

2979 \stex_if_smsmode:F{
2980   \group_begin:
2981   \tl_set:Nn \thisarchive {#1}
2982   \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_import_uri }
2983   \str_set:Ne \thismodulename {\stex_module_uri_name:N \l_stex_import_uri}
2984   \tl_clear:N \thisstyle
2985   \stex_style_apply:
2986   \group_end:
2987 }
2988 }

```

In sms mode, we change the definition:

```

2989 \cs_new_protected:Nn \__stex_import_import_module_presms:nn {
2990   \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
2991   \bool_if:NF \l_stex_relative_import_bool {
2992     \tl_gput_right:Ne \g_stex_sms_import_code {
2993       \stex_require_module_noerr:n { \l_stex_import_uri }
2994     }
2995   }
2996 }
2997
2998 \stex_sms_allow_import:Nn \importmodule {
2999   \stex_reactivate_macro:N \importmodule
3000   \let \__stex_import_import_module:nn \__stex_import_import_module_presms:nn
3001 }

```

(End of definition for `\importmodule`. This function is documented on page 145.)

sfunction

`\requiremodule`

```

3002 \stex_new_stylable_cmd:nnnn {requiremodule} { 0{ } m } {
3003   \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
3004   \stex_require_module:N \l_stex_import_uri
3005   \__stex_import_requiremodule:
3006 }{ }{\aftergroup\ignorespaces}
3007 \stex_deactivate_macro:Nn \requiremodule {module~environments}
3008 \stex_every_module:n {\stex_reactivate_macro:N \requiremodule}
3009
3010 \cs_new_protected:Nn \__stex_import_requiremodule: {
3011   \stex_debug:nn{requiremodule}{Done.}
3012   \stex_if_do_html:T {
3013     \hbox{\stex_annotate_invisible:nn
3014       {data-ftml-import=\stex_use_module_uri:N \l_stex_import_uri } {}}
3015   }
3016   \stex_if_smsmode:F{
3017     \group_begin:
3018     \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_import_uri }
3019     \str_set:Ne \thismodulename {\stex_module_uri_name:N \l_stex_import_uri}
3020     \tl_clear:N \thisstyle
3021     \stex_style_apply:
3022     \group_end:
3023   }
3024 }

```

(End of definition for `\requiremodule`. This function is documented on page 145.)

sfunction

`\stex_add_morphism:nnnn`

```

3025 \cs_new_protected:Nn \stex_add_morphism:nnnn {
3026   \exp_args:Nne \prop_gput:cnn{ \stex_morphisms_macro:N \l_stex_current_module_uri }{
3027     \tl_if_empty:nTF{#1}{[\stex_use_module_uri:n {#2}]}{#1}
3028   }{{#1}{#2}{#3}{#4}}
3029 }
3030 \cs_generate_variant:Nn \stex_add_morphism:nnnn {nonn,oooe}

```

(End of definition for `\stex_add_morphism:nnnn`. This function is documented on page 145.)

sfunction

31.3 Theory Morphisms

3031 `<@@=stex_morphisms>`

`\stex_structural_feature_morphism:nnnnn`

`\stex_structural_feature_morphism_with_macros:nnnnn`

`\stex_structural_feature_morphism_end:`

```

3032 \tl_new:N \l_stex_current_domain_uri
3033 \bool_new:N \l__stex_morphisms_with_macros_bool
3034
3035 \cs_new_protected:Npn \stex_structural_feature_morphism_with_macros:nnnnn {
3036   \bool_set_true:N \l__stex_morphisms_with_macros_bool
3037   \__stex_morphisms_morphism:nnnnn
3038 }
3039 \cs_new_protected:Npn \stex_structural_feature_morphism:nnnnn {
3040   \bool_set_false:N \l__stex_morphisms_with_macros_bool
3041   \__stex_morphisms_morphism:nnnnn
3042 }
3043
3044 \cs_new_protected:Nn \__stex_morphisms_morphism:nnnnn {
3045   \tl_clear:N \l_stex_current_domain_uri
3046   \stex_debug:nn{morphisms}{new~morphism:{#1}{#2}{#3}{#4}{#5}}
3047   \tl_if_empty:nTF{#3}{
3048     \stex_get_mathstructure:n{#4}
3049     \tl_set_eq:NN \l_stex_current_domain_uri \l_stex_get_structure_module_uri
3050   }
3051   \tl_if_empty:NT \l_stex_current_domain_uri {
3052     \stex_uri_from_pair:Nnn \l_stex_import_uri { #3 }{ #4 }
3053     \group_begin:
3054     \stex_require_module:N \l_stex_import_uri
3055     \exp_args:Nne \use:nn \group_end: {
3056       \tl_set:Nn \exp_not:N\l_stex_current_domain_uri {\l_stex_import_uri}
3057     }
3058   }
3059   \tl_if_empty:nTF{#1}{
3060     \msg_error:nn{stex}{error/morphism-needs-name}
3061   }
3062   \str_set:Nn \l_tmpa_str {#1}
3063
3064   \stex_debug:nn{morphisms}{#2:~\l_tmpa_str;~for~\stex_use_module_uri:N \l_stex_current_dom
3065

```

```

3066 \str_set:Nn \l__stex_morphisms_feature_str {#2}
3067 \str_set_eq:NN \l_stex_feature_name_str \l_tmpa_str
3068
3069 \stex_if_do_html:TF {
3070   \begin{stex_annotate_env} {
3071     data-ftml-feature-#2={\l_tmpa_str},
3072     data-ftml-domain={\stex_use_module_uri:N \l_stex_current_domain_uri}
3073     #5
3074   }
3075   \stex_annotate_invisible:n{}
3076 }\group_begin:
3077 \__stex_morphisms_setup:
3078 \__stex_morphisms_reactivate:
3079 \stex_metagroup_new:
3080 }
3081
3082 \cs_new_protected:Nn \__stex_morphisms_setup: {
3083   \seq_clear:N \l_stex_morphism_symbols_seq
3084   \seq_clear:N \l_stex_morphism_renames_seq
3085   \seq_clear:N \l_stex_morphism_morphisms_seq
3086   \seq_clear:N \l_stex_morphism_assigns_seq
3087   \__stex_morphisms_do_decls:
3088   \exp_args:No \__stex_morphisms_do:n \l_stex_current_domain_uri
3089 }
3090
3091 \cs_new_protected:Nn \__stex_morphisms_do_decls: {
3092   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_domain_uri {
3093     \exp_args:NNe \seq_put_right:Nn \l_stex_morphism_symbols_seq {
3094       { \stex_new_symbol_uri:nn{##1}{##3} }
3095       { {##2}{##4}{##5}{##6}{##7}{##8}##9 }
3096     }
3097   }
3098 }
3099
3100
3101 \cs_new_protected:Nn \__stex_morphisms_do:n {
3102   %\seq_clear:N \l__stex_morphisms_dones_seq
3103   \prop_map_inline:cn {\_stex_morphisms_macro:n{#1}}{
3104     %\seq_if_in:NnF \l__stex_morphisms_dones_seq {##2}{
3105       % \seq_push:Nn \l__stex_morphisms_dones_seq {##2}
3106       \stex_debug:nn{morphisms}{Doing ~ \tl_to_str:n{##2}}
3107       \__stex_morphisms_do_morph:nnnn ##2
3108     % }
3109   }
3110 }
3111
3112 \cs_new_protected:Nn \__stex_morphisms_do_morph:nnnn {
3113   \tl_if_empty:nF{#3}{
3114     \seq_put_right:Nn \l_stex_morphism_morphisms_seq {#{1}{#2}{#3}}
3115   }
3116   \__stex_morphisms_do:n{#2}
3117 }
3118
3119

```



```

3120 \cs_new_protected:Nn \stex_structural_feature_morphism_end: {
3121   \tl_gset_eq:NN \l_stex_current_domain_uri \l_stex_current_domain_uri
3122   \seq_gset_eq:NN \l_stex_morphism_symbols_seq \l_stex_morphism_symbols_seq
3123   \seq_gset_eq:NN \l_stex_morphism_renames_seq \l_stex_morphism_renames_seq
3124   \seq_gset_eq:NN \l_stex_morphism_assigns_seq \l_stex_morphism_assigns_seq
3125   \seq_gset_eq:NN \l_stex_morphism_morphisms_seq \l_stex_morphism_morphisms_seq
3126   \bool_gset_eq:NN \l__stex_morphisms_with_macros_bool \l__stex_morphisms_with_macros_bool
3127   \stex_if_do_html:TF{
3128     \end{stex_annotate_env}
3129   }\group_end:
3130   \__stex_morphisms_do_elaboration:
3131 }
3132
3133 \cs_new_protected:Nn \__stex_morphisms_do_elaboration: {
3134   \stex_debug:nn{morphisms}{
3135     Elaborating:^^J\meaning \l_stex_morphism_symbols_seq
3136     ^^J^^J
3137     Renamings:^^J
3138     \meaning \l_stex_morphism_renames_seq
3139     ^^J^^J
3140     Assignments:^^J
3141     \meaning \l_stex_morphism_assigns_seq
3142   }
3143
3144   \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
3145     \__stex_morphisms_elab:nn ##1
3146   }
3147
3148   \stex_add_morphism:oooo
3149     \l_stex_feature_name_str
3150     \l_stex_current_domain_uri
3151     \l__stex_morphisms_feature_str
3152     {\seq_map_function:NN \l_stex_morphism_renames_seq \__stex_morphisms_rename:n}
3153 }
3154
3155 \cs_new:Nn \__stex_morphisms_rename:n{
3156   \__stex_morphisms_rename:nn #1
3157 }
3158
3159 \cs_new:Nn \__stex_morphisms_rename:nn{
3160   {#1}{\use_ii:nn#2}
3161 }
3162
3163 \cs_new_protected:Nn \__stex_morphisms_elab:nn {
3164   \tl_clear:N \l__stex_morphisms_tmp
3165   \seq_map_inline:Nn \l_stex_morphism_renames_seq {
3166     \__stex_morphisms_elab_check_rename:nnn{#1}##1
3167   }
3168   \tl_if_empty:NTF \l__stex_morphisms_tmp {
3169     \exp_args:Ne \__stex_morphisms_elab_i:nnn {\stex_symbol_uri_name:n {#1}}{#1}
3170   }{
3171     \stex_debug:nn{morphisms}{Generating~1~\l__stex_morphisms_tmp}
3172     \use_i:nn{ \exp_args:Nno \use:nn {\__stex_morphisms_add_symbol:nnnnnnnnN {#1}} \l__stex
3173   }#2

```

```

3174
3175 \exp_args:No\stex_iterate_notations:nn\l_stex_current_domain_uri {
3176   \str_if_eq:nnT{#1}{##1}{
3177     \exp_args:Ne \stex_add_notation:nnnnn {
3178       \exp_args:Ne \stex_new_symbol_uri:n {\exp_after:wN \use_ii:nn \l__stex_morphisms_tm
3179       }{##2}{##3}{##4}{##5}
3180     }
3181   }
3182 }
3183
3184 \cs_new_protected:Nn \__stex_morphisms_elab_i:nnn {
3185   \tl_set:Ne \l__stex_morphisms_tmp {{{\l_stex_feature_name_str / #1}}}
3186   \stex_debug:nn{morphisms}{Generating-2~\l_stex_feature_name_str / #1}
3187   \bool_if:NTF \l__stex_morphisms_with_macros_bool {
3188     \exp_args:Nnno \__stex_morphisms_add_symbol:nnnnnnnnN {#2} {#3}
3189   }{
3190     \exp_args:Nnno \__stex_morphisms_add_symbol:nnnnnnnnN {#2} {}
3191   }
3192   {\l_stex_feature_name_str / #1}
3193 }
3194
3195 \cs_new_protected:Npn \__stex_morphisms_add_symbol:nnnnnnnnN #1 {
3196   \tl_clear:N \l__stex_morphisms_tmp_b
3197   \seq_map_inline:Nn \l_stex_morphism_assigns_seq {
3198     \__stex_morphisms_elab_check_assign:nnnnn{#1}##1
3199   }
3200   \tl_if_empty:NTF \l__stex_morphisms_tmp_b
3201   \stex_add_symbol:nnnnnnnnN
3202   \__stex_morphisms_add_symbol_i:nnnnnnnnN
3203 }
3204
3205 \cs_new_protected:Npn \__stex_morphisms_add_symbol_i:nnnnnnnnN #1 #2 #3 #4 #5 {
3206   \stex_add_symbol:nnnnnnnnN {#1}{#2}{#3}{#4}{assigned}
3207 }
3208
3209 \cs_new_protected:Nn \__stex_morphisms_elab_check_assign:nnnnn {
3210   \tl_if_eq:nnT{#1}{{#2}{#3}{#4}{#5}}{
3211     \seq_map_break:n{
3212       \tl_set:Nn \l__stex_morphisms_tmp_b {assigned}
3213     }
3214   }
3215 }
3216
3217 \cs_new_protected:Nn \__stex_morphisms_elab_check_rename:nnn {
3218   \tl_if_eq:nnT{#1}{#2}{
3219     \seq_map_break:n{
3220       \tl_set:Nn \l__stex_morphisms_tmp {#3}
3221     }
3222   }
3223 }
3224
3225 \cs_new_protected:Nn \__stex_morphisms_do_for_list: {
3226   \seq_clear:N \l_stex_fors_seq
3227   \clist_map_inline:Nn \l_stex_key_for_clist {

```

```

3228 \exp_args:N\stex_get_in_morphism:n{\tl_to_str:n{##1}}
3229 \seq_put_right:No \l_stex_fors_seq
3230 {\l_stex_get_symbol_uri}
3231 }
3232 }
3233
3234
3235 \cs_new_protected:Nn \__stex_morphisms_definiens_impl:nn {
3236 \tl_if_empty:NF {#1} {
3237 \stex_get_in_morphism:n { #1 }
3238 \tl_set_eq:NN \l_stex_current_def_uri \l_stex_get_symbol_uri
3239 }
3240 \tl_if_empty:NT \l_stex_current_def_uri {
3241 \msg_error:nn{stex}{error/definiensfor}
3242 }
3243 \stex_debug:nn{definiens}{Checking-\stex_use_symbol_uri:N \l_stex_current_def_uri }
3244 \stex_if_do_html:TF{
3245 \exp_after:wN \stex_metagroup_do_in:n \exp_after:wN {
3246 \exp_after:wN \seq_put_right:Nn \exp_after:wN \l_stex_morphism_assigns_seq
3247 \exp_after:wN { \l_stex_current_def_uri }
3248 }
3249 \mode_leave_vertical: \stex_annotate:nn{data-ftml-assign={\stex_use_symbol_uri:N \l_st
3250 \stex_annotate_force_break:n{
3251 \stex_annotate:nn{data-ftml-definiens={}}{#2}
3252 }
3253 }
3254 }{
3255 \exp_args:No \__stex_morphisms_add_definiens:nn \l_stex_current_def_uri {#2}
3256 \stex_if_smsmode:F{#2}
3257 }
3258 }
3259
3260 \cs_new_protected:Nn \__stex_morphisms_add_definiens:nn {
3261 \tl_set:Nn \l_stex_get_symbol_uri {#1}
3262 \stex_debug:nn{morphisms}{Adding-definiens~\tl_to_str:n{#1~#2}}
3263 %\__stex_morphisms_set_definiens_macros: #1\__stex_morphisms_break:
3264 \stex_assign_do:n{#2}
3265 }
3266
3267 %\cs_new_protected:Npn \__stex_morphisms_set_definiens_macros: #1?#2?#3\__stex_morphisms_br
3268 % \str_set:Nn \l_stex_get_symbol_uri { #1?#2} % TODO
3269 % \str_set:Nn \l_stex_get_symbol_name_str {#3}
3270 % \exp_args:Nne\use:nn{\__stex_morphisms_set_definiens_macros_i:nnnnnnn}{
3271 % \prop_item:Nn \l_stex_morphism_symbols_prop {[#1?#2]/[#3]}
3272 % }
3273 %}
3274
3275 %\cs_new_protected:Nn \__stex_morphisms_set_definiens_macros_i:nnnnnnn {
3276 % \tl_set:Nn \l_stex_get_symbol_def_tl{#4}
3277 %}
3278
3279
3280 \cs_new_protected:Nn \__stex_morphisms_rename_all: {
3281 % TODO

```

```

3282 }
3283
3284
3285 \cs_new_protected:Nn \stex_structural_feature_morphism_check_total: {
3286   \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
3287     \__stex_morphisms_total_check:nn ##1
3288   }
3289 }
3290
3291 \cs_new_protected:Nn \__stex_morphisms_total_check:nn {
3292   \__stex_morphisms_total_check:nnnnnnnN {#1} #2
3293 }
3294
3295 \cs_new_protected:Nn \__stex_morphisms_total_check:nnnnnnnN {
3296   \tl_if_empty:nT{#5}{
3297     \seq_if_in:NnF \l_stex_morphism_assigns_seq {#1} {
3298       \msg_error:nnxx{stex}{error/needsdefiniens}{\stex_use_symbol_uri:n {#1}}{total-morphi
3299     }
3300   }
3301 }
3302
3303 \cs_new:Npn \__stex_morphisms_split_qm:w #1 ? #2 ? #3 { #3 }
3304
3305
3306
3307 \cs_new_protected:Npn \__stex_morphisms_reactivate: {
3308   \stex_deactivate_macro:Nn \symdecl {module~environments}
3309   \stex_deactivate_macro:Nn \textsymdecl {module~environments}
3310   \stex_deactivate_macro:Nn \symdef {module~environments}
3311   \stex_deactivate_macro:Nn \notation {module~environments}
3312   \stex_deactivate_macro:Nn \importmodule {module~environments}
3313   \stex_deactivate_macro:Nn \requiremodule {module~environments}
3314   \stex_deactivate_macro:Nn \smodule {outside-of~morphisms}
3315   \stex_reactivate_macro:N \assign
3316   \stex_reactivate_macro:N \assignMorphism
3317   \stex_reactivate_macro:N \renamedec1
3318   \cs_set_eq:NN \stex_do_for_list: \__stex_morphisms_do_for_list:
3319   \cs_set_eq:NN \stex_add_definiens:nn \__stex_morphisms_add_definiens:nn
3320   \cs_set_eq:NN \stex_definiens_impl:nn \__stex_morphisms_definiens_impl:nn
3321 }

```

(End of definition for \stex_structural_feature_morphism:nnnnn, \stex_structural_feature_morphism_ with_macros:nnnnn, and \stex_structural_feature_morphism_end:. These functions are documented on page 145.)

sfunction

\stex_iterate_morphisms:nn

```

3322 \cs_new_protected:Nn \stex_iterate_morphisms:nn {
3323   \seq_clear:N \l__stex_morphisms_mods_seq
3324   \bool_set_true:N \l__stex_morphisms_continue_bool
3325   \cs_set:Npn \__stex_morphisms_cs:nnnn ##1 ##2 ##3 ##4 ##5 {
3326     #2
3327     \bool_if:NT \l__stex_morphisms_continue_bool {
3328       \str_if_eq:nnTF{##1}{[#2]}{
3329         \tl_put_right:Nn \l__stex_morphisms_todo_tl {

```

```

3330     \__stex_morphisms_iterate:nn{##5}{##2}
3331   }
3332   ){
3333     \tl_put_right:Nn \l__stex_morphisms_todo_tl {
3334       \__stex_morphisms_iterate:nn{##5 / ##1}{##2}
3335     }
3336   }
3337 }
3338 }
3339 \cs_set:Npn \stex_iterate_break:n ##1 {
3340   \bool_set_false:N \l__stex_morphisms_continue_bool
3341   \prop_map_break:n{##1}
3342 }
3343 \__stex_morphisms_iterate:nn{}{##1}
3344 }
3345 \cs_new_protected:Nn \__stex_morphisms_iterate:nn {
3346   \tl_clear:N \l__stex_morphisms_todo_tl
3347   \seq_if_in:NnF \l__stex_morphisms_mods_seq {#1 #2}{
3348     \seq_put_right:Nn \l__stex_morphisms_mods_seq {#1 #2}
3349     \prop_map_inline:cn{\_stex_morphisms_macro:n{#2}}{
3350       \__stex_morphisms_cs:nnnn ##2 {#1}
3351       % TODO
3352       % ##1: name or [mpath]
3353       % ##2 = {####1}{####2}{####3}{####4}
3354       % ####1 = name
3355       % ####2 = mpath
3356       % ####3 = type
3357       % ####4 = {origname}{newname}*
3358     }
3359     \bool_if:NT \l__stex_morphisms_continue_bool \l__stex_morphisms_todo_tl
3360   }
3361 }

```

(End of definition for `\stex_iterate_morphisms:nn`. This function is documented on page 145.)

sfunction

`\stex_get_in_morphism:n`

```

3362 \cs_new_protected:Nn \stex_get_in_morphism:n {
3363   \stex_debug:nn{morphisms}{Getting-in-morphism:~#1}
3364   \tl_clear:N \l_stex_get_symbol_uri
3365   \str_set:Nn \l__stex_morphisms_get_str {#1}
3366   \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
3367     \__stex_morphisms_get_check:nn ##1
3368   }
3369   \tl_if_empty:NT \l_stex_get_symbol_uri {
3370     \seq_map_inline:Nn \l_stex_morphism_renames_seq {
3371       \__stex_morphisms_renamed_check:nn##1
3372     }
3373     \tl_if_empty:NT \l_stex_get_symbol_uri {
3374       \msg_error:nnxx{stex}{error/unknownsymbolin}{#1}{
3375         morphism~\l_stex_feature_name_str
3376       }
3377     }
3378   }

```

```

3379 }
3380
3381
3382 \cs_new_protected:Nn \__stex_morphisms_get_check:nn {
3383   \__stex_morphisms_check_name:nnnn #1 #2
3384 }
3385
3386 \cs_new_protected:Nn \__stex_morphisms_check_name:nnnn {
3387   % TODO module name, path, archive, macroname ?
3388   \exp_args:No \str_if_eq:nnTF \l__stex_morphisms_get_str {#4} {
3389     \tl_set:Nn \l_stex_get_symbol_uri {{{#1}{#2}{#3}{#4}}}
3390     \__stex_morphisms_set:nnnnnnN
3391   }{
3392     \__stex_morphisms_macro:nnnnnnN{{{#1}{#2}{#3}{#4}}}
3393   }
3394 }
3395
3396 \cs_new_protected:Nn \__stex_morphisms_set:nnnnnnN {
3397   \seq_map_break:n{
3398     \str_set:Nn \l_stex_get_symbol_macro_str{#1}
3399     \int_set:Nn \l_stex_get_symbol_arity_int {#2}
3400     \tl_set:Nn \l_stex_get_symbol_args_tl {#3}
3401     \tl_set:Nn \l_stex_get_symbol_def_tl {#4}
3402     \tl_set:Nn \l_stex_get_symbol_type_tl {#5}
3403     \tl_set:Nn \l_stex_get_symbol_return_tl {#6}
3404     \tl_set:Nn \l_stex_get_symbol_invoke_cs {#7}
3405   }
3406 }
3407
3408 \cs_new_protected:Npn \__stex_morphisms_macro:nnnnnnN #1 #2 {
3409   \exp_args:No \str_if_eq:nnTF \l__stex_morphisms_get_str {#2}{
3410     \tl_set:Nn \l_stex_get_symbol_uri{#1}
3411     \__stex_morphisms_set:nnnnnnN
3412   }\__stex_morphisms_gobble:nnnnnnN
3413   {#2}
3414 }
3415
3416 \cs_new_protected:Nn \__stex_morphisms_gobble:nnnnnnN {}
3417
3418 \cs_new_protected:Nn \__stex_morphisms_renamed_check:nn {
3419   \stex_debug:nn{here}{\tl_to_str:n{{{#1}{#2}}}:~\l__stex_morphisms_get_str}
3420   \__stex_morphisms_renamed_check:nnnnnn #1 #2
3421 }
3422
3423 \cs_new_protected:Nn \__stex_morphisms_renamed_check:nnnnnn {
3424   \exp_args:No \str_if_eq:nnTF \l__stex_morphisms_get_str{#5}{
3425     \seq_map_break:n{
3426       \str_set:Nn \l__stex_morphisms_get_str {#4}
3427       \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
3428         \__stex_morphisms_get_check:nn ##1
3429       }
3430     }
3431   }{
3432     \exp_args:No \str_if_eq:nnT \l__stex_morphisms_get_str{#6}{

```

```

3433     \seq_map_break:n{
3434         \str_set:Nn \l__stex_morphisms_get_str {#4}
3435         \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
3436             \__stex_morphisms_get_check:nn ##1
3437         }
3438     }
3439 }
3440 }
3441 }

```

(End of definition for `\stex_get_in_morphism:n`. This function is documented on page 145.)

sfunction

`copymodule (env.)`

```

3442 \stex_new_stylable_env:nnnnnnn {copymodule}{m 0{} m}{
3443     \__stex_morphisms_begin_copy:Nnn\stex_structural_feature_morphism:nnnnn {#1}{#2}{#3}
3444 }{
3445     \stex_if_smsmode:F {
3446         \stex_style_apply:
3447     }
3448     \stex_structural_feature_morphism_end:
3449 }{}{}{}
3450
3451 \stex_new_stylable_env:nnnnnnn {copymodule*}{m 0{} m}{
3452     \cs_set:Npn \stex_style_apply: {
3453         \stex_apply_patch_begin:n{copymodule}
3454     }
3455     \__stex_morphisms_begin_copy:Nnn\stex_structural_feature_morphism_with_macros:nnnnn {#1}
3456 }{
3457     \cs_set:Npn \stex_style_apply: {
3458         \stex_apply_patch_end:n{copymodule}
3459     }
3460     \stex_if_smsmode:F {
3461         \stex_style_apply:
3462     }
3463     \stex_structural_feature_morphism_end:
3464 }{}{}{}
3465
3466 \cs_new_protected:Nn \__stex_morphisms_begin_copy:Nnnn {
3467     #1{#2}{morphism}{#3}{#4}{,data-ftml-total=false}
3468     \stex_if_smsmode:F {
3469         \tl_set:Nn \thiscopypname { #2 }
3470         \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
3471         \stex_style_apply:
3472     }
3473     \stex_smsmode_do:
3474 }
3475
3476
3477 \stex_deactivate_macro:Nn \copymodule {module-environments}
3478 \stex_every_module:n {
3479     \stex_reactivate_macro:N \copymodule
3480 }
3481 \stex_sms_allow_env:n{copymodule}

```

```

3482
3483 \exp_after:wN \stex_deactivate_macro:Nn \csname copymodule*\endcsname {module~environments}
3484 \stex_every_module:n {
3485   \exp_after:wN\stex_reactivate_macro:N \csname copymodule*\endcsname
3486 }
3487 \stex_sms_allow_env:n{copymodule*}
3488
3489 \stex_new_stylable_cmd:nnnn{copymod}{s m 0{} m m}{
3490   \IfBooleanTF#1\stex_structural_feature_morphism_with_macros:nnnnn
3491   \stex_structural_feature_morphism:nnnnn{#2}{morphism}{#3}{#4}{,data-ftml-total={false}}
3492   \clist_map_function:nN{#5}\__stex_morphisms_parse_assign:n
3493   \stex_if_smsmode:F {
3494     \tl_set:Nn \thiscopyname { #2 }
3495     \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
3496     \stex_style_apply:
3497   }
3498   \stex_structural_feature_morphism_end:
3499   \stex_smsmode_do:
3500 }{}
3501
3502 \stex_deactivate_macro:Nn \copymod {module~environments}
3503 \stex_every_module:n {
3504   \stex_reactivate_macro:N \copymod
3505 }
3506 \stex_sms_allow_escape:N\copymod

```

interpretmodule (env.)

```

3507 \stex_new_stylable_env:nnnnnnn {interpretmodule}{m 0{} m}{
3508   \__stex_morphisms_begin_interpret:Nnn\stex_structural_feature_morphism:nnnnn {#1}{#2}{#3}
3509 }{
3510   \stex_structural_feature_morphism_check_total:
3511   \stex_if_smsmode:F {
3512     \stex_style_apply:
3513   }
3514   \stex_structural_feature_morphism_end:
3515 }{}{}{}
3516
3517 \stex_new_stylable_env:nnnnnnn {interpretmodule*}{m 0{} m}{
3518   \cs_set:Npn \stex_style_apply: {
3519     \stex_apply_patch_begin:n{interpretmodule}
3520   }
3521   \__stex_morphisms_begin_interpret:Nnn\stex_structural_feature_morphism_with_macros:nnnnn
3522 }{
3523   \cs_set:Npn \stex_style_apply: {
3524     \stex_apply_patch_end:n{interpretmodule}
3525   }
3526   \stex_structural_feature_morphism_check_total:
3527   \stex_if_smsmode:F {
3528     \stex_style_apply:
3529   }
3530   \stex_structural_feature_morphism_end:
3531 }{}{}{}
3532
3533 \cs_new_protected:Nn \__stex_morphisms_begin_interpret:Nnnn {

```



```

3534 #1{#2}{morphism}{#3}{#4}{,data-ftml-total=true}
3535 \stex_if_smsmode:F {
3536   \tl_set:Nn \thiscopynome { #2 }
3537   \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
3538   \stex_style_apply:
3539 }
3540 \stex_smsmode_do:
3541 }
3542
3543 \stex_deactivate_macro:Nn \interpretmodule {module~environments}
3544 \stex_every_module:n {
3545   \stex_reactivate_macro:N \interpretmodule
3546 }
3547 \stex_sms_allow_env:n{interpretmodule}
3548
3549 \exp_after:wN \stex_deactivate_macro:Nn \csname interpretmodule*\endcsname {module~environm
3550 \stex_every_module:n {
3551   \exp_after:wN \stex_reactivate_macro:N \csname interpretmodule*\endcsname
3552 }
3553 \stex_sms_allow_env:n{interpretmodule*}
3554
3555 \stex_new_stylable_cmd:nnnn{interpretmod}{s m 0{} m m}{
3556   \IfBooleanTF#1\stex_structural_feature_morphism_with_macros:nnnnn
3557   \stex_structural_feature_morphism:nnnnn{#2}{morphism}{#3}{#4}{,data-ftml-total=true}
3558   \clist_map_function:nN{#5}\__stex_morphisms_parse_assign:n
3559   \stex_if_smsmode:F {
3560     \tl_set:Nn \thiscopynome { #2 }
3561     \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
3562     \stex_style_apply:
3563   }
3564   \stex_structural_feature_morphism_check_total:
3565   \stex_structural_feature_morphism_end:
3566   \stex_smsmode_do:
3567 }{}
3568
3569 \stex_deactivate_macro:Nn \interpretmod {module~environments}
3570 \stex_every_module:n {
3571   \stex_reactivate_macro:N \interpretmod
3572 }
3573 \stex_sms_allow_escape:N\interpretmod

```

\assign

```

3574 \stex_new_stylable_cmd:nnnn {assign} { m m }{
3575   \stex_get_in_morphism:n{#1}
3576   \stex_if_do_html:TF{
3577     \stex_annotate_invisible:n{\hbox{$\_stex_assign_do:n{#2}$}}
3578   }{
3579     \_stex_assign_do:n{#2}
3580   }
3581   \stex_smsmode_do:
3582 }{}
3583 \stex_sms_allow_escape:N\assign
3584 \stex_deactivate_macro:Nn \assign {morphism~environments}
3585

```

```

3586 \cs_new_protected:Nn \_stex_assign_do:n{
3587   \stex_debug:nn{assign}{Assigning~\stex_use_symbol_uri:N \l_stex_get_symbol_uri~to~\tl_to_
3588   \stex_check_term:nn{assignment}{#1}
3589   \stex_if_do_html:T{
3590     \stex_annotate_invisible:nn{data-ftml-assign={\stex_use_symbol_uri:N \l_stex_get_symbol
3591     \_stex_annotate_force_break:n{
3592       \mode_if_math:TF{\stex_annotate:nn{data-ftml-definiens={}}{#1}}{\hbox{$\stex_annota
3593     }
3594   }
3595 }
3596 \exp_after:wN \stex_metagroup_do_in:n \exp_after:wN {
3597   \exp_after:wN \seq_put_right:Nn \exp_after:wN \l_stex_morphism_assigns_seq
3598   \exp_after:wN { \l_stex_get_symbol_uri }
3599 }
3600 }
3601
3602 \cs_new_protected:Nn \__stex_morphisms_parse_assign:n {
3603   \str_clear:N \l__stex_morphisms_name_str
3604   \str_clear:N \l__stex_morphisms_newname_str
3605   \tl_clear:N \l__stex_morphisms_ass_tl
3606   \stex_debug:nn{morphisms}{Parsing~#1}
3607   \exp_args:NNe \seq_set_split:Nnn \l__stex_morphisms_seq {\tl_to_str:n{0}} {#1}
3608   \int_compare:nNnTF {\seq_count:N \l__stex_morphisms_seq} = 1 {
3609     \stex_debug:nn{morphisms}{No~0}
3610     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_next_tl
3611   }{
3612     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3613     \stex_debug:nn{morphisms}{Name:~\l__stex_morphisms_name_str}
3614     \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3615     \tl_set:Ne \l__stex_morphisms_next_tl {\seq_use:Nn \l__stex_morphisms_seq 0}
3616   }
3617   \exp_args:NNNo \seq_set_split:Nnn \l__stex_morphisms_seq = \l__stex_morphisms_next_tl
3618   \str_if_empty:NTF \l__stex_morphisms_name_str {
3619     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3620     \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3621     \tl_set:Ne \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3622   }{
3623     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_newname_str
3624     \exp_args:NNo \str_set:Nn \l__stex_morphisms_newname_str \l__stex_morphisms_newname_str
3625     \tl_set:Ne \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3626   }
3627   \__stex_morphisms_do_parsed_assign:
3628 }
3629
3630 \cs_new_protected:Nn \__stex_morphisms_do_parsed_assign: {
3631   \exp_args:No \stex_get_in_morphism:n \l__stex_morphisms_name_str
3632   \str_if_empty:NF \l__stex_morphisms_newname_str {
3633     \stex_debug:nn{morphisms}{renaming~\l__stex_morphisms_name_str;~to~\l__stex_morphisms_n
3634     \exp_after:wN \__stex_morphisms_do_parsed_newname: \l__stex_morphisms_newname_str \__st
3635   }
3636   \tl_if_empty:NF \l__stex_morphisms_ass_tl {
3637     \stex_debug:nn{morphisms}{assigning~\l__stex_morphisms_name_str;~to~\exp_args:No \tl_to
3638     \exp_args:No \_stex_assign_do:n \l__stex_morphisms_ass_tl
3639   }

```

```

3640 }
3641
3642 \cs_new_protected:Nn \__stex_morphisms_do_parsed_newname: {
3643   \peek_charcode:NTF [ {
3644     \__stex_morphisms_do_parsed_newname:w
3645   }{
3646     \__stex_morphisms_do_parsed_newname:w []
3647   }
3648 }
3649
3650 \cs_new_protected:Npn \__stex_morphisms_do_parsed_newname:w [#1] #2 \__stex_morphisms_end:
3651   \stex_renamedec1_do:nn{#1}{#2}
3652 }

```

(End of definition for \assign. This function is documented on page 146.)

sfunction

\renamedec1

```

3653 \stex_new_stylable_cmd:nnnn {renamedec1} { m 0{} m }{
3654   \stex_get_in_morphism:n{#1}
3655   \stex_renamedec1_do:nn{#2}{#3}
3656   \stex_smsmode_do:
3657 }{}
3658 \stex_sms_allow_escape:N\renamedec1
3659 \stex_deactivate_macro:Nn \renamedec1 {morphism~environments}
3660
3661 \cs_new_protected:Nn \stex_renamedec1_do:nn {
3662   \stex_debug:nn{renamedec1}{Renaming~\stex_use_symbol_uri:N \l_stex_get_symbol_uri~to~[#1]
3663   \stex_if_do_html:T{
3664     \exp_args:Ne \stex_annotate_invisible:nn{
3665       data-ftml-rename={\stex_use_symbol_uri:N \l_stex_get_symbol_uri},
3666       data-ftml-macroname={#2}
3667       \str_if_empty:nF{#1}{ ,data-ftml-to={#1} }
3668     }}
3669   }
3670   \stex_metagroup_do_in:e {
3671     \seq_push:Nn \exp_not:N \l_stex_morphism_renames_seq
3672     {{\l_stex_get_symbol_uri}{#2}{
3673       \tl_if_empty:nTF{#1}{
3674         \l_stex_feature_name_str/\stex_symbol_uri_name:N \l_stex_get_symbol_uri
3675       }{#1}
3676     }}}
3677   }
3678 }

```

(End of definition for \renamedec1. This function is documented on page 146.)

sfunction

\assignMorphism

```

3679 \stex_new_stylable_cmd:nnnn {assignMorphism} { m m }{
3680   \str_clear:N \l__stex_morphisms_morphism_dom_str
3681   \exp_args:No \stex_iterate_morphisms:nn\l_stex_current_domain_uri{
3682     \stex_debug:nn{assignMorphism}{
3683       Checking:~#1~vs:^^J##1^^J##2^^J##3^^J##4

```

```

3684     }
3685     \str_if_eq:nnTF{#1}{##1}{
3686       \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3687     }{
3688       \stex_str_if_ends_with:nnT{##2}{#1}{
3689         \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3690       }
3691     }
3692   }
3693   \str_if_empty:NT \l__stex_morphisms_morphism_dom_str {
3694     \msg_error:nnn{stex}{error/nomorphism}{#1}
3695   }
3696   \bool_set_false:N \l_tmpa_bool
3697   \stex_iterate_morphisms:nn \l_stex_current_module_str {
3698     \stex_debug:nn{assignMorphism}{
3699       Checking:~#2~vs:^^J##1^^J##2^^J##3^^J##4
3700     }
3701     \str_if_eq:nnTF{#2}{##1}{
3702       \stex_debug:nn{assignMorphism}{match!}
3703       \stex_iterate_break:n{
3704         \stex_annotate_invisible:nn{
3705           data-ftml-assignmorphismfrom={\l__stex_morphisms_morphism_dom_str}
3706           data-ftml-assignmorphismto={\l_stex_current_module_str?##1}
3707         }{}
3708         \bool_set_true:N \l_tmpa_bool
3709       }
3710     }{
3711       \stex_str_if_ends_with:nnT{##2}{#2}{
3712         \stex_debug:nn{assignMorphism}{match!}
3713         \stex_iterate_break:n{
3714           \stex_annotate_invisible:nn{
3715             data-ftml-assignmorphismfrom={\l__stex_morphisms_morphism_dom_str},
3716             data-ftml-assignmorphismto={\l_stex_current_module_str?##1}
3717           }{}
3718           \bool_set_true:N \l_tmpa_bool
3719         }
3720       }
3721     }
3722   }
3723   \bool_if:NF \l_tmpa_bool {
3724     \msg_error:nnn{stex}{error/nomorphism}{#2}
3725   }
3726 }{}
3727 \stex_sms_allow_escape:N \assignMorphism
3728 \stex_deactivate_macro:Nn \assignMorphism {morphism~environments}
3729
3730 \cs_new_protected:Nn \__stex_morphisms_do_morph_assign:nnn {
3731   \stex_iterate_break:n{
3732     \str_set:Nx \l__stex_morphisms_morphism_dom_str { \l_stex_current_domain_str ? #1 }
3733     \stex_debug:nn{assignMorphism}{match!}
3734     \stex_iterate_symbols:nn{#2}{
3735       \stex_debug:nn{assignMorphism}{removing~##1?##3}
3736       % TODO: non-trivial assignments
3737       \prop_remove:Nn \l_stex_morphism_symbols_prop {

```

```

3738         [[#1]] / [[#3]]
3739     }
3740 }
3741 }
3742 }

```

(End of definition for \assignMorphism. This function is documented on page 146.)

sfunction

Chapter 32

Symbols

3743 <@@=stex_syms>

32.1 Declarations

Keys used in various symbol declarations:

```
3744 \stex_keys_define:nnnn{symargs}{
3745   \str_clear:N \l_stex_key_args_str
3746   \str_clear:N \l_stex_key_role_str
3747   \str_clear:N \l_stex_key_reorder_str
3748   \str_clear:N \l_stex_key_assoc_str
3749 }{
3750   args      .str_set:N = \l_stex_key_args_str ,
3751   reorder   .str_set:N = \l_stex_key_reorder_str ,
3752   assoc     .choices:nn = {bin,binl,binr,pre,conj,pwconj}
3753   { \str_set:Ne \l_stex_key_assoc_str \l_keys_choice_tl},
3754   role      .str_set:N = \l_stex_key_role_str
3755 }{}
3756
3757 \stex_keys_define:nnnn{decl}{
3758   \str_clear:N \l_stex_key_name_str
3759   \str_clear:N \l_stex_key_args_str
3760   \tl_clear:N \l_stex_key_type_tl
3761   \tl_clear:N \l_stex_key_def_tl
3762   \tl_clear:N \l_stex_key_return_tl
3763   \str_clear:N \l_stex_key_wikidata_str
3764   \clist_clear:N \l_stex_key_argtypes_clist
3765 }{
3766   name      .str_set:N = \l_stex_key_name_str ,
3767   return    .tl_set:N = \l_stex_key_return_tl ,
3768   argtypes  .clist_set:N = \l_stex_key_argtypes_clist ,
3769   wikidata  .str_set:N = \l_stex_key_wikidata_str ,
3770   type      .tl_set:N = \l_stex_key_type_tl ,
3771   def       .tl_set:N = \l_stex_key_def_tl ,
3772   align     .code:n = {},
3773   gf        .code:n = {}
3774 }{style,deprecate,symargs}
```

\symdecl

```

3775 \str_new:N \l_stex_macroname_str
3776
3777 \stex_new_stylable_cmd:nnnn {symdecl} { s m 0{}} {
3778   \stex_keys_set:nn{decl}{#3}
3779   \IfBooleanTF #1 {
3780     \str_clear:N \l_stex_macroname_str
3781   }{
3782     \str_set:Ne \l_stex_macroname_str { #2 }
3783   }
3784   \__stex_syms_top:n{#2}
3785   \stex_if_smsmode:F \__stex_syms_style:
3786   \stex_smsmode_do:
3787 }{}
3788
3789 \stex_deactivate_macro:Nn \symdecl {module~environments}
3790 \stex_every_module:n {\stex_reactivate_macro:N \symdecl}
3791 \stex_sms_allow_escape:N \symdecl
3792
3793 \cs_new_protected:Nn \__stex_syms_style: {
3794   \group_begin:
3795     \tl_set:Ne \thisdecluri {\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
3796     \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3797     \tl_set_eq:NN \thistype \l_stex_key_type_tl
3798     \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
3799     \tl_set_eq:NN \thisargs \l_stex_key_args_str
3800     \tl_clear:N \thisstyle
3801     \stex_style_apply:
3802   \group_end:
3803 }

```

(End of definition for \symdecl. This function is documented on page 147.)

sfunction

\symdef

```

3804 \stex_new_stylable_cmd:nnnn {symdef} { m 0{ } m } {
3805   \stex_keys_set:nn{symdef}{#2}
3806   \str_set:Ne \l_stex_macroname_str { #1 }
3807   \__stex_syms_top:n{#1}
3808   \stex_debug:nn{symdef}{Doing~\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
3809   \str_set_eq:NN \l_stex_get_symbol_uri \l_stex_current_symbol_uri
3810   \stex_notation_parse:n{#3}
3811   \stex_if_check_terms:T{ \_stex_notation_check: }
3812   \_stex_notation_add:
3813   \stex_if_do_html:T{
3814     \_stex_notation_do_html:n{\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
3815   }
3816   \stex_if_smsmode:F \__stex_syms_def_style:
3817   \stex_smsmode_do:
3818 }{}
3819
3820 \stex_deactivate_macro:Nn \symdef {module~environments}
3821 \stex_every_module:n {\stex_reactivate_macro:N \symdef}
3822 \stex_sms_allow_escape:N \symdef
3823

```

```

3824 \cs_new_protected:Nn \__stex_syms_def_style: {
3825   \group_begin:
3826   \tl_set:Nx \thisdecluri {\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
3827   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3828   \tl_set_eq:NN \thistype \l_stex_key_type_tl
3829   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
3830   \tl_set_eq:NN \thisargs \l_stex_key_args_str
3831   \tl_clear:N \thisstyle
3832   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
3833   \def\thisnotation{
3834     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}}{
3835     \stex_notation_make_args:
3836   }
3837 }
3838 \stex_style_apply:
3839 \group_end:
3840 }

```

(the symdef keyset is defined after notations as

`\stex_keys_define:nnnn{symdef}{-}{-}{decl,notation}`)

(End of definition for `\symdef`. This function is documented on page 147.)

sffunction

`\textsymdecl`

```

3841 \stex_keys_define:nnnn{textsymdecl}{
3842   \str_clear:N \l_stex_key_name_str
3843   \tl_clear:N \l_stex_key_type_tl
3844   \tl_clear:N \l_stex_key_def_tl
3845 }{
3846   name      .str_set:N = \l_stex_key_name_str ,
3847   type      .tl_set:N  = \l_stex_key_type_tl ,
3848   def       .tl_set:N  = \l_stex_key_def_tl,
3849   gf        .code:n    = {}
3850 }{style,deprecate}
3851
3852 \stex_new_stylable_cmd:nnnn {textsymdecl} {m 0{ } m} {
3853   \stex_keys_set:nn{symdef}{-}
3854   \stex_keys_set:nn{textsymdecl}{#2}
3855   \str_set:Ne \l_stex_macroname_str { #1 }
3856   \str_if_empty:NT \l_stex_key_name_str {
3857     \str_set:Nn \l_stex_key_name_str {#1}
3858   }%{
3859   % \str_set:Ne \l_stex_key_name_str {\l_stex_key_name_str-sym}
3860   %}
3861   \str_set:Nn \l_stex_key_role_str {textsymdecl}
3862   \tl_set:Ne \l_stex_current_symbol_uri {
3863     \exp_args:No \stex_new_symbol_uri:n \l_stex_key_name_str
3864   }
3865   \stex_symdecl_do:
3866   \stex_check_terms:
3867   \exp_args:Nne \use:nn {\stex_add_symbol:nnnnnnN}{
3868     {\l_stex_macroname_str}
3869     {\l_stex_key_name_str}
3870     {0}{-}

```



```

3871     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
3872     {}% type
3873     {\use:c{#1name_nospace}}}% return
3874     \stex_invoke_text_symbol:
3875 }
3876 \exp_args:Ne \stex_ref_new_symbol:n
3877 { \stex_use_symbol_uri:N \l_stex_current_symbol_uri }
3878 \stex_if_do_html:T {
3879     \stex_symdecl_html:
3880 }
3881
3882 \int_set:Nn \l_stex_get_symbol_arity_int 0
3883 \tl_clear:N \l_stex_key_op_tl
3884 \str_clear:N \l_stex_key_intent_str
3885 \str_clear:N \l_stex_key_prec_str
3886 \tl_set_eq:NN \l_stex_get_symbol_uri \l_stex_current_symbol_uri
3887 \stex_notation_parse:n{\hbox{#3}}
3888 \stex_notation_add:
3889 \stex_if_do_html:T {
3890     \def\comp{\_comp}
3891     \stex_notation_do_html:n{ \stex_use_symbol_uri:N \l_stex_current_symbol_uri }
3892 }
3893 \stex_execute_in_module:e{
3894     \__stex_syms_set_textsymdecl_macro:nnn{#1}{ \stex_use_symbol_uri:N \l_stex_current_symbol_uri }
3895     \exp_not:n{#3}}
3896 }
3897
3898 \stex_if_smsmode:F{
3899     \group_begin:
3900     \tl_set:Ne \thisdecluri { \stex_use_symbol_uri:N \l_stex_current_symbol_uri }
3901     \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3902     \tl_clear:N \thisstyle
3903     \stex_style_apply:
3904     \group_end:
3905 }
3906 \stex_smsmode_do:
3907 }{}
3908
3909 \stex_deactivate_macro:Nn \textsymdecl {module~environments}
3910 \stex_every_module:n { \stex_reactivate_macro:N \textsymdecl }
3911 \stex_sms_allow_escape:N \textsymdecl
3912
3913 \cs_new_protected:Nn \__stex_syms_set_textsymdecl_macro:nnn {
3914     \cs_set_protected:cpn{#1name_nospace}{#3}
3915     \cs_set_protected:cpn{#1name}{
3916         \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3917         \mode_if_math:T{\hbox{\let\xspace\relax #3}
3918         \mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3919     }
3920 }

```

(End of definition for `\textsymdecl`. This function is documented on page 147.)

sfunction

`__stex_syms_top:n` shared behaviour for `\symdecl` and `\symdef`; calls `\stex_symdecl_do:`, optionally checks

the term components, adds the symbol to the current module and produces the FTML for the declaration.

```

3921 \cs_new_protected:Nn \__stex_syms_top:n {
3922   \str_if_empty:NT \l_stex_key_name_str {
3923     \str_set:Ne \l_stex_key_name_str { #1 }
3924   }
3925   \tl_set:Ne \l_stex_current_symbol_uri {
3926     \exp_args:No \stex_new_symbol_uri:n \l_stex_key_name_str
3927   }
3928
3929   \stex_symdecl_do:
3930   \_stex_check_terms:
3931   \__stex_syms_add_decl:
3932   \stex_if_do_html:T \_stex_symdecl_html:
3933 }

```

(End of definition for __stex_syms_top:n.)

Adds the symbol to the current module:

```

3934 \cs_new_protected:Nn \__stex_syms_add_decl: {
3935   \exp_args:Nne \use:nn {\stex_add_symbol:nnnnnnN}{
3936     {\l_stex_macroname_str}
3937     {\l_stex_key_name_str}
3938     {\int_use:N \l_stex_get_symbol_arity_int}
3939     {\l_stex_get_symbol_args_tl}
3940     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
3941     {}
3942     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
3943     \stex_invoke_symbol:
3944   }
3945   \exp_args:Ne \stex_ref_new_symbol:n
3946     {\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
3947 }

```

\stex_symdecl_do:

```

3948 \cs_new_protected:Nn \stex_symdecl_do: {
3949   \_stex_do_deprecation:n \l_stex_key_name_str
3950   \__stex_syms_parse_arity:
3951   \__stex_syms_do_args:
3952 }
3953
3954 \cs_new:Nn \_stex_return_args:nn {
3955   {\svar{ARGUMENT_#1}\_stex_eat_exclamation_point:}
3956 }
3957
3958 \cs_new_protected:Nn \__stex_syms_do_args: {
3959   \tl_clear:N \l_stex_get_symbol_args_tl
3960   \int_step_inline:nn \l_stex_get_symbol_arity_int {
3961     \tl_put_right:Nn \l_stex_get_symbol_args_tl {##1}
3962     \tl_put_right:Ne \l_stex_get_symbol_args_tl {
3963       \str_item:Nn \l_stex_key_args_str {##1}
3964     }
3965   }
3966 }

```

Constructs the arity string of the symbol:

```

3967 \int_new:N \l_stex_assoc_args_count
3968
3969 \cs_new_protected:Nn \__stex_syms_parse_arity: {
3970   \int_zero:N \l_stex_get_symbol_arity_int
3971   \int_zero:N \l_stex_assoc_args_count
3972   \str_map_inline:Nn \l_stex_key_args_str {
3973     \str_case:nnF ##1 {
3974       0 { \str_map_break: }
3975       1 { \str_map_break:n{
3976         \int_set:Nn \l_stex_get_symbol_arity_int {1}
3977         \str_set:Nn \l_stex_key_args_str {i}
3978       } }
3979       2 { \str_map_break:n{
3980         \int_set:Nn \l_stex_get_symbol_arity_int {2}
3981         \str_set:Nn \l_stex_key_args_str {ii}
3982       } }
3983       3 { \str_map_break:n{
3984         \int_set:Nn \l_stex_get_symbol_arity_int {3}
3985         \str_set:Nn \l_stex_key_args_str {iii}
3986       } }
3987       4 { \str_map_break:n{
3988         \int_set:Nn \l_stex_get_symbol_arity_int {4}
3989         \str_set:Nn \l_stex_key_args_str {iiii}
3990       } }
3991       5 { \str_map_break:n{
3992         \int_set:Nn \l_stex_get_symbol_arity_int {5}
3993         \str_set:Nn \l_stex_key_args_str {iiiii}
3994       } }
3995       6 { \str_map_break:n{
3996         \int_set:Nn \l_stex_get_symbol_arity_int {6}
3997         \str_set:Nn \l_stex_key_args_str {iiiiiii}
3998       } }
3999       7 { \str_map_break:n{
4000         \int_set:Nn \l_stex_get_symbol_arity_int {7}
4001         \str_set:Nn \l_stex_key_args_str {iiiiiii}
4002       } }
4003       8 { \str_map_break:n{
4004         \int_set:Nn \l_stex_get_symbol_arity_int {8}
4005         \str_set:Nn \l_stex_key_args_str {iiiiiii}
4006       } }
4007       9 { \str_map_break:n{
4008         \int_set:Nn \l_stex_get_symbol_arity_int {9}
4009         \str_set:Nn \l_stex_key_args_str {iiiiiii}
4010       } }
4011       i {\int_incr:N \l_stex_get_symbol_arity_int}
4012       b {\int_incr:N \l_stex_get_symbol_arity_int}
4013       a {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
4014       B {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
4015     }{
4016       \msg_error:nnxx{stex}{error/wrongargs}{-}{##1}
4017     }
4018   }
4019 }
```

(End of definition for `\stex_symdecl_do`:. This function is documented on page 147.)

sfunction

`_stex_symdecl_html`:

```

4020 \cs_new_protected:Nn \_stex_symdecl_html: {
4021   \stex_annotate_invisible:n{
4022     \hbox{\exp_args:Ne \stex_annotate:nn {
4023       data-ftml-symdecl = { \l_stex_key_name_str},
4024       data-ftml-args = {\l_stex_key_args_str}
4025       \str_if_empty:NF \l_stex_macroname_str {,
4026         data-ftml-macroname={\l_stex_macroname_str}
4027       }
4028       \str_if_empty:NF \l_stex_key_wikidata_str {,
4029         data-ftml-wikidata={\l_stex_key_wikidata_str}
4030       }
4031       \str_if_empty:NF \l_stex_key_assoc_str {,
4032         data-ftml-assoctype={\l_stex_key_assoc_str}
4033       }
4034       \str_if_empty:NF \l_stex_key_reorder_str {,
4035         data-ftml-reorderargs={\l_stex_key_reorder_str}
4036       }
4037       \str_if_empty:NF \l_stex_key_role_str {,
4038         data-ftml-role={\l_stex_key_role_str}
4039       }
4040     }\stex_annotate_force_break:n{
4041       \bool_set_true:N \stex_in_invisible_html_bool
4042       \tl_if_empty:NF \l_stex_key_type_tl {
4043         $\stex_annotate:nn{data-ftml-type={}}{\l_stex_key_type_tl}$
4044       }
4045       \tl_if_empty:NF \l_stex_key_def_tl {
4046         $\stex_annotate:nn{data-ftml-definiens={}}{\l_stex_key_def_tl}$
4047       }
4048       \tl_if_empty:NF \l_stex_key_return_tl{
4049         \exp_args:Nno \use:n{
4050           \cs_generate_from_arg_count:NNnn \l__stex_syms_cs
4051           \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4052           \tl_set:Ne \l__stex_syms_args_tl {\_stex_map_args:N \_stex_return_args:nn}
4053           $\stex_annotate:nn{data-ftml-returntype={}}{
4054             \exp_after:wN \l__stex_syms_cs \l__stex_syms_args_tl!
4055           }$
4056         }
4057       \clist_if_empty:NF \l_stex_key_argtypes_clist {
4058         \stex_annotate:nn{data-ftml-argtypes={}}{\_stex_annotate_force_break:n{
4059           \clist_map_inline:Nn \l_stex_key_argtypes_clist {
4060             $\stex_annotate:nn{data-ftml-type={}}{##1}$
4061           }
4062         }}
4063       }
4064     }}
4065   }}
4066 }

```

(End of definition for `_stex_symdecl_html`:. This function is documented on page 147.)

sfunction

\stex_add_symbol:nnnnnnN

```

4067 \cs_new_protected:Nn \stex_add_symbol:nnnnnnN {
4068   \stex_debug:nn{declaration}{New~declaration:~\stex_use_module_uri:N \l_stex_current_modul
4069     Macro:#1^^JArity:#3~(#4)^^J
4070     Def:~\tl_to_str:n{#5}^^J
4071     Type:~\tl_to_str:n{#6}^^J
4072     Returns:~\tl_to_str:n{#7}^^J
4073     Invokes:~\tl_to_str:n{#8}
4074   }
4075   \prop_gput:cnn{ \_stex_symbol_macro:N \l_stex_current_module_uri }
4076   {#2}{{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}
4077   \tl_if_empty:nF{#1}{
4078     \stex_do_up_to_module:n {
4079       \__stex_syms_activate_i:nnnnnnnn{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}
4080     }
4081   }
4082 }

```

Generate semantic macros etc.:

```

4083 \cs_new_protected:Nn \_stex_activate_symbols: {
4084   \stex_debug:nn{activating}{All~symbols:~\cs_meaning:c{ \_stex_symbol_macro:N \l_stex_curr
4085   \prop_map_inline:cn{ \_stex_symbol_macro:N \l_stex_current_module_uri }{
4086     \__stex_syms_maybe_activate:nnnnnnnn ##2
4087   }
4088 }
4089
4090 \cs_new_protected:Nn \__stex_syms_maybe_activate:nnnnnnnn {
4091   \str_if_empty:nF{#1}{
4092     \__stex_syms_activate_i:nnnnnnnn {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}
4093   }
4094 }
4095
4096 \cs_new_protected:Nn \__stex_syms_activate_i:nnnnnnnn {
4097   \stex_debug:nn{activating}{macro~#1:\stex_use_module_uri:N \l_stex_current_module_uri ? #
4098   \tl_to_str:n{{#3}{#4}{#5}{#6}{#7}{#8}}
4099 }
4100   \cs_set:cpe{#1} {
4101     \_stex_invoke_symbol:nnnnnnN
4102     {\exp_args:No \stex_new_symbol_uri:nn {\l_stex_current_module_uri} {#2} }
4103     \exp_not:n{{#3}{#4}{#5}{#6}{#7}{#8}}
4104   }
4105 }

```

(End of definition for \stex_add_symbol:nnnnnnN. This function is documented on page 147.)

sfunction

\stex_has_definiens_p:N

\stex_has_definiens:N \overline{TF}

\stex_has_definiens_p:n

\stex_has_definiens:n \overline{TF}

```

4106 \prg_new_conditional:Nnn \stex_has_definiens:N { p, T, F, TF } {
4107   \exp_args:No \stex_has_definiens:nTF #1 \prg_return_true: \prg_return_false:
4108 }
4109 \prg_new_conditional:Nnn \stex_has_definiens:n { p, T, F, TF } {
4110   \exp_args:Nne \use:nn{\__stex_syms_has_definiens:nnnnnnnn}{\exp_args:Nne \prop_item:cn{
4111     \exp_args:Ne \_stex_symbol_macro:n {\stex_symbol_uri_module:n{#1}}
4112   }{

```

```

4113     \stex_symbol_uri_name:n {#1}
4114   }}
4115 }
4116
4117 \cs_new:Nn \__stex_syms_has_definiens:nnnnnnnN {
4118   \tl_if_empty:nTF{#5}\prg_return_false:\prg_return_true:
4119 }

```

(End of definition for `\stex_has_definiens:NTF` and `\stex_has_definiens:nTF`. These functions are documented on page 147.)

sfunction

32.2 Retrieval

```

\stex_iterate_symbols:n
  \stex_iterate_break:
  \stex_iterate_break:n
4120 \cs_new_protected:Nn \stex_iterate_symbols:n {
4121   \stex_pseudogroup_with:nn{\__stex_syms_sym_cs:nnnnnnnnN\stex_iterate_break:\stex_iterate_
4122     \cs_set:Npn \__stex_syms_sym_cs:nnnnnnnnN
4123     ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
4124     \cs_set:Npn \stex_iterate_break: {
4125       \prop_map_break:n{\seq_map_break:}
4126     }
4127     \cs_set:Npn \stex_iterate_break:n ##1 {
4128       \prop_map_break:n{\seq_map_break:n{##1}}
4129     }
4130     \seq_map_inline:Nn \l_stex_all_modules_seq {
4131       \prop_map_inline:cn{\_stex_symbol_macro:n {##1}}{
4132         \__stex_syms_sym_cs:nnnnnnnnN {##1} #####2
4133       }
4134     }
4135   }
4136 }

```

(End of definition for `\stex_iterate_symbols:n`, `\stex_iterate_break:`, and `\stex_iterate_break:n`. These functions are documented on page 148.)

sfunction

```

\stex_iterate_symbols:nn
4137 \cs_new_protected:Nn \stex_iterate_symbols:nn {
4138   \seq_clear:N \l__stex_syms_mods_seq
4139   \stex_pseudogroup_with:nn{\__stex_syms_sym_cs:nnnnnnnnN}{
4140     \cs_set:Npn \__stex_syms_sym_cs:nnnnnnnnN
4141     ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #2 }
4142     \clist_map_function:nN {#1} \__stex_syms_it_decl_i:n
4143   }
4144 }
4145
4146 \cs_new_protected:Nn \__stex_syms_it_decl_i:n {
4147   \seq_if_in:NnF \l__stex_syms_mods_seq {#1} {
4148     \seq_put_left:Nn \l__stex_syms_mods_seq {#1}
4149     \prop_map_inline:cn{\_stex_morphisms_macro:n {#1}}{
4150       \__stex_syms_it_decl_check:nnnn ##2
4151     }

```

```

4152 \prop_map_inline:cn{\_stex_symbol_macro:n {#1} }{
4153   \__stex_syms_sym_cs:nnnnnnnnN {#1} ##2
4154 }
4155 }
4156 }
4157
4158 \cs_new_protected:Nn \__stex_syms_it_decl_check:nnnn {
4159   \tl_if_empty:nT{#1}{
4160     \__stex_syms_it_decl_i:n {#2}
4161   }
4162 }

```

(End of definition for `\stex_iterate_symbols:nn`. This function is documented on page 148.)

sfunction

```

\stex_get_symbol:n
\stex_get_symbol:nTF
\stex_get_symbol:nF
4163 \int_new:N \l_stex_get_symbol_arity_int
4164
4165 \cs_new_protected:Nn \stex_get_symbol:n {
4166   \stex_get_symbol:nF{ #1 }{
4167     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4168   }
4169 }
4170
4171 \cs_new_protected:Npn \stex_get_symbol:nTF #1 #2 #3 {
4172   \tl_clear:N \l_stex_get_symbol_uri
4173   \cs_if_exist:cTF { #1 }{
4174     \cs_set_eq:Nc \l__stex_syms_cs { #1 }
4175     % command name
4176     \exp_args:Ne \tl_if_empty:nTF { \cs_argument_spec:N \l__stex_syms_cs }{
4177       % ...that takes no arguments
4178       \exp_args:Ne \cs_if_eq:NNTF {\tl_head:N \l__stex_syms_cs}
4179         \stex_invoke_symbol:nnnnnnN
4180         \__stex_syms_get_symbol_from_cs:
4181         {\__stex_syms_get_symbol_from_string:n { #1 }}
4182     }{
4183       \__stex_syms_get_symbol_from_string:n { #1 }
4184     }
4185   }{
4186     \__stex_syms_get_symbol_from_string:n { #1 }
4187   }
4188   \tl_if_empty:nTF \l_stex_get_symbol_uri { #3 } { #2 }
4189 }
4190
4191 \cs_new_protected:Npn \stex_get_symbol:nF #1 #2 {
4192   \stex_get_symbol:nTF{ #1 }{}{ #2 }
4193 }
4194
4195 \cs_new_protected:Nn \__stex_syms_get_symbol_from_cs: {
4196   \stex_pseudogroup_with:nn{\stex_invoke_symbol:nnnnnnN}{
4197     \cs_set:Npn \stex_invoke_symbol:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 {
4198       \tl_set:Nn \l_stex_get_symbol_uri { ##1 }
4199       \int_set:Nn \l_stex_get_symbol_arity_int {##2}
4200       \tl_set:Nn \l_stex_get_symbol_args_tl {##3}

```

```

4201     \tl_set:Nn \l_stex_get_symbol_def_tl {##4}
4202     \tl_set:Nn \l_stex_get_symbol_type_tl {##5}
4203     \tl_set:Nn \l_stex_get_symbol_return_tl {##6}
4204     \tl_set:Nn \l_stex_get_symbol_invoke_cs {##7}
4205   }
4206   \l__stex_syms_cs
4207 }
4208 }
4209
4210 \cs_new_protected:Nn \__stex_syms_get_symbol_from_string:n {
4211   \stex_debug:nn{symbols}{Getting~from~string~#1...}
4212   \seq_set_split:Nnn \l__stex_syms_seq ? {#1}
4213   \seq_pop_right:NN \l__stex_syms_seq \l__stex_syms_name
4214   \seq_if_empty:NTF \l__stex_syms_seq {
4215     \exp_args:No \__stex_syms_get_from_one_string:n {#1}
4216   }{
4217     \exp_args:NNe \exp_args:Nno \__stex_syms_get_symbol_from_modules:nn {
4218       \seq_use:Nn \l__stex_syms_seq ?
4219     } \l__stex_syms_name
4220   }
4221 }
4222
4223 \cs_new_protected:Nn \__stex_syms_sym_from_str_i:nnnn {
4224   \bool_lazy_any:nTF{
4225     {\str_if_eq_p:nn{#2}{#3}}
4226     {\str_if_eq_p:nn{#2}{#4}}
4227     {\stex_str_if_ends_with_p:nn{#4}{/#2}}
4228   }{
4229     \__stex_syms_sym_i_finish:nnnnnnN{#1}{#4}
4230   }{
4231     \__stex_syms_sym_i_gobble:nnnnnn
4232   }
4233 }
4234 \cs_new_protected:Nn \__stex_syms_sym_i_gobble:nnnnnn {}
4235
4236 \cs_new_protected:Nn \__stex_syms_sym_i_finish:nnnnnnN {
4237   \prop_map_break:n{\seq_map_break:n{
4238     \tl_set:Ne \l_stex_get_symbol_uri { \stex_new_symbol_uri:nn {#1} {#2}}
4239     \int_set:Nn \l_stex_get_symbol_arity_int {#3}
4240     \tl_set:Nn \l_stex_get_symbol_args_tl {#4}
4241     \tl_set:Nn \l_stex_get_symbol_def_tl {#5}
4242     \tl_set:Nn \l_stex_get_symbol_type_tl {#6}
4243     \tl_set:Nn \l_stex_get_symbol_return_tl {#7}
4244     \tl_set:Nn \l_stex_get_symbol_invoke_cs {#8}
4245   }}
4246 }
4247
4248 \cs_new_protected:Nn \__stex_syms_get_symbol_from_modules:nn {
4249   %\seq_set_split:Nnn \l_tmpa_seq ? {#1}
4250   %\seq_pop_right:NN \l_tmpa_seq \l__stex_syms_name_str
4251   %\str_set:Ne \l__stex_syms_path_str {\seq_use:Nn \l_tmpa_seq ?}
4252   \stex_debug:nn{symbols}{Getting~#2~in~#1...}
4253   \seq_map_inline:Nn \l_stex_all_modules_seq {
4254     %\stex_debug:nn{symbols}{comparing~\stex_module_uri_as_qm:n{##1}~and~#1}

```



```

4255 \exp_args:Ne \stex_str_if_ends_with:nnT{\stex_module_uri_as_qm:n{##1}}{#1}{
4256 \prop_map_inline:cn{\_stex_symbol_macro:n {##1} }{
4257 \_stex_syms_sym_from_str_i:nnnn{##1}{#2} ####2
4258 }
4259 }
4260
4261 %\str_if_empty:NTF \l__stex_syms_path_str {
4262 % \exp_args:NNe \exp_args:Nno \stex_str_if_ends_with:nnT {\stex_module_uri_name:n{##1}}
4263 %}{
4264 % \exp_args:NNNe \exp_args:No \str_if_eq:nnT\l__stex_syms_name_str{
4265 % \stex_module_uri_name:n {##1}
4266 % }
4267 %}{
4268 % \str_if_empty:NTF \l__stex_syms_path_str {
4269 % \prop_map_inline:cn{\_stex_symbol_macro:n {##1} }{
4270 % \_stex_syms_sym_from_str_i:nnnn{##1}{#2} ####2
4271 % }
4272 % }{
4273 % \stex_debug:nn{symbols}{Comparing~\l__stex_syms_path_str;~with~\tl_to_str:n{##1}}
4274 % \exp_args:NNe \exp_args:Nno \stex_str_if_ends_with:nnT{\stex_module_uri_path:n {##1}}
4275 % \prop_map_inline:cn{\_stex_symbol_macro:n {##1} }{
4276 % \_stex_syms_sym_from_str_i:nnnn{##1}{#2} ####2
4277 % }
4278 % }
4279 % }
4280 %}
4281 }
4282 }
4283
4284 \prg_new_conditional:Nnn \_stex_syms_uri_match:n {T,F,TF} {
4285 \str_if_eq:nnT
4286 }
4287
4288
4289 \cs_new_protected:Nn \_stex_syms_get_from_one_string:n {
4290 \stex_debug:nn{symbols}{Getting~#1~anywhere...}
4291 \stex_iterate_symbols:n{
4292 %\stex_debug:nn{symbols}{>#1==#2~|~#1==#3<...}
4293 \bool_lazy_any:nT{
4294 {\str_if_eq_p:nn{#1}{##2}}
4295 {\str_if_eq_p:nn{#1}{##3}}
4296 {\stex_str_if_ends_with_p:nn{##3}{/#1}}
4297 }{
4298 \stex_iterate_break:n{
4299 \tl_set:Ne \l_stex_get_symbol_uri { \stex_new_symbol_uri:nn {##1} {##3}}
4300 \int_set:Nn \l_stex_get_symbol_arity_int {##4}
4301 \tl_set:Nn \l_stex_get_symbol_args_tl {##5}
4302 \tl_set:Nn \l_stex_get_symbol_def_tl {##6}
4303 \tl_set:Nn \l_stex_get_symbol_type_tl {##7}
4304 \tl_set:Nn \l_stex_get_symbol_return_tl {##8}
4305 \tl_set:Nn \l_stex_get_symbol_invoke_cs {##9}
4306 }
4307 }
4308 }

```

4309 }

(End of definition for `\stex_get_symbol:n`, `\stex_get_symbol:nTF`, and `\stex_get_symbol:nF`. These functions are documented on page 148.)

sfunction

32.3 Variables

4310 `<@@=stex_vars>`

`\l_stex_variables_prop`

4311 `\prop_new:N \l_stex_variables_prop`

(End of definition for `\l_stex_variables_prop`. This variable is documented on page 148.)

`\vardef`

```

4312 \bool_new:N \l__stex_vars_bind_bool
4313 \cs_new_protected:Nn \_stex_variable:nnnnnnnN {}
4314
4315 \stex_new_stylable_cmd:nnnn {vardef} { m O{} m} {
4316   \stex_keys_set:nn{vardef}{#2}
4317   \str_set:Ne \l_stex_macroname_str { #1 }
4318   \str_if_empty:NT \l_stex_key_name_str {
4319     \str_set:Ne \l_stex_key_name_str { #1 }
4320   }
4321
4322   \stex_symdecl_do:
4323   \stex_if_check_terms:T \_stex_check_terms:
4324   \__stex_vars_add:
4325   \__stex_vars_macro:
4326   \stex_if_do_html:T \__stex_vars_html:
4327
4328   \int_set:Nn \l_stex_get_symbol_arity_int {\l_stex_get_symbol_arity_int}
4329   \stex_debug:nn{vardef}{Doing~\l_stex_key_name_str}
4330   \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4331   \stex_notation_parse:n{#3}
4332   \stex_if_check_terms:T{ \_stex_notation_check: }
4333   \_stex_var_notation_macro:
4334   \stex_if_do_html:T {
4335     \def\comp{\_varcomp}
4336     \_stex_notation_do_html:n \l_stex_key_name_str
4337   }
4338   \group_begin:
4339   \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4340   \tl_clear:N \thisstyle
4341   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4342   \def\thisnotation{
4343     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{ }{
4344       \_stex_notation_make_args:
4345     }
4346   }
4347   \stex_style_apply:
4348   \group_end:\ignorespaces
4349 }{}

```

(End of definition for `\vardef`. This function is documented on page 148.)

sfunction

Adds a new variable to the current scope:

```

4350 \cs_new_protected:Nn \__stex_vars_add: {
4351   \exp_args:NNNo \exp_args:NNne
4352   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4353     {\l_stex_macroname_str}
4354     {\l_stex_key_name_str}
4355     {\int_use:N \l_stex_get_symbol_arity_int}
4356     {\l_stex_get_symbol_args_tl}
4357     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4358     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4359     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4360     \stex_invoke_symbol:
4361   }
4362 }
```

Defines the macro for a variable:

```

4363 \cs_new_protected:Nn \__stex_vars_macro: {
4364   \tl_set:ce{\l_stex_macroname_str}{
4365     \_stex_invoke_variable:nnnnnnN
4366     {\l_stex_key_name_str}
4367     {\int_use:N \l_stex_get_symbol_arity_int}
4368     {\l_stex_get_symbol_args_tl}
4369     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4370     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4371     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4372     \stex_invoke_symbol:
4373   }
4374 }
```

Insert the HTML for a variable declaration:

```

4375
4376 \cs_new_protected:Nn \__stex_vars_html: {
4377   \stex_if_do_html:T {
4378     \hbox\bgroup\exp_args:Ne \stex_annotate_invisible:nn {
4379       data-ftml-vardef = {\l_stex_key_name_str},
4380       data-ftml-args = {\l_stex_key_args_str}
4381       \str_if_empty:NF \l_stex_macroname_str {,
4382         data-ftml-macroname={\l_stex_macroname_str}
4383       }
4384       \str_if_empty:NF \l_stex_key_assoc_str {,
4385         data-ftml-assoctype={\l_stex_key_assoc_str}
4386       }
4387       \str_if_empty:NF \l_stex_key_role_str {,
4388         data-ftml-role={\l_stex_key_role_str}
4389       }
4390       \str_if_empty:NF \l_stex_key_reorder_str {,
4391         data-ftml-reorderargs={\l_stex_key_reorder_str}
4392       }
4393       \bool_if:NT \l__stex_vars_bind_bool {,
4394         data-ftml-bind={true}
4395       }
4396     }\}
```

```

4397 \stex_annotate_force_break:n{
4398   \bool_set_true:N \stex_in_invisible_html_bool
4399   \tl_if_empty:NF \l_stex_key_type_tl {
4400     \stex_annotate:nn{data-ftml-type={}}{${\l_stex_key_type_tl$}
4401   }
4402   \tl_if_empty:NF \l_stex_key_def_tl {
4403     \stex_annotate:nn{data-ftml-definiens={}}{${\l_stex_key_def_tl$}
4404   }
4405   \tl_if_empty:NF \l_stex_key_return_tl{
4406     \exp_args:Nno \use:n{
4407       \cs_generate_from_arg_count:NNnn \l__stex_vars_cs
4408       \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4409       \tl_set:Ne \l__stex_vars_args_tl {\stex_map_args:N \stex_return_args:nn}
4410       ${\stex_annotate:nn{data-ftml-returntype={}}}{
4411         \exp_after:wN \l__stex_vars_cs \l__stex_vars_args_tl!}$
4412     }
4413     \tl_if_empty:NF \l_stex_key_argtypes_clist {
4414       \stex_annotate:nn{data-ftml-argtypes={}}{
4415         \stex_annotate_force_break:n{
4416           \clist_map_inline:Nn \l_stex_key_argtypes_clist {
4417             ${\stex_annotate:nn{data-ftml-type={}}}{##1}$
4418           }
4419         }
4420       }
4421     }
4422   }
4423 } \egroup
4424 }
4425 }

```

`\newvar`

```

4426 \NewDocumentCommand\newvar{ m O{} }{
4427   \vardef{v#1}[name=#1,#2]{#1}
4428 }

```

(End of definition for `\newvar`. This function is documented on page 148.)

sfunction

`\newseq`

```

4429 \NewDocumentCommand\newseq{ m O{} m }{
4430   \varseq{v#1}[name=#1,#2]{#3}{\maincomp{#1}}\c_math_subscript_token{##1}}
4431 }

```

(End of definition for `\newseq`. This function is documented on page 148.)

sfunction

`\stex_get_var:n`

```

4432 \cs_new_protected:Nn \stex_get_var:n {
4433   \str_clear:N \l_stex_get_variable_str
4434   \__stex_vars_get_var:n{#1}
4435   \str_if_empty:NT \l_stex_get_variable_str {
4436     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4437   }
4438 }

```

```

4439
4440 \cs_new_protected:Nn \__stex_vars_get_var:n {
4441   \prop_map_inline:Nn \l_stex_variables_prop {
4442     \__stex_vars_check_var:nnnnnnnnN {#1} ##2
4443   }
4444 }
4445
4446 \cs_new_protected:Nn \__stex_vars_check_var:nnnnnnnnN {
4447   \str_if_eq:nnTF{#1}{#2}{
4448     \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}}
4449   }{
4450     \str_if_eq:nnT{#1}{#3}{
4451       \prop_map_break:n{\__stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}}
4452     }
4453   }
4454 }
4455
4456 \cs_new_protected:Nn \__stex_vars_set_vars:nnnnnnN {
4457   \stex_debug:nn{symbols}{Variable~#1~found}
4458   \cs_set:Npn \_stex_variable:nnnnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {}
4459   \str_set:Nn \l_stex_get_variable_str {#1}
4460   \int_set:Nn \l_stex_get_symbol_arity_int {#2}
4461   \tl_set:Nn \l_stex_get_symbol_args_tl {#3}
4462   \tl_set:Nn \l_stex_get_symbol_def_tl {#4}
4463   \tl_set:Nn \l_stex_get_symbol_type_tl {#5}
4464   \tl_set:Nn \l_stex_get_symbol_return_tl {#6}
4465   \tl_set:Nn \l_stex_get_symbol_invoke_cs {#7}
4466 }

```

(End of definition for `\stex_get_var:n`. This function is documented on page 149.)

sfunction

32.3.1 Variable Sequences

```

4467 <@@=stex_seqs>

\varseq

4468 \stex_new_stylable_cmd:nnnn {varseq}{m 0}{ m m} {
4469   \stex_keys_set:nn{symdef}{#2}
4470   \str_set:Ne \l_stex_macroname_str { #1 }
4471   \str_if_empty:NT \l_stex_key_name_str {
4472     \str_set:Ne \l_stex_key_name_str { #1 }
4473   }
4474   \str_if_empty:NT \l_stex_key_args_str {
4475     \str_set:Nn \l_stex_key_args_str {1}
4476   }
4477   \stex_symdecl_do:
4478
4479   \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4480   \clist_set:Nn \l__stex_seqs_range_clist {#3}
4481   \tl_if_empty:NTF \l_stex_key_op_tl {
4482     \stex_notation_parse:n{#4}
4483     \tl_clear:N \l_stex_key_op_tl
4484   }{

```

```

4485     \stex_notation_parse:n{#4}
4486   }
4487   \stex_if_do_html:T \__stex_seqs_html:
4488   \stex_if_check_terms:T \_stex_check_terms:
4489   \__stex_seqs_add:
4490   \__stex_seqs_macro:
4491   \stex_if_check_terms:T \_stex_notation_check:
4492   \_stex_var_notation_macro:
4493   \stex_if_do_html:T {
4494     \def\comp{\_varcomp}
4495     \_stex_notation_do_html:n \l_stex_key_name_str
4496   }
4497   \group_begin:
4498   \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4499   \tl_clear:N \thisstyle
4500   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4501   \def\thisnotation{
4502     \l_stex_notation_macrocode_cs{}!
4503   }
4504   \stex_style_apply:
4505   \group_end:\ignorespaces
4506 }{}
4507
4508 \cs_new_protected:Nn \__stex_seqs_add: {
4509   \exp_args:NNNo \exp_args:NNne
4510   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4511     {\l_stex_macroname_str}
4512     {\l_stex_key_name_str}
4513     {\int_use:N \l_stex_get_symbol_arity_int}
4514     {\l_stex_get_symbol_args_tl}
4515     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4516     {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
4517     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4518     \stex_invoke_sequence:
4519   }
4520 }
4521
4522 \cs_new_protected:Nn \__stex_seqs_macro: {
4523   \tl_set:ce{\l_stex_macroname_str}{
4524     \_stex_invoke_variable:nnnnnnN
4525     {\l_stex_key_name_str}
4526     {\int_use:N \l_stex_get_symbol_arity_int}
4527     {\l_stex_get_symbol_args_tl}
4528     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4529     {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
4530     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4531     \stex_invoke_sequence:
4532   }
4533 }
4534
4535 \cs_new_protected:Nn \__stex_seqs_html: {
4536   \exp_args:Ne \stex_annotate_invisible:nn {
4537     data-ftml-varseq = {\l_stex_key_name_str},
4538     data-ftml-args = {\l_stex_key_args_str}

```

```

4539 \str_if_empty:NF \l_stex_macroname_str {,
4540   data-ftml-macroname={\l_stex_macroname_str}
4541 }
4542 \str_if_empty:NF \l_stex_key_assoc_str {,
4543   data-ftml-assoctype={\l_stex_key_assoc_str}
4544 }
4545 \str_if_empty:NF \l_stex_key_role_str {,
4546   data-ftml-role={\l_stex_key_role_str}
4547 }
4548 \str_if_empty:NF \l_stex_key_reorder_str {,
4549   data-ftml-reorderargs={\l_stex_key_reorder_str}
4550 }
4551 }{\hbox\bgroup
4552   \stex_annotate_force_break:n{
4553
4554     $\stex_annotate:nn{data-ftml-seqrange={}}{
4555       \exp_args:Nno\symuse{Metatheory?sequence-range}\l__stex_seqs_range_clist
4556     }$
4557
4558     \tl_if_empty:NF \l_stex_key_type_tl {
4559       \stex_annotate:nn{data-ftml-type={}}{${\l_stex_key_type_tl$}
4560     }
4561     \tl_if_empty:NF \l_stex_key_def_tl {
4562       \stex_annotate:nn{data-ftml-definiens={}}{${\l_stex_key_def_tl$}
4563     }
4564     \tl_if_empty:NF \l_stex_key_return_tl{
4565       \exp_args:Nno \use:n{
4566         \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs
4567         \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4568         \tl_set:Nx \l__stex_seqs_args_tl {\_stex_map_args:N \_stex_return_args:nn}
4569         \stex_annotate:nn{data-ftml-returntype={}}{
4570           $\exp_after:wN \l__stex_seqs_cs \l__stex_seqs_args_tl!$}
4571       }
4572     \tl_if_empty:NF \l_stex_key_argtypes_clist {
4573       \stex_annotate:nn{data-ftml-argtypes={}}{
4574         \stex_annotate_force_break:n{
4575           \clist_map_inline:Nn \l_stex_key_argtypes_clist {
4576             \stex_annotate:nn{data-ftml-type={}}{###1$}
4577           }
4578         }
4579       }
4580     }
4581   }
4582 \egroup}
4583 }

```

(End of definition for \varseq. This function is documented on page 149.)

sfunction

32.4 Expressions

```

4584 <@@=stex_expr>

```

\symuse

```

4585 \cs_new_protected:Npn \symuse #1 {
4586   \stex_get_symbol:n{#1}
4587   \exp_args:Nno \use:n {\_stex_invoke_symbol:NeooooN
4588   \l_stex_get_symbol_uri
4589   {\int_use:N \l_stex_get_symbol_arity_int}
4590   \l_stex_get_symbol_args_tl
4591   \l_stex_get_symbol_def_tl
4592   \l_stex_get_symbol_type_tl
4593   \l_stex_get_symbol_return_tl}
4594   \l_stex_get_symbol_invoke_cs
4595 }

```

(End of definition for \symuse. This function is documented on page 149.)
sfunction

_stex_next_symbol:n

```

4596 \tl_new:N \l__stex_expr_reset_tl
4597
4598 \cs_new_protected:Nn \_stex_next_symbol:n {
4599   \tl_set:Ne \l_stex_every_symbol_tl {
4600     \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
4601       \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
4602         \exp_args:No \exp_not:n \l_stex_every_symbol_tl
4603       }
4604       \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
4605         \exp_args:No \exp_not:n \l__stex_expr_reset_tl
4606       }
4607     }
4608     \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
4609       \exp_args:No \exp_not:n \l_stex_every_symbol_tl
4610     }
4611     \exp_not:n{ \aftergroup \l__stex_expr_reset_tl }
4612     \exp_not:N \l_stex_every_symbol_tl
4613     \exp_not:n{ #1 }
4614   }
4615 }
4616 \cs_generate_variant:Nn \_stex_next_symbol:n {e}

```

(End of definition for _stex_next_symbol:n. This function is documented on page 149.)
sfunction

_stex_invoke_symbol:NnnnnnN
_stex_invoke_symbol:NeooooN
_stex_invoke_symbol:nnnnnnN

- \l_stex_current_symbol_uri,
- \l_stex_current_symbol_arity_int,
- \l_stex_current_symbol_args_tl,
- definiens (currently not used),
- \l_stex_current_symbol_type_tl,
- \l_stex_current_symbol_return_tl,

- the invocation macro (is called after setup).

```

4617 \cs_new_protected:Nn \_stex_invoke_symbol:nnnnnnN {
4618   \bool_if:NTF \l_stex_allow_semantic_bool{
4619     \group_begin:
4620     \tl_set:Nn \l_stex_current_symbol_uri {#1}
4621     \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_current_symbol_uri
4622     \tl_set:Nn \l_stex_current_display_tl {
4623       \stex_symbol_uri_name:N \l_stex_current_symbol_uri
4624     }
4625     \__stex_expr_invoke:nnnnN{#2}{#3}{#5}{#6}{#7}
4626   }{
4627     \msg_error:nnxx{stex}{error/notallowed}{
4628       \stex_use_symbol_uri:n{#1}
4629     }{
4630       \l_stex_current_full_tl
4631     }
4632   }
4633 }
4634
4635 \cs_new_protected:Npn \_stex_invoke_symbol:NnnnnnnN {
4636   \exp_args:No \_stex_invoke_symbol:nnnnnnN
4637 }
4638 \cs_generate_variant:Nn \_stex_invoke_symbol:NnnnnnnN {NeooooN}

```

Whether semantic macros are allowed currently. This is false e.g. in the body of a semantic macro outside of one of its arguments:

```

4639 \bool_new:N \l_stex_allow_semantic_bool
4640 \bool_set_true:N \l_stex_allow_semantic_bool

```

Code to execute every time a semantic macro is invoked:

```

4641 \tl_set:Nn \l_stex_every_symbol_tl {
4642   \bool_set_false:N \l_stex_allow_semantic_bool
4643 }

```

The current top-level term; may be undefined:

```

4644 \tl_new:N \l_stex_current_term_tl

```

Keeps track of all set up so it can be reexecuted; this may be necessary to e.g. typeset `\this` in a struct field:

```

4645 \tl_new:N \l_stex_current_redo_tl

```

Setup for symbols; takes: arity, args, type, return, invoke. Calls invoke at the end.

In HTML mode, we make sure we enter horizontal mode *before* any annotations are inserted

```

4646 \cs_new_protected:Nn \__stex_expr_invoke:nnnnN {
4647   \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
4648   \__stex_expr_setup:nnnnN{\comp}{#1}{#2}{#4}{#3}
4649   \cs_set_eq:NN \_stex_term_oms_or_omv:nn \_stex_term_oms:nn
4650   \tl_put_right:Nn \l_stex_current_redo_tl{
4651     \cs_set_eq:NN \_stex_term_oms_or_omv:nn \_stex_term_oms:nn
4652   }
4653   #5
4654 }

```

general setup; takes: comp, arity, args, return, type

```

4655 \cs_new_protected:Nn \__stex_expr_setup:nnnnn {
4656   \tl_clear:N \l_stex_return_notation_tl
4657   \tl_set:Nn \l_stex_current_redo_tl {
4658     \let \this \stex_current_this:
4659     \def\comp{#1}
4660     \def\maincomp{\comp}
4661     \str_set:Nn \l_stex_current_arity_str{ #2 }
4662     \tl_set:Nn \l_stex_current_args_tl{ #3 }
4663     \tl_set:Nn \l_stex_current_return_tl{ #4 }
4664     \tl_set:Nn \l_stex_current_type_tl{ #5 }
4665     \tl_clear:N \l_stex_current_term_tl
4666   }
4667   \tl_put_right:N \l_stex_current_redo_tl \l_stex_every_symbol_tl
4668   \tl_put_left:Ne \l_stex_current_redo_tl {
4669     \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl}
4670   }
4671   \l_stex_current_redo_tl
4672 }

```

(End of definition for \stex_invoke_symbol:NnnnnnN and \stex_invoke_symbol:nnnnnnN. These functions are documented on page 149.)

sfunction

\stex_invoke_symbol: Math or text mode?

```

4673 \cs_new_protected:Nn \stex_invoke_symbol: {
4674   \stex_debug:nn{expressions}{Invoking~ \l_stex_current_full_tl}
4675   \mode_if_math:TF \__stex_expr_math: \__stex_expr_text:
4676 }

```

(End of definition for \stex_invoke_symbol:. This function is documented on page 149.)

sfunction

\stex_invoke_text_symbol:

```

4677 \cs_new_protected:Nn \stex_invoke_text_symbol: {
4678   \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
4679   \stex_term_oms_or_omv:nn{}{\maincomp{\let\xspace\relax\l_stex_current_return_tl}}
4680   \group_end:\mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
4681 }

```

(End of definition for \stex_invoke_text_symbol:. This function is documented on page 149.)

sfunction

\stex_invoke_sequence:

```

4682 \cs_new_protected:Nn \stex_invoke_sequence: {
4683   \peek_charcode_remove:NTF ! {
4684     \peek_charcode:NTF [ \__stex_expr_seq_op:w { \__stex_expr_seq_op:w [] }
4685   } \__stex_expr_seq_first:
4686 }
4687
4688 \cs_new_protected:Npn \__stex_expr_seq_op:w [#1] {
4689   \stex_use_op_notation:nnTF \l_stex_current_full_tl{#1}{
4690     %\stex_debug:nn{vars}{Here: \meaning\l_stex_notation_cs}
4691     \stex_maybe_brackets:nn{\neginfprec}{

```

```

4692     \_stex_term_oms_or_omv:nn{#1}
4693     {\l_stex_notation_cs}\group_end:
4694 }
4695 }{
4696     \__stex_expr_get_index_notation:n{#1}
4697     \peek_charcode:NTF [ \__stex_expr_range:w { \__stex_expr_range:w[] }
4698 }
4699 }
4700
4701 \cs_new_protected:Nn \__stex_expr_get_index_notation:n {
4702     \stex_use_notation:nnTF \l_stex_current_full_tl{#1}{}{
4703         \stex_do_default_notation:
4704         \cs_set_eq:NN \l_stex_notation_cs \l_stex_default_notation
4705     }
4706 }
4707
4708 \cs_new_protected:Npn \__stex_expr_range:w [#1] {
4709     \bool_set_true:N \l_stex_allow_semantic_bool
4710     \clist_clear:N \l__stex_expr_clist
4711     \clist_map_function:NN \l_stex_current_type_tl \__stex_expr_seq_arg:n
4712     \stex_annotate:nn{
4713         data-ftml-term=OMV,
4714         data-ftml-head={\l_stex_current_full_tl},
4715         data-ftml-notationid={}
4716     }{
4717         \l__stex_expr_clist
4718     }
4719     \group_end:
4720 }
4721
4722 \cs_new_protected:Nn \__stex_expr_seq_arg:n {
4723     \tl_if_eq:nnTF{#1}{\ellipses}{
4724         \clist_put_right:Nn \l__stex_expr_clist {
4725             \ellipses
4726         }
4727     }{
4728         \clist_put_right:Nn \l__stex_expr_clist {
4729             \exp_args:No \str_if_eq:nnTF \l_stex_current_arity_str {1}{
4730                 \group_begin:
4731                 \l_stex_notation_cs \group_end: {#1}
4732             }{
4733                 \group_begin:
4734                 \l_stex_notation_cs \group_end: #1
4735             }
4736         }
4737     }
4738 }
4739 }
4740
4741 \cs_new_protected:Nn \__stex_expr_seq_first: {
4742     \exp_args:Nne \use:nn{
4743         \cs_generate_from_arg_count:NNnn \l_stex_notation_cs \cs_set:Npn
4744         \l_stex_current_arity_str {
4745             \tl_set:Nn \exp_not:N \l__stex_expr_first_args_tl {

```

```

4746     \int_step_function:nN \l_stex_current_arity_str \__stex_expr_do_first_arg:n
4747   }
4748   \exp_not:N \__stex_expr_do_first_next:
4749 }
4750 \l_stex_notation_cs
4751 }
4752
4753 \cs_new:Nn \__stex_expr_do_first_arg:n {{\exp_not:n{## #1}}}
4754
4755 \cs_new_protected:Nn \__stex_expr_do_first_next: {
4756   \peek_charcode_remove:NTF ! {
4757     \peek_charcode:NTF [ \__stex_expr_do_one:w {\__stex_expr_do_one:w []}
4758   }{
4759     \peek_charcode:NTF [ \__stex_expr_do_all:w {\__stex_expr_do_all:w []}
4760   }
4761 }
4762
4763 \cs_new_protected:Npn \__stex_expr_do_one:w [#1] {
4764   \__stex_expr_get_index_notation:n{#1}
4765   \exp_args:Nno\use:nn{\l_stex_notation_cs\group_end:}\l__stex_expr_first_args_tl
4766 }
4767
4768 \cs_new_protected:Npn \__stex_expr_do_all:w [#1] {
4769   \exp_args:Nno\use:nn{\__stex_expr_notation:w [#1]}\l__stex_expr_first_args_tl
4770 }

```

(End of definition for `\stex_invoke_sequence:..` This function is documented on page 150.)

sfunction

`\stex_invoke_structure:`

```

4771 \cs_new_protected:Nn \stex_invoke_structure: {
4772   \tl_set:Nn \l__stex_expr_set_comp_tl {\__stex_expr_set_thiscomp:}
4773   \__stex_expr_struct_top:n {}
4774 }
4775
4776 \cs_new_protected:Nn \__stex_expr_set_thiscomp: {
4777   \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_thiscomp {
4778     \edef\maincomp {\_thiscomp{\comp}}
4779   }
4780 }
4781
4782 \cs_new_protected:Nn \__stex_expr_struct_top:n {
4783   \stex_debug:nn{structure}{
4784     invoking~structure~{\l_stex_current_type_tl}<\tl_to_str:n{#1}>
4785   }
4786   \peek_charcode:NTF [ {
4787     \__stex_expr_merge:nw{#1}
4788   }{
4789     \__stex_expr_struct_type:n {#1}
4790     \tl_set:Nn \l_stex_struct_this_tl {}
4791     \peek_charcode_remove:NTF ! {
4792       \peek_charcode:NTF [ {
4793         \__stex_expr_maybe_notation:w
4794       }{

```

```

4795     \_stex_expr_maybe_notation:w []
4796   }
4797   }{
4798     \_stex_expr_invoke_this:n
4799   }
4800 }
4801 }
4802
4803 \cs_new_protected:Npn \_stex_expr_merge:nw #1 [ #2 ] {
4804   \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{comp=}{
4805     \_stex_expr_set_customcomp: #2 \_stex_expr_end:
4806     \_stex_expr_struct_top:n{#1}
4807   }{
4808     \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{this=}{
4809       \_stex_expr_set_thisnotation: #2 \_stex_expr_end:
4810       \_stex_expr_struct_top:n{#1}
4811     }{
4812       \tl_if_empty:nTF{#1}{
4813         \_stex_expr_struct_top:n{#2}
4814       }{
4815         \tl_if_empty:nTF{#2}{
4816           \_stex_expr_struct_top:n{#1}
4817         }{
4818           \_stex_expr_struct_top:n{#1,#2}
4819         }
4820       }
4821     }
4822   }
4823 }
4824
4825 \cs_new_protected:Npn \_stex_expr_set_thisnotation: this= #1 \_stex_expr_end: {
4826   \tl_set:Nn \l_stex_return_notation_tl { \comp{#1} }
4827   \tl_set:Nn \l__stex_expr_set_comp_tl {}
4828 }
4829
4830 \cs_new_protected:Npn \_stex_expr_set_customcomp: comp= #1 \_stex_expr_end: {
4831   \tl_set:Nn \l__stex_expr_set_comp_tl {
4832     \_stex_expr_set_custom_comp:n{#1}
4833   }
4834   \tl_set:Nn \l_stex_return_notation_tl { \comp{} }
4835 }

```

The structure type:

```

4836 \cs_new_protected:Nn \_stex_expr_struct_type:n {
4837   \_stex_expr_do_assign_list:n{#1}
4838   \clist_if_empty:NTF \l__stex_expr_fields_clist {
4839     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
4840       = 1 {
4841       \tl_set:Ne \l_stex_expr_current_type_tl {
4842         \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl }
4843         \exp_args:No \exp_not:n \l_stex_current_redo_tl
4844         \_stex_term_oms_or_omv:nn{}{}
4845       }
4846     }{
4847       \exp_args:No \_stex_expr_make_type:n \l_stex_current_type_tl

```

```

4848   }
4849   ){
4850     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
4851       = 1 {
4852         \__stex_expr_make_type:n {}
4853       }{
4854         \exp_args:No \__stex_expr_make_type:n \l_stex_current_type_tl
4855       }
4856   }
4857 }
4858
4859 \cs_new_protected:Nn \__stex_expr_do_assign_list:n {
4860   \clist_clear:N \l__stex_expr_fields_clist
4861   \tl_if_empty:NF {#1} {
4862     \keyval_parse:NNn\TODO\__stex_expr_do_assign:nn{#1}
4863   }
4864 }
4865
4866 \cs_new_protected:Nn \__stex_expr_do_assign:nn {
4867   \clist_put_right:Nn \l__stex_expr_fields_clist {{#1}{#2}}
4868 }
4869
4870 \cs_new_protected:Nn \__stex_expr_make_type:n {
4871   \tl_if_empty:NTF{#1}{
4872     \seq_clear:N \l_tmpa_seq
4873   }{
4874     \seq_set_split:Nnn \l_tmpa_seq ,{#1}
4875     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
4876     \seq_reverse:N \l_tmpa_seq
4877   }
4878   \tl_set:Ne \l__stex_expr_current_type_tl {
4879     \symuse{Metatheory?module~type~merge}{
4880       {
4881         \exp_args:No \exp_not:n \l_stex_current_redo_tl
4882         \stex_term_oms_or_omv:nn{}{}}
4883     }
4884     \seq_map_function:NN \l_tmpa_seq \__stex_expr_make_mod:n
4885     \clist_if_empty:NF \l__stex_expr_fields_clist {
4886       ,\symuse{Metatheory?anonymous~record}{
4887         \exp_args:Ne \tl_tail:n{
4888           \clist_map_function:NN \l__stex_expr_fields_clist \__stex_expr_make_oml:n
4889         }
4890       }
4891     }
4892   }
4893 }
4894 }
4895
4896 \cs_new:Nn \__stex_expr_make_mod:n {
4897   ,\symuse{Metatheory?module~type}{
4898     \exp_args:Ne \stex_annotate:nn{data-ftml-term=OMMOD,data-ftml-head={\stex_use_module_ur
4899   }
4900 }
4901

```

```

4902
4903 \cs_new:Nn \__stex_expr_make_oml:n {
4904   \__stex_expr_make_oml:nn #1
4905 }
4906 \cs_new:Nn \__stex_expr_make_oml:nn {
4907   ,\stex_annotate:nn{
4908     data-ftml-term=OML,
4909     data-ftml-head={#1}
4910   }{
4911     \stex_annotate_force_break:n{
4912       \stex_annotate:nn{data-ftml-definiens={}}{\exp_not:n{#2!}}
4913     }
4914   }
4915 }

```

Insert the structure type as a term:

```

4916 \cs_new:Nn \__stex_expr_current_type: {
4917   %\exp_args:No \exp_not:n \l_stex_current_redo_tl
4918   \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
4919     \exp_args:No\exp_not:n\l__stex_expr_current_type_tl
4920   }
4921   \stex_term_oms_or_omv:nn{}{}
4922 }

```

The structure type itself:

```

4923 \cs_new_protected:Npn \__stex_expr_maybe_notation:w [ #1 ] {
4924   \tl_set_eq:NN \l_stex_current_term_tl \l__stex_expr_current_type_tl
4925   \stex_use_notation:nnTF \l_stex_current_full_tl {#1}{
4926     \l_stex_notation_cs\group_end:
4927   }{
4928     \__stex_expr_make_prop:
4929     \__stex_expr_make_prop_assign:
4930     \__stex_expr_present_i:w [ #1 ]
4931   }
4932 }
4933
4934 \cs_new_protected:Nn \__stex_expr_present: {
4935   \peek_charcode:NTF [ {
4936     \__stex_expr_present_i:w
4937   }{
4938     \__stex_expr_present:nn{}{}
4939   }
4940 }
4941
4942 \cs_new_protected:Npn \__stex_expr_present_i:w [ #1 ] {
4943   \int_compare:nNnTF{\clist_count:n{#1}} = 1 {
4944     \__stex_expr_present:nn{}{#1}
4945   }{
4946     \peek_charcode:NTF [ {
4947       \__stex_expr_present_ii:nw{#1}
4948     }{
4949       \__stex_expr_present:nn{#1}{}
4950     }
4951   }
4952 }

```

```

4953
4954 %First: clist, second:notation-id
4955 \cs_new_protected:Npn \__stex_expr_present_ii:nw #1 [#2] {
4956   \__stex_expr_present:nn{#1}{#2}
4957 }
4958
4959 \cs_new_protected:Nn \__stex_expr_present:nn {
4960   \clist_clear:N \l__stex_expr_clist
4961   \tl_if_empty:nTF{#1}{
4962     \cs_set:Npn \l__stex_expr_cs ##1 ##2 ##3 {
4963       \tl_if_empty:nF{##2}{
4964         \__stex_expr_present_entry:nn {##1}{##3}
4965       }
4966     }
4967   }{
4968     \cs_set:Npn \l__stex_expr_cs ##1 ##2 ##3 {
4969       \exp_args:Ne \clist_if_in:nnT{\tl_to_str:n{#1}}{##1}{
4970         \__stex_expr_present_entry:nn {##1}{##3}
4971       }
4972     }
4973   }
4974   \prop_map_inline:Nn \l__stex_expr_prop {
4975     \l__stex_expr_cs {##1} ##2
4976   }
4977   \stex_term_oms_or_omv:nn{}{
4978     \exp_args:Nno \use:n{
4979       \bool_set_true:N \l_stex_allow_semantic_bool
4980       \symuse{Metatheory?mathematical~structure}[#2]
4981     }{\l__stex_expr_clist}
4982   }\group_end:
4983 }
4984
4985 \cs_new_protected:Nn \__stex_expr_present_entry:nn {
4986   \seq_if_in:NnTF \l__stex_expr_assigned_seq {#1}{
4987     \clist_put_right:Nn \l__stex_expr_clist {#2!}
4988   }{
4989     \exp_args:Nne \clist_put_right:Nn \l__stex_expr_clist {
4990       \stex_next_symbol:n {
4991         \exp_args:No \exp_not:n \l__stex_expr_set_comp_tl
4992         \tl_set:Nn \exp_not:N \l_stex_struct_this_tl {
4993           \exp_args:No \exp_not:n \l_stex_struct_this_tl
4994         }
4995         \exp_not:n {
4996           \tl_set_eq:NN \this \l_stex_struct_this_tl
4997         }
4998         \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
4999           \exp_args:No \exp_not:n \l_stex_return_notation_tl
5000         }
5001       }
5002     }
5003   }
5004 }
5005 }
5006

```



```

5007 %\cs_new_protected:Nn \__stex_expr_set_custom_comp:n {
5008 % \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
5009 % \cs_set_protected:Npx \_customthiscomp ##1 {
5010 % \group_begin:
5011 % \bool_set_true:N \l_stex_allow_semantic_bool
5012 % \exp_not:n{
5013 % \cs_set:Npn \l__stex_expr_comp_cs ##1 {
5014 % #1
5015 % }
5016 % \def\maincomp
5017 % }{\comp}
5018 % \exp_not:N\l__stex_expr_comp_cs{\comp{##1}}
5019 % \group_end:
5020 % }
5021 % \def\maincomp {\_customthiscomp}
5022 % }
5023 %}

5024
5025 \cs_new_protected:Nn \__stex_expr_set_custom_comp:n {
5026 \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
5027 \stex_debug:nn{structs}{Setting~custom~comp:~\tl_to_str:n{#1}}
5028 \exp_args:Nne \__stex_expr_set_custom_i:nn{#1}{\comp}
5029 \def\maincomp {\_customthiscomp}
5030 }
5031 }

5032
5033 \cs_new_protected:Nn \__stex_expr_set_custom_i:nn {
5034 \cs_set_protected:Npn \_customthiscomp ##1 {
5035 \group_begin:
5036 \bool_set_true:N \l_stex_allow_semantic_bool
5037 \cs_set:Npn \l__stex_expr_comp_cs #####1 { #1 }
5038 \def \maincomp {#2}
5039 \l__stex_expr_comp_cs{#2{##1}}
5040 \group_end:
5041 }
5042 }

this (of type structure):
5043 \cs_new_protected:Nn \__stex_expr_invoke_this:n {
5044 \peek_charcode_remove:NTF ! {
5045 \exp_args:Nne\use:nn{
5046 \group_end:\symuse{Metatheory?of~type}[invisible]{
5047 \tl_if_empty:nTF{#1}{\_stex_annotate_force_break:n{}}{#1}
5048 }
5049 }{
5050 {
5051 \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl}
5052 \__stex_expr_current_type:
5053 }
5054 }
5055 }{
5056 \__stex_expr_invoke_maybe_field:nn{#1}
5057 }
5058 }
5059

```

```

5060 \cs_new_protected:Nn \__stex_expr_invoke_maybe_field:nn {
5061   \__stex_expr_make_prop:
5062   \__stex_expr_set_this:n{#1}
5063   \tl_if_empty:nTF{#2}{
5064     \__stex_expr_make_prop_assign:
5065     \__stex_expr_present:
5066   }{
5067     \__stex_expr_invoke_field:n{#2}
5068   }
5069 }
5070
5071 \cs_new_protected:Nn \__stex_expr_set_this:n {
5072   \tl_if_empty:nTF{#1}{
5073     %\tl_put_right:Nn \l_stex_current_redo_tl {
5074       % \tl_clear:N \l_stex_struct_this_tl
5075     %}
5076   }{
5077     \tl_set:Ne \l_stex_struct_this_tl {{
5078       \bool_set_true:N \l_stex_allow_semantic_bool
5079       \bool_set_true:N \l_stex_in_this_bool
5080       \tl_set:Nn \exp_not:N \this {
5081         \exp_args:No \exp_not:n \this
5082       }
5083       \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl }
5084       \exp_not:n{#1}
5085     }}
5086     \tl_set_eq:NN \this \l_stex_struct_this_tl
5087     % \l_stex_return_notation_tl
5088   }
5089 }
5090
5091 \cs_new_protected:Nn \__stex_expr_get_field_name:n {
5092   \str_set:Ne \l_stex_expr_field_name_str {
5093     \exp_args:Nne \use:n {\exp_after:wN \use_i:nn \use:n}
5094     {\prop_item:Nn \l__stex_expr_prop {#1}}
5095   }
5096   \str_if_empty:NT \l_stex_expr_field_name_str {
5097     \str_set:Nn \l_stex_expr_field_name_str {#1}
5098   }
5099 }
5100
5101 \cs_new_protected:Nn \__stex_expr_invoke_field:n {
5102   \prop_if_in:NnTF \l__stex_expr_prop {#1}{
5103     \__stex_expr_get_field_name:n{#1}
5104     \tl_clear:N \l__stex_expr_more_nextsymbol_tl
5105     %\exp_args:Nne \seq_if_in:NnF \l__stex_expr_assigned_seq {\tl_to_str:n{#1}}{
5106       \tl_set:Ne \l__stex_expr_more_nextsymbol_tl {
5107         \tl_set:Nn \exp_not:N \l_stex_struct_this_tl {
5108           \exp_args:No \exp_not:n \l_stex_struct_this_tl
5109         }
5110         \exp_not:n {
5111           \tl_set_eq:NN \this \l_stex_struct_this_tl
5112         }
5113         \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {

```

```

5114         \exp_args:No \exp_not:n \l_stex_return_notation_tl
5115     }
5116     \exp_args:No \exp_not:n \l__stex_expr_set_comp_tl
5117 }
5118 %}
5119 \exp_args:NNe \use:nn \group_end: {
5120     \_stex_next_symbol:n {
5121         \exp_args:No \exp_not:n \l__stex_expr_redo_tl
5122         \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5123             \symuse{Metatheory?record-field}{
5124                 \symuse{Metatheory?of~type}{
5125                     \exp_args:No\tl_if_empty:nTF \l_stex_struct_this_tl {\_stex_annotate_force_br
5126                 }{
5127                     \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl}
5128                     \__stex_expr_current_type:
5129                 }
5130             }{
5131                 \stex_annotate:nn{data-ftml-term=OML,data-ftml-head={\l__stex_expr_field_name_s
5132             }
5133         }
5134         \exp_args:No \exp_not:n \l__stex_expr_more_nextsymbol_tl
5135     }
5136     \exp_not:N \use_ii:nn
5137     \prop_item:Nn \l__stex_expr_prop {#1}
5138 }
5139 }{
5140     \msg_error:nnn{stex}{error/unknownfield}{#1}
5141 }
5142 }
5143
5144 \cs_new_protected:Nn \__stex_expr_make_prop: {
5145     \prop_clear:N \l__stex_expr_prop
5146     \seq_clear:N \l__stex_expr_seq
5147     \seq_clear:N \l__stex_expr_assigned_seq
5148     \tl_clear:N \l__stex_expr_redo_tl
5149     \__stex_expr_prop_do_decls:
5150 }
5151
5152 \cs_new_protected:Nn \__stex_expr_prop_do_decls: {
5153     \group_begin:
5154     \stex_debug:nn{expressions}{constructing-record}
5155     \seq_clear:N \l_stex_all_module_seq
5156     \exp_args:Ne \stex_activate_module:n {\clist_item:Nn \l_stex_current_type_tl 1}
5157     \exp_args:No \stex_iterate_symbols:nn \l_stex_current_type_tl {
5158         % uri, macro name, name, arity, args, type, def, return, macro
5159         \tl_if_empty:nTF{##2}{
5160             \exp_args:Nne\__stex_expr_do_decl_nomacro:nnnnnnnn{##3}
5161         }{
5162             \exp_args:Nne\__stex_expr_do_decl:nnnnnnnn{##2}
5163         }
5164         {\stex_new_symbol_uri:nn{##1}{##3}}{##3}{##4}{##5}{##6}{##7}{##8}##9
5165     }
5166     \exp_after:wN \tl_set:Nn \exp_after:wN \l__stex_expr_after_tl \exp_after:wN {
5167         \exp_after:wN \tl_set:Nn \exp_after:wN \l__stex_expr_prop \exp_after:wN { \l__stex_ex

```

```

5168     }
5169     \__stex_expr_prop_do_notations:
5170     \stex_debug:nn{expressions}{record:~\meaning\l__stex_expr_after_tl}
5171     \tl_put_right:Ne \l__stex_expr_after_tl {\tl_set:Nn \exp_not:N \l__stex_expr_redo_tl {\
5172     \exp_after:wN \group_end: \l__stex_expr_after_tl
5173   }
5174
5175   % id, uri, name, arity, args, type, def, return, macro
5176   \cs_new_protected:Nn \__stex_expr_do_decl_nomacro:nnnnnnnnn {
5177     \prop_if_in:NnF \l__stex_expr_prop {#1} {
5178       \seq_put_left:Ne \l__stex_expr_seq {\tl_to_str:n{#2}}
5179       \prop_put:Nnn \l__stex_expr_prop {#1}{
5180         }{
5181           \stex_invoke_symbol:nnnnnnN
5182           {#2}
5183           {#4}
5184           {#5}{#6}{#7}{#8}#9
5185         }
5186       }
5187     }
5188   }
5189
5190   \cs_new_protected:Nn \__stex_expr_do_decl:nnnnnnnnn {
5191     \prop_if_in:NnF \l__stex_expr_prop {#1} {
5192       \seq_put_left:Ne \l__stex_expr_seq {\tl_to_str:n{#2}}
5193       \prop_put:Nnn \l__stex_expr_prop {#1}{
5194         {#3}{
5195           \stex_invoke_symbol:nnnnnnN
5196           {#2}
5197           {#4}
5198           {#5}{#6}{#7}{#8}#9
5199         }
5200       }
5201     }
5202   }
5203
5204   \cs_new_protected:Nn \__stex_expr_make_prop_assign: {
5205     \clist_if_empty:NF \l__stex_expr_fields_clist {
5206       \clist_map_inline:Nn \l__stex_expr_fields_clist {
5207         \__stex_expr_make_prop_assign:nn ##1
5208       }
5209     }
5210   }
5211
5212   \cs_new_protected:Nn \__stex_expr_make_prop_assign:nn {
5213     \prop_if_in:NnTF \l__stex_expr_prop {#1}{
5214       \exp_args:NNe \seq_put_right:Nn \l__stex_expr_assigned_seq {\tl_to_str:n{#1}}
5215       \exp_args:Nne \use:nn {\__stex_expr_make_prop_assign_replace:nnnn {#1}{#2}}
5216       {\prop_item:Nn \l__stex_expr_prop {#1}}
5217     }{
5218       \msg_error:nnn{stex}{error/unknownfieldass}{#1}
5219     }
5220   }
5221   \cs_new_protected:Nn \__stex_expr_make_prop_assign_replace:nnnn {

```

```

5222 \prop_put:Nnn \l__stex_expr_prop {#1}{#{3}{#2}}
5223 \tl_if_empty:NF{#3}{
5224   \tl_set:cn{#1}{ #2 }
5225   \tl_put_right:Nn \l__stex_expr_redo_tl {
5226     \tl_set:cn{#1}{ #2 }
5227   }
5228 }
5229 }
5230
5231 \cs_new_protected:Nn \__stex_expr_prop_do_notations: {
5232   \exp_args:No \stex_iterate_notations:nn \l_stex_current_type_tl {
5233     \exp_args:NNe \seq_if_in:NnT \l__stex_expr_seq {\tl_to_str:n{##1}}{
5234       \tl_set:Nn \l_tmpa_tl {
5235         \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}?##2_cs}{
5236           \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}?##2_cs}{##4}
5237         }
5238         \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}?_cs}{
5239           \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}?_cs}{##4}
5240         }
5241       }
5242       \exp_args:NNo \tl_put_right:Nn \l__stex_expr_redo_tl \l_tmpa_tl
5243       \exp_args:NNo \tl_put_right:Nn \l__stex_expr_after_tl \l_tmpa_tl
5244       \tl_if_empty:NF{##5}{
5245         \tl_set:Nn \l_tmpa_tl {
5246           \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}?##2_op_cs}{
5247             \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}?##2_op_cs}{##5}
5248           }
5249           \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}?_op_cs}{
5250             \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}?_op_cs}{##5}
5251           }
5252         }
5253         \exp_args:NNo \tl_put_right:Nn \l__stex_expr_redo_tl \l_tmpa_tl
5254         \exp_args:NNo \tl_put_right:Nn \l__stex_expr_after_tl \l_tmpa_tl
5255       }
5256     }
5257   }
5258 }

```

(End of definition for `\stex_invoke_structure:`. This function is documented on page 150.)

sfunction

`_stex_invoke_variable:nnnnnn`

```

5259 \cs_new_protected:Nn \_stex_invoke_variable:nnnnnnN {
5260   \bool_if:NTF \l_stex_allow_semantic_bool{
5261     \group_begin:
5262     \tl_set:Nn \l_stex_current_full_tl { #1 }
5263     \tl_set:Nn \l_stex_current_display_tl { #1 }
5264     \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
5265     \__stex_expr_setup:nnnnn{\_varcomp}{#2}{#3}{#6}{#5}
5266     \cs_set_eq:NN \_stex_term_oms_or_omv:nn \_stex_term_omv:nn
5267     \tl_put_right:Nn \l_stex_current_redo_tl {
5268       \cs_set_eq:NN \_stex_term_oms_or_omv:nn \_stex_term_omv:nn
5269     }
5270     #7

```

```

5271   }{
5272   \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_full_tl}
5273   }
5274 }

```

(End of definition for `_stex_invoke_variable:nnnnnn`. This function is documented on page 150.)

sfunction

`\svar`

```

5275 \NewDocumentCommand \svar {0{ } m}{
5276   \group_begin:
5277   \tl_if_empty:nTF{#1}{
5278     \tl_set:Nn \l_stex_current_full_tl { #2 }
5279     \tl_set:Nn \l_stex_current_display_tl { #2 }
5280   }{
5281     \tl_set:Nn \l_stex_current_full_tl { #1 }
5282     \tl_set:Nn \l_stex_current_display_tl { #1 }
5283   }
5284   \bool_if:NTF \l_stex_allow_semantic_bool{
5285     \tl_clear:N \l_stex_current_term_tl
5286     \stex_term_omv:nn{}{\_varcomp{#2}}
5287   }{
5288     \msg_error:nnxx{stex}{error/notallowed}{Variable}{\l_stex_current_full_tl}
5289   }
5290   \group_end:
5291 }

```

(End of definition for `\svar`. This function is documented on page 150.)

sfunction

Now for the brunt of the work:

Math

Modifiers (! or *)?

```

5292 \cs_new_protected:Nn \_stex_expr_math: {
5293   \stex_debug:nn{expressions}{math~mode}
5294   \peek_charcode_remove:NTF ! {
5295     % operator
5296     \peek_charcode_remove:NTF * \_stex_expr_op_custom:n {
5297       % op notation
5298       \peek_charcode:NTF [ \_stex_expr_op_notation:w {
5299         \_stex_expr_op_notation:w []
5300       }
5301     }
5302   }{
5303     \peek_charcode_remove:NTF * \_stex_expr_custom:n {
5304       % normal
5305       \peek_charcode:NTF [ \_stex_expr_notation:w {
5306         \_stex_expr_notation:w []
5307       }
5308     }
5309   }
5310 }

```

Text

Operator?

```
5311 \cs_new_protected:Nn \__stex_expr_text: {  
5312   \stex_debug:nn{expressions}{text~mode}  
5313   \peek_charcode_remove:NTF ! \__stex_expr_op_custom:n \__stex_expr_custom:n  
5314 }
```

Notation

Operator?

```
5315 \cs_new_protected:Npn \__stex_expr_notation:w [ #1 ] {  
5316   \peek_charcode_remove:NTF ! {  
5317     \__stex_expr_op_notation:w [ #1]  
5318   }{  
5319     \__stex_expr_full_notation:n { #1}  
5320   }  
5321 }
```

(Possibly) complex notation taking arguments with optional particular identifier:

```
5322 \cs_new_protected:Nn \__stex_expr_full_notation:n {  
5323   \stex_use_notation:nnTF \l_stex_current_full_tl { #1 } {  
5324     \stex_debug:nn{expressions}{using~notation~" #1 " ~for~\l_stex_current_full_tl}  
5325     \tl_if_empty:NTF \l_stex_current_return_tl {  
5326       \stex_debug:nn{expressions}{return~empty}  
5327       \l_stex_notation_cs{\group_end:\stex_eat_exclamation_point:}  
5328     }{  
5329       \stex_debug:nn{expressions}{return?}  
5330       \exp_after:wN \__stex_expr_maybe_return:n \exp_after:wN {  
5331         \l_stex_notation_cs{  
5332       }  
5333     }  
5334   }{  
5335     \stex_debug:nn{expressions}{notation~" #1 " ~for~\l_stex_current_full_tl{ }~does~not~exist;  
5336     \stex_do_default_notation:  
5337     \tl_if_empty:NTF \l_stex_current_return_tl {  
5338       \l_stex_default_notation{\group_end:\stex_eat_exclamation_point:}  
5339     }{  
5340       \exp_after:wN  
5341       \__stex_expr_maybe_return:n  
5342       \exp_after:wN  
5343       {\l_stex_default_notation { } }  
5344     }  
5345   }  
5346 }
```

Operator notation:

```
5347 \cs_new_protected:Npn \__stex_expr_op_notation:w [ #1 ] {  
5348   \stex_debug:nn{expressions}{op~notation~for~\l_stex_current_full_tl}  
5349   \stex_use_op_notation:nnTF \l_stex_current_full_tl { #1 } {  
5350     \stex_maybe_brackets:nn{\neginfprec}{  
5351       \stex_term_oms_or_omv:nn{ #1 }  
5352       {\l_stex_notation_cs}  
5353     }  
5354     \group_end:  
5355   }{
```

```

5356 \int_compare:nNnTF \l_stex_current_arity_str = 0 {
5357   \tl_clear:N \l_stex_current_return_tl
5358   \__stex_expr_notation:w [#1]
5359 }{
5360   \stex_do_default_notation_op:
5361   \_stex_maybe_brackets:nn{\neginfprec}{
5362     \_stex_term_oms_or_omv:nn{#1}
5363     {\l_stex_default_notation}
5364   }
5365   \group_end:
5366 }
5367 }
5368 }

```

Custom Notations

Operator:

```

5369 \cs_new_protected:Nn \__stex_expr_op_custom:n {
5370   \stex_debug:nn{expressions}{custom-op}
5371   \bool_set_true:N \l_stex_allow_semantic_bool
5372   \_stex_term_oms_or_omv:nn{}{\maincomp{#1}}
5373   \group_end:
5374 }

```

Complex:

```

5375 \int_new:N \l__stex_expr_arg_counter_int
5376
5377 \cs_new_protected:Nn \__stex_expr_custom:n {
5378   \stex_debug:nn{custom}{custom-notation-for-\l_stex_current_full_tl}
5379   \stex_pseudogroup:nn{
5380     \bool_set_true:N \l_stex_allow_semantic_bool
5381     \prop_gclear:N \l__stex_expr_customs_prop
5382     \seq_gclear:N \l__stex_expr_customs_seq
5383     \int_gzero:N \l__stex_expr_arg_counter_int
5384     \tl_if_empty:NF \l_stex_current_args_tl {
5385       \exp_after:wN \__stex_expr_add_prop_arg:nnw \l_stex_current_args_tl \_stex_args_end:
5386       \cs_set_eq:NN \arg \__stex_expr_arg:n
5387     }
5388     \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
5389     \cs_set_eq:NN \__stex_expr_do_ab_next:nn \_stex_term_oma:nn
5390     \_stex_map_args:N \__stex_expr_check_b:nn
5391     \__stex_expr_do_ab_next:nn{}{#1}
5392   }{
5393     \prop_if_exist:NT \l__stex_expr_customs_prop {
5394       \prop_gset_from_keyval:Nn \exp_not:N \l__stex_expr_customs_prop {
5395         \prop_to_keyval:N \l__stex_expr_customs_prop
5396       }
5397     }
5398     \int_gset:Nn \l__stex_expr_arg_counter_int { \int_use:N \l__stex_expr_arg_counter_int}
5399     \seq_if_exist:NT \l__stex_expr_customs_seq {
5400       \seq_gset_split:Nnn \exp_not:N \l__stex_expr_customs_seq , {
5401         \seq_use:Nn \l__stex_expr_customs_seq ,
5402       }
5403     }
5404   }

```



```

5405 % TODO check that all arguments are present
5406 \group_end:
5407 }
5408
5409 \cs_new_protected:Npn \__stex_expr_add_prop_arg:nnw #1 #2 #3\stex_args_end: {
5410   \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {}
5411   \seq_gput_right:Nn \l__stex_expr_customs_seq {#2}
5412   \tl_if_empty:nF{#3}{\__stex_expr_add_prop_arg:nnw #3 \stex_args_end:}
5413 }
5414
5415 \cs_new:Nn \__stex_expr_check_b:nn {
5416   \str_case:nn #2 {
5417     b {\cs_set_eq:NN \__stex_expr_do_ab_next:nn \stex_term_omb:nn}
5418     B {\cs_set_eq:NN \__stex_expr_do_ab_next:nn \stex_term_omb:nn}
5419   }
5420 }
5421
5422 Arguments:
5423 \NewDocumentCommand \__stex_expr_arg:n {s O{} m} {
5424   \IfBooleanTF #1 {
5425     \stex_annotate_invisible:n{
5426       \__stex_expr_arg_inner:nn{#2}{#3}
5427     }
5428   }{
5429     \__stex_expr_arg_inner:nn{#2}{#3}
5430   }
5431 }
5432
5433 \cs_new_protected:Nn \__stex_expr_arg_inner:nn {
5434   \tl_if_empty:nTF{#1}{
5435     \int_gincr:N \l__stex_expr_arg_counter_int
5436     \exp_args:Ne \__stex_expr_check:nTF{ \int_use:N \l__stex_expr_arg_counter_int }{
5437       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl
5438     }{
5439       \__stex_expr_arg_inner:nn{}
5440     }{ #2 }
5441   }{
5442     \__stex_expr_check:nTF {#1}{
5443       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5444     }{
5445       \exp_args:No \str_case:nnTF \l_tmpb_tl {
5446         {a}{
5447           \exp_args:NNne \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
5448             \l_tmpa_tl X
5449           }
5450           \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5451         }
5452         {B}{
5453           \exp_args:NNne \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
5454             \l_tmpa_tl X
5455           }
5456           \tl_set:Ne \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5457         }
5458       }
5459     }{
5460       \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5461     }
5462   }

```

```

5458     }{
5459         \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_full_tl}
5460     }
5461 }
5462 }
5463 }
5464
5465 \prg_new_conditional:Nnn \__stex_expr_check:n {TF} {
5466     \exp_args:NNe \prop_get:NnNTF \l__stex_expr_customs_prop {#1} \l_tmpa_tl {
5467         \tl_set:Ne \l_tmpb_tl {\seq_item:Nn \l__stex_expr_customs_seq {#1} }
5468         \tl_if_empty:NTF \l_tmpa_tl {
5469             \exp_args:NNe \prop_gput:Nnn \l__stex_expr_customs_prop
5470                 { #1 }{X}
5471             \exp_args:No \str_case:nnF \l_tmpb_tl {
5472                 {a}{
5473                     \tl_set:Ne \l_tmpa_tl{ #1 1 }
5474                 }
5475                 {B}{
5476                     \tl_set:Ne \l_tmpa_tl{ #1 1 }
5477                 }
5478             }{
5479                 \tl_set:Ne \l_tmpa_tl{ #1 }
5480             }
5481             \prg_return_true:
5482         }{
5483             \prg_return_false:
5484         }
5485     }{
5486         \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_full_tl}
5487         \prg_return_false:
5488     }
5489 }
5490
5491 % #1 argnum #2 argmode #3 code
5492 \cs_new_protected:Nn \__stex_expr_arg_do:nnn {
5493     \stex_debug:nn{custom}{Doing~argument~#1~of~mode~#2:~\tl_to_str:n{#3}}
5494     \group_begin:
5495         \bool_set_true:N \l_stex_allow_semantic_bool
5496         \stex_term_arg:nnn {#2}{#1}{#3}
5497     \group_end:
5498 }
5499 \cs_generate_variant:Nn \__stex_expr_arg_do:nnn {oon}

```

Return code

```

5500 \cs_new_protected:Nn \__stex_expr_maybe_return:n {
5501     \tl_set:Nn \l__stex_expr_return_this_tl {#1}
5502     \tl_clear:N \l__stex_expr_return_args_tl
5503     \exp_args:Nne \use:n {
5504         \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs:
5505         \cs_set:Npn \l_stex_current_arity_str } {
5506             \int_step_function:nN \l_stex_current_arity_str \__stex_expr_return_arg:n
5507             \__stex_expr_return_next:
5508         }

```

```

5509 \__stex_expr_ret_cs:
5510 }
5511
5512 \cs_new:Nn \__stex_expr_return_arg:n {
5513 \tl_put_right:Nn \exp_not:N \l__stex_expr_return_args_tl {{## #1}}
5514 }
5515
5516 \cs_new_protected:Nn \__stex_expr_return_next: {
5517 \peek_charcode_remove:NTF ! {
5518 \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl \group_end:
5519 }\__stex_expr_return:
5520 }
5521
5522 \cs_new_protected:Nn \__stex_expr_return: {
5523 \tl_set:Ne \l__stex_expr_return_this_tl {
5524 \tl_if_empty:NTF \l_stex_return_notation_tl {
5525 \exp_after:wN \exp_after:wN \exp_after:wN
5526 \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
5527 \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl
5528 }
5529 }{
5530 \l_stex_return_notation_tl
5531 }
5532 }
5533 \stex_debug:nn{return}{Notation:~\meaning\l__stex_expr_return_this_tl}
5534
5535 \exp_after:wN
5536 \tl_put_left:Nn \exp_after:wN \l__stex_expr_return_this_tl \exp_after:wN {
5537 \exp_after:wN \group_begin: \l_stex_current_redo_tl
5538 }
5539 \exp_args:Nne \use:n {
5540 \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs:
5541 \cs_set:Npn \l_stex_current_arity_str } {
5542 \exp_args:No \exp_not:n \l_stex_current_return_tl
5543 }
5544 \stex_debug:nn{return){
5545 \meaning\__stex_expr_ret_cs:^^J
5546 \meaning\l__stex_expr_return_this_tl^^J
5547 \exp_args:No \exp_not:n \l__stex_expr_return_args_tl^^J
5548 }
5549 \exp_args:Nne \use:nn {
5550 \exp_after:wN \group_end: \__stex_expr_ret_cs:
5551 }{
5552 \exp_args:No \exp_not:n \l__stex_expr_return_args_tl
5553 {
5554 \exp_args:No \exp_not:n \l__stex_expr_return_this_tl
5555 \group_end:
5556 }
5557 }
5558 }

```

32.4.1 Term Annotations

`\stex_eat_exclamation_point:`

```
5559 \cs_new_protected:Nn \stex_eat_exclamation_point: {
5560   \peek_charcode_remove:NT ! \stex_eat_exclamation_point:
5561 }
```

(End of definition for \stex_eat_exclamation_point:. This function is documented on page 150.)

sfunction

We occasionally allow for semantic macros where they otherwise shouldn't be, if we are in “invisible” parts:

```
5562 \bool_new:N \stex_in_invisible_html_bool
```

`\stex_term_oms:nn`

```
5563 \stex_if_html_backend:TF {
5564   \cs_new_protected:Nn \stex_term_oms:nn {
5565     \bool_if:NTF\l_stex_in_this_bool{#2}{
5566       \__stex_expr_check_comp:
5567       \tl_if_empty:NTF \l_stex_current_term_tl {
5568         \__stex_expr_annotate:nnn{OMID}{#1}{#2}
5569       }{
5570         \__stex_expr_do_headterm:nn{#1}{#2}
5571       }
5572     }
5573   }{
5574     \cs_new_protected:Nn \stex_term_oms:nn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_check_comp:}
5575   }
5576 }
5577
5578 \cs_set_eq:NN \stex_term_oms_or_omv:nn \stex_term_oms:nn
```

(End of definition for \stex_term_oms:nn. This function is documented on page 150.)

sfunction

`\stex_term_omv:nn`

```
5579 \stex_if_html_backend:TF {
5580   \cs_new_protected:Nn \stex_term_omv:nn {
5581     \bool_if:NTF\l_stex_in_this_bool{#2}{
5582       \__stex_expr_check_comp:
5583       \tl_if_empty:NTF \l_stex_current_term_tl {
5584         \__stex_expr_annotate:nnn{OMV}{#1}{#2}
5585       }{
5586         \__stex_expr_do_headterm:nn{#1}{#2}
5587       }
5588     }
5589   }{
5590     \cs_new_protected:Nn \stex_term_omv:nn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_check_comp:}
5591   }
5592 }
```

(End of definition for \stex_term_omv:nn. This function is documented on page 150.)

sfunction

In the case where the “symbol” (OMS or OMV) is the field of some structure or otherwise a complex expression, we have to insert the full term representing the operator:

```

5593 \stex_if_html_backend:TF {
5594   \cs_new_protected:Nn \__stex_expr_do_headterm:nn {
5595     \bool_if:NTF \stex_in_invisible_html_bool {
5596       {\bool_set_true:N \l_stex_allow_semantic_bool
5597        \ensuremath{\l_stex_current_term_tl}
5598       }
5599     }{
5600       \__stex_expr_annotate:nnn{complex}{#1}{
5601         \stex_annotate_invisible:nn{data-ftml-headterm={}}{
5602           {\bool_set_true:N \l_stex_allow_semantic_bool
5603            \ensuremath{\l_stex_current_term_tl}
5604           }
5605         }
5606         #2
5607       }
5608     }
5609   }
5610 }{
5611   \cs_new_protected:Nn \__stex_expr_do_headterm:nn { #2 }
5612 }

```

`_stex_term_oma:nn`

```

5613 \stex_if_html_backend:TF {
5614   \cs_new_protected:Nn \_stex_term_oma:nn {
5615     \bool_if:NTF \l_stex_in_this_bool{#2}{
5616       \__stex_expr_check_comp:
5617       \__stex_expr_annotate:nnn{OMA}{#1}{
5618         \tl_if_empty:NF \l_stex_current_term_tl {
5619           \stex_annotate_invisible:nn{data-ftml-headterm={}}{
5620             \bool_set_true:N \l_stex_allow_semantic_bool
5621             \l_stex_current_term_tl
5622           }}
5623         }
5624         #2
5625       }
5626     }
5627   }
5628 }{
5629   \cs_new_protected:Nn \_stex_term_oma:nn {\bool_if:NF \l_stex_in_this_bool \__stex_expr_check_comp:
5630 }

```

(End of definition for `_stex_term_oma:nn`. This function is documented on page 150.)

sfunction

`_stex_term_omb:nn`

```

5631 \stex_if_html_backend:TF {
5632   \cs_new_protected:Nn \_stex_term_omb:nn {
5633     \bool_if:NTF \l_stex_in_this_bool{#2}{
5634       \__stex_expr_check_comp:
5635       \__stex_expr_annotate:nnn{OMBIND}{#1}{
5636         \tl_if_empty:NF \l_stex_current_term_tl {
5637           \stex_annotate_invisible:nn{data-ftml-headterm={}}{
5638             \bool_set_true:N \l_stex_allow_semantic_bool
5639             \l_stex_current_term_tl

```

```

5640         }}
5641     }
5642     #2
5643 }
5644 }
5645 }
5646 }{
5647   \cs_new_protected:Nn \_stex_term_omb:nn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_check:
5648 }

```

(End of definition for `_stex_term_omb:nn`. This function is documented on page 150.)

sfunction

Does the actual annotating: **TODO: This shouldn't use `\l_stex_current_symbol_uri`!**

```

5649 % kind, notation id, body
5650 \stex_if_html_backend:TF {
5651   \cs_new_protected:Nn \__stex_expr_annotate:nnn {
5652     \exp_args:Ne \stex_annotate:nn{
5653       data-ftml-term=#1,
5654       data-ftml-head={\l_stex_current_full_tl},
5655       data-ftml-notationid={#2},
5656     }{
5657       \_stex_annotate_force_break:n{#3}
5658     }
5659   }
5660 }{
5661   \cs_new_protected:Nn \__stex_expr_annotate:nnn { #3 }
5662 }
5663

```

32.4.2 Symbol Arguments

`_stex_term_arg:nnnnn` #1 : argument number
 #2 : argument mode
 #3 : precedence
 #4 : argument name (WiP)
 #5 : code / body

```

5664 \cs_new_protected:Nn \_stex_term_arg:nnnnn {
5665   \group_begin:
5666     \tl_clear:N \l_stex_current_symbol_uri
5667     \tl_clear:N \l_stex_current_full_tl
5668     \tl_clear:N \l_stex_current_display_tl
5669     \tl_clear:N \l_stex_current_term_tl
5670     \int_set:Nn \l_stex_notation_downprec { #3 }
5671     \bool_set_true:N \l_stex_allow_semantic_bool
5672     \_stex_term_arg:nnn {#2}{#1}{#5}
5673   \group_end:
5674 }

```

(End of definition for `_stex_term_arg:nnnnn`. This function is documented on page 151.)

sfunction

`_stex_term_arg:nnn`

```

5675 \cs_new_protected:Nn \__stex_expr_check_comp: {
5676   \bool_if:NT \g_stex_in_comp_bool {
5677     \msg_error:nn{stex}{error/argincomp}
5678   }
5679 }
5680 \stex_if_html_backend:TF {
5681   \cs_new_protected:Nn \_stex_term_arg:nnn {
5682     \stex_if_do_html:TF{
5683       \bool_if:NTF\l_stex_in_this_bool{#3}{
5684         \stex_annotate_force_break:n{
5685           \stex_annotate:nn{
5686             data-ftml-arg={#2}, data-ftml-argmode={#1}
5687           }{
5688             \stex_annotate_force_break:n{}
5689             \__stex_expr_check_comp:
5690             #3
5691           }
5692         }
5693       }
5694     }{ #3 }
5695   }
5696 }{
5697   \cs_new_protected:Nn \_stex_term_arg:nnn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_che
5698 }

```

(End of definition for `_stex_term_arg:nnn`. This function is documented on page 151.)

sfunction

`_stex_term_arg_aB:nnnnn` The following macros fill up `\l_stex_aB_args_seq` and ultimately call `_stex_term_do_aB_clist:`. By default, this does:

```

5699 \cs_new:Nn \__stex_expr_do_aB_clist: {
5700   \stex_annotate_force_break:n{
5701     \seq_use:Nn \l_stex_aB_args_seq {
5702       \mathpunct{\comp{,}}
5703     }
5704   }
5705 }
5706
5707 \cs_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5708
5709 \cs_new_protected:Nn \_stex_is_sequentialized:n {
5710   \group_begin: #1 \group_end:
5711 }

```

Setup:

```

5712 \int_new:N \l__stex_expr_count_int
5713 \cs_new_protected:Nn \_stex_term_arg_aB:nnnnn {
5714   \stex_debug:nn{sequences}{Processing~argument:~ \tl_to_str:n{#5}}
5715   \tl_if_empty:NTF{#5}{
5716     \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{}
5717   }{
5718     \seq_clear:N \l_stex_aB_args_seq
5719     \int_zero:N \l__stex_expr_count_int

```

```

5720 \clist_map_inline:nn{#5}{
5721   \__stex_expr_aB_arg:nnnnn{##1}{#1}{#2}{#3}{#4}
5722 }
5723 \_stex_term_do_aB_clist:
5724 }
5725 }

```

We “pattern match” the sequence argument - is it a comma-separated list? A sequence variable macro? `\argmap?` `\seqmap?`

```

5726 % 1: code 2: argnum 3: argmode 4: precedence 5: argname
5727 \cs_new_protected:Npn \__stex_expr_aB_arg:nnnnn #1 #2 #3 {
5728   \int_incr:N \l__stex_expr_count_int
5729   \stex_debug:nn{expressions}{matching~argument~ #2 #3:~ \tl_to_str:n{#1}}
5730   \__stex_expr_is_varseq:nTF{#1}{
5731     \stex_debug:nn{expressions}{=sequence~variable}
5732     \exp_after:wN \exp_after:wN \exp_after:wN
5733     \__stex_expr_assoc_seq:nnnnnnn
5734     \exp_after:wN
5735     \__stex_expr_gobble:nnnnnnnn #1 \__stex_expr_end:
5736   }{
5737     \__stex_expr_is_seqmap:nTF{#1}{
5738       \stex_debug:nn{expressions}{=seqmap}
5739       \exp_args:NNe \use:nn \__stex_expr_do_seqmap:nnnnnn {\tl_tail:n{#1}}
5740     }{
5741       \stex_debug:nn{expressions}{=simple}
5742       \__stex_expr_aB_simple_arg:nnnnn{#1}
5743     }
5744   }
5745   {#2}{#3}
5746 }
5747
5748 \cs_new:Npn \__stex_expr_gobble:nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8 #9 \__stex_expr_end: {
5749   {#2} #3 {#6}
5750 }

```

Is it a varseq?

```

5751 \prg_new_conditional:Nnn \__stex_expr_is_varseq:n {TF} {
5752   \int_compare:nNnTF {\tl_count:n{#1}} = 1 {
5753     \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_head:n{#1}}
5754     \_stex_invoke_variable:nnnnnnN {
5755       \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No\tl_item:nn{#1}{8}}
5756       \stex_invoke_sequence:
5757       \prg_return_true:\prg_return_false:
5758     }\prg_return_false:
5759   }\prg_return_false:
5760 }

```

Is it a seqmap?

```

5761 \prg_new_conditional:Nnn \__stex_expr_is_seqmap:n {TF} {
5762   \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
5763     \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\seqmap}
5764     \prg_return_true:\prg_return_false:
5765   }\prg_return_false:
5766 }

```


Is it an argsep?

```

5767 \prg_new_conditional:Nnn \__stex_expr_is_argsep:n {TF} {
5768   \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
5769     \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\argsep}
5770     \prg_return_true:\prg_return_false:
5771   }\prg_return_false:
5772 }

```

Simple argument:

```

5773 \cs_new_protected:Nn \__stex_expr_aB_simple_arg:nnnnn{
5774   \seq_put_right:Ne \l_stex_aB_args_seq {
5775     \stex_term_arg:nnnnn{#2\int_use:N\l__stex_expr_count_int}{#3}{#4}{#5}{
5776       \exp_not:n{
5777         \tl_set_eq:NN \stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5778         #1
5779       }
5780     }
5781   }
5782 }

```

Sequence variable:

```

5783 % 1: name 2: arity 3: clist 4: argnum 5: argmode 6: precedence 7: argname
5784 \cs_new_protected:Nn \__stex_expr_assoc_seq:nnnnnnn {
5785   \group_begin:
5786     \seq_clear:N \l_stex_aB_args_seq
5787     \tl_set:Nn \l_stex_current_full_tl{#1}
5788     \tl_set:Nn \l_stex_current_display_tl{#1}
5789     \__stex_expr_assoc_make_seq:nnn{#1}{#3}{#2}
5790   \exp_args:NNe \use:nn \group_end: {
5791     \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5792       \stex_is_sequentialized:n{
5793         \stex_term_arg:nnnnn{#4\int_use:N\l__stex_expr_count_int}{#5}{#6}{#7}{
5794           \bool_set_true:N \l_stex_allow_semantic_bool
5795           \tl_if_empty:NF \l_stex_current_term_tl {
5796             \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5797               \exp_args:No \exp_not:n \l_stex_current_term_tl
5798             }
5799           }
5800           \stex_annotate:nn{
5801             data-ftml-term=OMV,
5802             data-ftml-head={#1},
5803             data-ftml-notationid={}
5804           }{
5805             \stex_annotate_force_break:n{
5806               \stex_term_do_aB_clist:
5807             }
5808           }
5809         }
5810       }
5811     }
5812   }
5813 }
5814
5815 % #1: name, #2: clist, #3:arity
5816 \cs_new_protected:Nn \__stex_expr_assoc_make_seq:nnn {

```

```

5817 \stex_use_notation:nnTF {#1}{}{
5818 \cs_set_eq:NN \l__stex_expr_cs \l_stex_notation_cs
5819 }{
5820 \stex_do_default_notation:
5821 \cs_set_eq:NN \l__stex_expr_cs \l_stex_default_notation
5822 }
5823 \clist_map_inline:nn{#2}{
5824 \tl_if_eq:nnTF{##1}{\ellipses}{
5825 \seq_put_right:Nn \l_stex_aB_args_seq { ##1 }
5826 }{
5827 \int_compare:nNnTF {#3} = 1 {
5828 \tl_set:Nn \l__stex_expr_iarg_tl { {##1} }
5829 }{
5830 \tl_set:Nn \l__stex_expr_iarg_tl { ##1 }
5831 }
5832 \seq_put_right:Ne \l_stex_aB_args_seq {
5833 \group_begin:
5834 \exp_not:n {
5835 \tl_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5836 \def\comp{\_varcomp}
5837 \tl_set:Nn \l_stex_current_full_tl{#1}
5838 }
5839 \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
5840 \exp_after:wN \exp_after:wN \exp_after:wN {
5841 \exp_after:wN \l__stex_expr_cs \exp_after:wN \group_end: \l__stex_expr_iarg_tl
5842 }
5843 }
5844 }
5845 }
5846 }

seqmap:
5847 % 1: fun 2: clist 3: argnum 4: argmode 5: precedence 6: argname
5848 \cs_new_protected:Nn \__stex_expr_do_seqmap:nnnnnn {
5849 \group_begin:
5850 \cs_set:Npn \l_tmpa_cs ##1 {#1}
5851 \seq_clear:N \l_stex_aB_args_seq
5852 \clist_map_inline:nn{#2}{
5853 \__stex_expr_is_varseq:nTF{##1}{
5854 \exp_after:wN
5855 \__stex_expr_varseq_in_map:nnnnnnnn ##1
5856 }{
5857 \seq_put_right:Nn \l_stex_aB_args_seq {##1}
5858 }
5859 }
5860 \seq_clear:N \l__stex_expr_old_seq
5861 \seq_map_inline:Nn \l_stex_aB_args_seq {
5862 \tl_if_eq:nnTF{##1}{\ellipses}{
5863 \seq_put_right:Nn \l__stex_expr_old_seq {##1}
5864 }
5865 % TODO \_stex_is_sequentialized:n
5866 {
5867 \exp_args:NNo \seq_put_right:Nn \l__stex_expr_old_seq {
5868 \l_tmpa_cs {##1}
5869 }

```

```

5870     }
5871   }
5872   \seq_set_eq:NN \l_stex_aB_args_seq \l__stex_expr_old_seq
5873   \tl_set:Ne \l_tmpa_tl {
5874     \symuse{Metatheory?bind}{
5875       \svar[MAP_DUMMY]{\exp_not:N\mathcal{m}}
5876     }{
5877       \exp_args:No\exp_not:n{\l_tmpa_cs{\svar[MAP_DUMMY]{\mathcal{m}}}}
5878     }
5879   }
5880   \exp_args:NNe \use:nn \group_end: {
5881     \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5882       \_stex_is_sequentialized:n{
5883         \_stex_term_arg:nnnnn{#3\int_use:N\l__stex_expr_count_int}{#4}{#5}{#6}{
5884           \bool_set_true:N \l_stex_allow_semantic_bool
5885           \tl_set:Nn \exp_not:N \l_stex_current_full_tl
5886             {\l_stex_current_full_tl}
5887           \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5888             \exp_not:N \symuse{Metatheory?sequence-map}
5889             {\exp_not:n{#2}}
5890             {\exp_args:No \exp_not:n \l_tmpa_tl}
5891           }
5892           \__stex_expr_do_headterm:nn{}{
5893             \_stex_term_do_aB_clist:
5894           }
5895         }
5896       }
5897     }
5898   }
5899 }
5900
5901 \cs_new_protected:Nn \__stex_expr_varseq_in_map:nnnnnnnn {
5902   \__stex_expr_assoc_make_seq:nnn{#2}{#6}{#3}
5903 }
5904

```

(End of definition for `_stex_term_arg_aB:nnnnn`. This function is documented on page 151.)

sfunction

`\seqmap`

```

5905 \cs_new_protected:Npn \seqmap #1 #2 {
5906   \symuse{Metatheory?sequence-expression}{\seqmap{#1}{#2}}
5907   % \cs_set:Npn \l_tmpa_cs ##1 {#1}
5908   % \tl_set:Ne \l_tmpa_tl {
5909   %   \symuse{Metatheory?bind}{
5910   %     \svar[MAP_DUMMY]{\exp_not:N\mathcal{m}}
5911   %   }{
5912   %     \exp_args:No\exp_not:n{\l_tmpa_cs{\svar[MAP_DUMMY]{\mathcal{m}}}}
5913   %   }
5914   % }
5915   % \__stex_expr_annotate:nnn{complex}{OMS}{
5916   %   \stex_annotate_invisible:nn{data-ftml-headterm={}}{
5917   %     {\bool_set_true:N \l_stex_allow_semantic_bool
5918   %       \ensuremath{

```

```

5919 %           \exp_args:Nnno \symuse{Metatheory?sequence~map}{#2}\l_tmpa_tl
5920 %           }
5921 %       }
5922 %   }
5923 %       \symuse{Metatheory?sequence~expression}{\seqmap{#1}{#2}}
5924 %   }
5925 }

```

(End of definition for `\seqmap`. This function is documented on page 151.)

sfunction

32.4.3 Notation components

```

5926 <@@=stex_comps>

```

```

\comp
\compemph@uri
\compemph

```

```

5927 \cs_set_protected:Npn \comp {}
5928
5929 \cs_new_protected:Npn \compemph@uri #1 #2 {
5930   \compemph{ #1 }
5931 }
5932
5933 \cs_new_protected:Npn \compemph #1 { #1 }
5934
5935 \cs_new_protected:Npn \_comp {
5936   \_do_comp:nnNn {comp}{ }\compemph@uri
5937 }
5938
5939 \cs_new_protected:Npn \_thiscomp #1 #2 {
5940   {\tl_set:cn{this}{ }}#1{#2}\c_math_subscript_token{
5941     \group_begin:
5942       \bool_set_true:N \l_stex_allow_semantic_bool
5943       \bool_set_true:N \l_stex_in_this_bool
5944       \l_stex_struct_this_tl
5945     \group_end:
5946   }
5947 }
5948 }
5949
5950 \def\maincomp{\comp}

```

(End of definition for `\comp`, `\compemph@uri`, and `\compemph`. These functions are documented on page 151.)

sfunction

```

\varemp@uri
\varemp

```

```

5951 \cs_new_protected:Npn \varemp@uri #1 #2 {
5952   \varemp{#1}
5953 }
5954
5955 \cs_new_protected:Npn \varemp #1 { #1 }
5956
5957 \cs_new_protected:Npn \_varcomp {
5958   \_do_comp:nnNn {comp}{ }\varemp@uri
5959 }

```

(End of definition for \varemp@uri and \varemp. These functions are documented on page 151.)
sfunction

```
\defemph@uri
\defemph
5960 \cs_new_protected:Npn \defemph@uri #1 #2 {
5961   \defemph{#1}
5962 }
5963
5964 \cs_new_protected:Npn \defemph #1 {
5965   \ifmmode\else\expandafter\textbf\fi{#1}
5966 }
5967
5968 \cs_new_protected:Npn \_defcomp {
5969   \_do_comp:nnNn {defcomp}{ }\defemph@uri
5970 }
```

(End of definition for \defemph@uri and \defemph. These functions are documented on page 151.)
sfunction

```
\symrefemph@uri
\symrefemph
5971 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
5972   \symrefemph{#1}
5973 }
5974
5975 \cs_new_protected:Npn \symrefemph #1 { \emph{#1} }
```

(End of definition for \symrefemph@uri and \symrefemph. These functions are documented on page 151.)

sfunction

The following commands do the actual attributes:

```
\_do_comp:nnNn
5976 \bool_new:N \g_stex_in_comp_bool
5977
5978 \cs_new_protected:Nn \_do_comp:nnNn {
5979   \stex_pseudogroup_with:nn{\comp}{
5980     \def\comp##1{##1}
5981     \bool_set_true:N \g_stex_in_comp_bool
5982     \stex_if_html_backend:TF {
5983       \stex_annotate:nn { data-ftml-#1 = {#2}}{ #4 }
5984     }{
5985       \tl_if_empty:NTF \l_stex_current_full_tl {
5986         #4
5987       }{
5988         #3 { #4 } \l_stex_current_full_tl
5989       }
5990     }
5991   }
5992   \bool_set_false:N \g_stex_in_comp_bool
5993 }
5994 }
```

(End of definition for _do_comp:nnNn. This function is documented on page 151.)
sfunction

32.4.4 Symbol References

Keys:

```

5995 \stex_keys_define:nnnn{symname}{
5996   \tl_clear:N \l_stex_key_pre_tl
5997   \tl_clear:N \l_stex_key_post_tl
5998   %\tl_clear:N \l_stex_key_root_tl
5999 }{
6000   pre   .tl_set:N = \l_stex_key_pre_tl ,
6001   post  .tl_set:N = \l_stex_key_post_tl ,
6002   %root .code:n   = {}%.tl_set:N = \l_stex_key_root_tl
6003 }{}
6004
6005 \stex_keys_define:nnnn{symref}{
6006   %\tl_clear:N \l_stex_key_root_tl
6007 }{
6008   root .code:n   = {}%.tl_set:N = \l_stex_key_root_tl
6009 }{}

```

\symref

\sr

```

6010 \NewDocumentCommand \symref { 0{} m m } {
6011   \group_begin:
6012   \stex_keys_set:nn{symname}{#1}
6013   \stex_get_symbol:n{#2}
6014   \__stex_comps_do_ref:nNn{#3}\symrefemph@uri\stex_term_oms:nn
6015 }
6016 \let\sr\symref

```

(End of definition for \symref and \sr. These functions are documented on page 152.)

sfunction

\symname

\sn

\sns

\Symname

\Sn\Sns

```

6017 \cs_new:Nn \__stex_comps_split_slash:n {
6018   \__stex_comps_split_slash_i:nw #1//
6019 }
6020
6021 \cs_new:Npn \__stex_comps_split_slash_i:nw #1/#2/ {
6022   \tl_if_empty:nTF{#2}{#1}{
6023     \__stex_comps_split_slash_i:nw #2/
6024   }
6025 }
6026
6027 \NewDocumentCommand \symname { 0{} m } {
6028   \group_begin:
6029   \stex_keys_set:nn{symname}{#1}
6030   \stex_get_symbol:n{#2}
6031   \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }
6032   \tl_set:Ne \l_stex_current_display_tl {
6033     \exp_args:Ne \__stex_comps_split_slash:n {\stex_symbol_uri_name:N \l_stex_get_symbol_uri
6034   }
6035   \__stex_comps_do_ref:nNn{
6036     \l_stex_key_pre_tl \l_stex_current_display_tl \l_stex_key_post_tl
6037   }\symrefemph@uri\stex_term_oms:nn
6038 }

```

```

6039 \let\sn\symname
6040 \protected\def\sns{\symname[post=s]}
6041
6042 \cs_new:Nn \__stex_comps_uppercase:n { \uppercase{#1}}
6043
6044 \NewDocumentCommand \Symname { 0{} m} {
6045   \group_begin:
6046   \stex_keys_set:nn{\symname}{#1}
6047   \stex_get_symbol:n{#2}
6048   \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }
6049   \tl_set:Ne \l_stex_current_display_tl {
6050     \exp_args:Ne \__stex_comps_split_slash:n {\stex_symbol_uri_name:N \l_stex_get_symbol_uri
6051   }
6052   \__stex_comps_do_ref:nNn{
6053     \l_stex_key_pre_tl \exp_args:NNe \use:nn \__stex_comps_uppercase:n \l_stex_current_disp
6054   }\symrefemph@uri\stex_term_oms:nn
6055 }
6056 \let\Sn\Symname
6057 \protected\def\Sns{\Symname[post=s]}

```

(End of definition for \symname and others. These functions are documented on page 152.)

sfunction

```

\varref
\varname
\Varname
6058 \NewDocumentCommand \varref { 0{} m m} {
6059   \group_begin:
6060   \stex_keys_set:nn{\symname}{#1}
6061   \stex_get_var:n{#2}
6062   \__stex_comps_do_ref:nNn{#3}\varemp@uri{
6063     \tl_set:Nn \l_stex_current_full_tl { \l_stex_get_variable_str }
6064     \tl_set:Nn \l_stex_current_display_tl {\l_stex_get_variable_str}
6065     \def\comp{\_varcomp}
6066     \stex_term_omv:nn
6067   }
6068 }
6069
6070 \NewDocumentCommand \varname { 0{} m} {
6071   \group_begin:
6072   \stex_keys_set:nn{\symname}{#1}
6073   \stex_get_var:n{#2}
6074   \__stex_comps_do_ref:nNn{
6075     \l_stex_key_pre_tl\l_stex_get_variable_str\l_stex_key_post_tl
6076   }\varemp@uri{
6077     \tl_set:Nn \l_stex_current_full_tl { \l_stex_get_variable_str }
6078     \tl_set:Nn \l_stex_current_display_tl {\l_stex_get_variable_str}
6079     \def\comp{\_varcomp}
6080     \stex_term_omv:nn
6081   }
6082 }
6083
6084 \NewDocumentCommand \Varname { 0{} m} {
6085   \group_begin:
6086   \stex_keys_set:nn{\symname}{#1}
6087   \stex_get_var:n{#2}

```

```

6088 \_stex_comps_do_ref:nNn{
6089   \l_stex_key_pre_tl\exp_after:wN\_stex_comps_uppercase:n\l_stex_get_variable_str\l_stex
6090 } \vareph@uri{
6091   \tl_set:Nn \l_stex_current_full_tl { \l_stex_get_variable_str }
6092   \tl_set:Nn \l_stex_current_display_tl {\l_stex_get_variable_str}
6093   \def\comp{\_varcomp}
6094   \_stex_term_omv:nn
6095 }
6096 }

```

(End of definition for `\varref`, `\varname`, and `\Varname`. These functions are documented on page 152.)

sfunction

`\definiendum`

`\definame`

`\Definame`

`\defnotation`

```

6097 \stex_keys_define:nnnn{defname}{-}{-}{
6098   gf      .code:n      = {},
6099   root    .code:n      = {}
6100 }{symname}
6101
6102 \stex_keys_define:nnnn{definiendum}{-}{-}{
6103   gf      .code:n      = {},
6104   root    .code:n      = {}
6105 }{-}
6106
6107 \NewDocumentCommand \definiendum { 0{} m m } {
6108   \stex_keys_set:nn{definiendum}{#1 }
6109   \_stex_comps_do_defref:nn{#2}{#3}
6110 }
6111 \stex_deactivate_macro:Nn \definiendum {definition~environments}
6112
6113 \NewDocumentCommand \definame { 0{} m } {
6114   \stex_keys_set:nn{definame}{#1}
6115   \_stex_comps_do_defref:nn{#2}{
6116     \l_stex_key_pre_tl\l_stex_current_display_tl\l_stex_key_post_tl
6117   }
6118 }
6119 \protected\def\definames{\definame[post=s]}
6120 \stex_deactivate_macro:Nn \definame {definition~environments}
6121
6122
6123 \NewDocumentCommand \Definame { 0{} m } {
6124   \stex_keys_set:nn{definame}{#1}
6125   \_stex_comps_do_defref:nn{#2}{
6126     \l_stex_key_pre_tl \exp_args:NNe \use:nn \_stex_comps_uppercase:n \l_stex_current_disp
6127   }
6128 }
6129 \protected\def\Definames{\Definame[post=s]}
6130 \stex_deactivate_macro:Nn \Definame {definition~environments}
6131
6132
6133 \NewDocumentCommand \defnotation{ m } {
6134   \_stex_next_symbol:n { \def\comp{\_defcomp}}#1
6135 }
6136 \stex_deactivate_macro:Nn \defnotation {definition~environments}

```


(End of definition for `\definiendum` and others. These functions are documented on page 152.)

sfunction

Does the actual referencing:

```

6137 \cs_new_protected:Nn \__stex_comps_do_ref:nNn {
6138   \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
6139   \bool_if:NTF \l_stex_allow_semantic_bool{
6140     \tl_set_eq:NN \l_stex_current_symbol_uri \l_stex_get_symbol_uri
6141     \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_current_symbol_uri
6142       %\str_set:Ne \l_stex_current_symbol_name_str { \stex_symbol_uri_name:N \l_stex_current_
6143       %\str_if_in:NnT \l_stex_current_symbol_name_str / {
6144       % \str_set:Ne \l_stex_current_symbol_name_str {
6145       %   \exp_after:wN \__stex_comps_slash:w \l_stex_current_symbol_name_str
6146       %   /\_stex_args_end:
6147       % }
6148       %}
6149       \tl_clear:N \l_stex_current_term_tl
6150       \def\comp{\_comp}
6151       \let\compemph@uri#2
6152       #3{\_comp{#1}}
6153   }{
6154     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_full_tl}
6155   }
6156   \group_end:
6157 }

\definame et al:

6158 \cs_new_protected:Nn \__stex_comps_do_defref:nn {
6159   \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
6160   \group_begin:
6161   \stex_get_symbol:n{#1}
6162   \bool_if:NTF \l_stex_allow_semantic_bool{
6163     \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }
6164     \tl_set:Nn \l_stex_current_display_tl {
6165       \stex_symbol_uri_name:N \l_stex_get_symbol_uri
6166     }
6167     %\str_if_in:NnT \l_stex_get_svariable_str / {
6168     % \str_set:Ne \l_stex_get_variable_str {
6169     %   \exp_after:wN \__stex_comps_slash:w \l_stex_get_variable_str
6170     %   /\_stex_args_end:
6171     % }
6172     %}
6173     \exp_args:Ne \stex_ref_new_sym_target:n \l_stex_current_full_tl
6174     \_do_comp:nnNn {definiendum}{\l_stex_current_full_tl}\defemph@uri{#2}
6175   }{
6176     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_full_tl}
6177   }
6178   \group_end:
6179 }

6180
6181 \cs_new:Npn \__stex_comps_slash:w #1/#2/#3\_stex_args_end: {
6182   \tl_if_empty:nTF{#3}{
6183     #2
6184   }{
6185     \__stex_comps_slash:w #2 / #3 \_stex_args_end:

```

```

6186   }
6187 }

\foldexpr

6188 \stex_if_html_backend:TF {
6189   \NewDocumentCommand \foldexpr { O{} mm } {
6190     \stex_annotate:nn{
6191       data-ftml-fold-expression={\exp_args:Ne \str_if_eq:nnT{#1}{show}{show}}
6192     }{
6193       \stex_annotate:nn{
6194         data-ftml-fold-short={}
6195       }{#2}
6196       #3
6197     }
6198   }
6199 }{
6200   \NewDocumentCommand \foldexpr { O{} mm } {
6201     \exp_args:Ne \str_if_eq:nnTF{#1}{hide}{
6202       \underbrace{...}{#2}
6203     }{
6204       \underbrace{#3}{#2}
6205     }
6206   }
6207 }

```

(End of definition for `\foldexpr`. This function is documented on page 152.)

sfunction

32.5 Checking

```

\stex_if_check_terms_p:
\stex_if_check_terms:TF
6208 <@@=stex_check>
6209 \stex_if_html_backend:TF {
6210   \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
6211     \prg_return_false:
6212   }
6213 }{
6214   \stex_get_env:Nn\__stex_check_env_str{STEX_CHECKTERMS}
6215   \str_if_empty:NF\__stex_check_env_str{
6216     \exp_args:No \str_if_eq:nnF \__stex_check_env_str{false}{
6217       \bool_set_true:N \c_stex_check_terms_bool
6218     }
6219   }
6220   \bool_if:NTF \c_stex_check_terms_bool {
6221     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
6222       \prg_return_true:
6223     }
6224   }{
6225     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
6226       \prg_return_false:
6227     }
6228   }
6229 }

```

(End of definition for `\stex_if_check_terms:TF`. This function is documented on page 152.)

sfunction

`\stex_check_term:nn`

```

6230 \stex_if_check_terms:TF{
6231   \cs_new_protected:Nn \stex_check_term:nn {
6232     \marginpar {\tiny Checking~#1:~
6233       \group_begin:
6234         $$2$
6235       \group_end:
6236     }
6237   }
6238 }{
6239   \cs_new_protected:Nn \stex_check_term:nn {}
6240 }
```

(End of definition for `\stex_check_term:nn`. This function is documented on page 152.)

sfunction

`_stex_check_terms:`

```

6241 \stex_if_check_terms:TF{
6242   \cs_new_protected:Nn \_stex_check_terms: {
6243     \stex_debug:nn{check_terms}{Checking~type...}
6244     \tl_if_empty:NF \l_stex_key_type_tl {
6245       \stex_check_term:nn{type}\l_stex_key_type_tl
6246     }
6247     \stex_debug:nn{check_terms}{Checking~definiens...}
6248     \tl_if_empty:NF \l_stex_key_def_tl {
6249       \stex_check_term:nn{definiens}\l_stex_key_def_tl
6250     }
6251     \stex_debug:nn{check_terms}{Checking~return...}
6252     \tl_if_empty:NF \l_stex_key_return_tl {
6253       \stex_check_term:nn{return}\l_stex_key_return_tl!}
6254     }
6255     \stex_debug:nn{check_terms}{Checking~argument~types...}
6256     \group_begin:\l_stex_key_argtypes_clist\group_end:
6257     \tl_if_empty:NF \l_stex_key_argtypes_clist {
6258       \stex_check_term:nn{argument~types}\l_stex_key_argtypes_clist}
6259     }
6260   }
6261 }{
6262   \cs_new_protected:Nn \_stex_check_terms: {}
6263 }
6264
```

(End of definition for `_stex_check_terms:.` This function is documented on page 152.)

sfunction

Chapter 33

Notations

```
6265 <@@=stex_notations>

      keys:
6266 \stex_keys_define:nnnn{notation}{
6267   \str_clear:N \l_stex_key_variant_str
6268   \str_clear:N \l_stex_key_prec_str
6269   \str_clear:N \l_stex_key_op_tl
6270   \str_clear:N \l_stex_key_intent_str
6271   \clist_clear:N \l_stex_key_intent_args_clist
6272 }{
6273   variant      .str_set_x:N = \l_stex_key_variant_str ,
6274   prec         .str_set_x:N = \l_stex_key_prec_str ,
6275   op           .tl_set:N     = \l_stex_key_op_tl ,
6276   intent       .str_set:N     = \l_stex_key_intent_str ,
6277   argnames     .clist_set:N   = \l_stex_key_intent_args_clist ,
6278   unknown      .code:n       = {
6279     \str_if_empty:NTF \l_keys_key_str {
6280       \str_set:Nc \l_stex_key_variant_str {\l_keys_key_tl}
6281     }{
6282       \str_set_eq:NN \l_stex_key_variant_str \l_keys_key_str
6283     }
6284   }
6285 }{style}

6286
6287 \stex_keys_define:nnnn{symdef}{ }{ }{decl,notation}

6288
6289 \stex_keys_define:nnnn{vardef}{
6290   \bool_set_false:N \l__stex_notations_bind_bool
6291 }{
6292   bind .bool_set:N = \l__stex_notations_bind_bool
6293 }{symdef}
6294

\notation

6295 \stex_new_stylable_cmd:nnnn {notation} { s m O{ } m } {
6296   \stex_keys_set:nn{notation}{#3}
6297   \stex_get_symbol:n{#2}
6298   \stex_notation_parse:n{#4}
6299   \stex_if_check_terms:T{ \_stex_notation_check: }
```

```

6300 \stex_notation_add:
6301 \stex_if_do_html:T {
6302   \def\comp{\_comp}
6303   \_stex_notation_do_html:n{\stex_use_symbol_uri:N \l_stex_get_symbol_uri}
6304 }
6305 \IfBooleanTF#1{
6306   \_stex_notation_set_default:n{
6307     \l_stex_get_symbol_uri
6308   }
6309 }{}
6310 \stex_if_smsmode:F{
6311   \group_begin:
6312   \__stex_notations_styledefs:
6313   \stex_style_apply:
6314   \group_end:
6315 }
6316 \stex_smsmode_do:
6317 }{}
6318
6319 \stex_deactivate_macro:Nn \notation {module~environments}
6320 \stex_every_module:n {\stex_reactivate_macro:N \notation}
6321 \stex_sms_allow_escape:N \notation
6322
6323 \cs_new_protected:Nn \__stex_notations_styledefs: {
6324   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
6325   \str_set:Nn \thisdeclname {\stex_symbol_uri_name:N \l_stex_get_symbol_uri}
6326   \tl_set:Ne \thisdecluri {\stex_use_symbol_uri:N \l_stex_get_symbol_uri}
6327   \def\thisnotation{
6328     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{ }{
6329       \_stex_notation_make_args:
6330     }
6331   }
6332 }

```

(End of definition for \notation. This function is documented on page 153.)

sfunction

\varnotation

```

6333 \stex_new_stylable_cmd:nnnn {\varnotation} { s m O{} m} {
6334   \stex_keys_set:nn{notation}{#3}
6335   \stex_get_var:n{#2}
6336   \str_set_eq:NN \l_stex_key_name_str \l_stex_get_variable_str
6337   \stex_notation_parse:n{#4}
6338   \stex_if_check_terms:T{ \_stex_notation_check: }
6339   \_stex_var_notation_macro:
6340   \IfBooleanTF#1{
6341     \_stex_notation_set_default:n{\l_stex_get_variable_str}
6342   }{}
6343   \group_begin:
6344   \tl_set_eq:NN \thisvarname \l_stex_get_variable_str
6345   \tl_clear:N \thisstyle
6346   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
6347   \def\thisnotation{
6348     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{ }{

```

```

6349     \_stex_notation_make_args:
6350   }
6351 }
6352 \stex_style_apply:
6353 \group_end:
6354 }{}

```

(End of definition for \varnotation. This function is documented on page 153.)

sfunction

\stex_notation_parse:n

```

6355 \cs_new_protected:Nn \stex_notation_parse:n {
6356   \tl_if_empty:NF \l_stex_key_op_tl {
6357     \tl_set:Nx \l_stex_key_op_tl { \exp_not:N\maincomp {
6358       \exp_args:No \exp_not:n \l_stex_key_op_tl
6359     } }
6360   }
6361   \seq_clear:N \l__stex_notations_precs_seq
6362   \tl_clear:N \l_stex_notation_args_tl
6363   \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0 {
6364     \__stex_notations_const_precs:
6365     \tl_if_empty:NT \l_stex_key_op_tl {
6366       \tl_set:Nn \l_stex_key_op_tl { \maincomp{#1} }
6367     }
6368   }{
6369     \__stex_notations_fun_precs:
6370     \__stex_notations_do_missing_args:n{#1}
6371     \tl_if_empty:NT \l_stex_key_op_tl {
6372       \hbox_set:Nn \l_tmpa_box {
6373         \tl_clear:N \l_stex_current_full_tl
6374         \tl_clear:N \l_stex_current_display_tl
6375         \cs_set:Npn \l_tmpa_cs ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
6376         \cs_set:Npn \maincomp ##1 {
6377           \tl_gset:Nn \l_stex_key_op_tl { \maincomp{##1} }
6378           ##1
6379         }
6380         \cs_set:Npn \argsep ##1 ##2 {##1 ##2}
6381         \cs_set:Npn \argmap ##1 ##2 ##3 {##1 ##3}
6382         \cs_set:Npn \argarraymap ##1 ##2 ##3 ##4 {
6383           ##1 ##2
6384         }
6385         \stex_suppress_html:n{$\l_tmpa_cs abcdefghj$}
6386       }
6387     }
6388   }
6389   \exp_args:NNe
6390   \tl_set:Nn \l_stex_notation_macrocode_cs {
6391     \STEXInternalNotation
6392     { \l_stex_key_variant_str }
6393     { \l__stex_notations_opprec_tl }
6394     { \l_stex_key_intent_str }
6395     { \l_stex_notation_args_tl }
6396     {
6397       \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0

```

```

6398     { \exp_not:n { \maincomp{ #1 } } }
6399     { \exp_not:n { #1 } \l__stex_notations_missing_tl }
6400   }
6401 }
6402 \stex_debug:nn{notation}{Notation:~\meaning\l_stex_notation_macrocode_cs}
6403 }

precedences:
6404 \cs_new_protected:Nn \__stex_notations_const_precs: {
6405   \str_if_empty:NTF \l_stex_key_prec_str {
6406     \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
6407   }{
6408     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
6409       \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
6410     }{
6411       \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
6412     }
6413   }
6414 }
6415
6416 \cs_new_protected:Nn \__stex_notations_fun_precs: {
6417   \str_if_empty:NTF \l_stex_key_prec_str {
6418     \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
6419   }{
6420     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
6421       \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
6422     }{
6423       \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
6424     }
6425   }
6426   \str_if_empty:NTF \l_stex_key_prec_str {
6427     \tl_set:Nn \l__stex_notations_opprec_tl { 0 }
6428     \int_step_inline:nn \l_stex_get_symbol_arity_int {
6429       \seq_put_right:Nn \l__stex_notations_precs_seq {0}
6430     }
6431   }{
6432     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
6433       \stex_debug:nn{notation}{No~brackets}
6434       \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
6435       \int_step_inline:nn \l_stex_get_symbol_arity_int {
6436         \exp_args:NNo \seq_put_right:Nn \l__stex_notations_precs_seq \infprec
6437       }
6438     } \__stex_notations_parse_precs:
6439   }
6440   \__stex_notations_do_argnames:
6441 }
6442
6443 \cs_new_protected:Nn \__stex_notations_parse_precs: {
6444   \stex_debug:nn{notation}{parsing~precedence~\l_stex_key_prec_str}
6445   \seq_set_split:NnV \l__stex_notations_seq ; \l_stex_key_prec_str
6446   \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
6447     \tl_set_eq:NN \l__stex_notations_opprec_tl \l__stex_notations_str
6448     \seq_pop_left:NNT \l__stex_notations_seq \l__stex_notations_str {
6449       \exp_args:NNo \seq_set_split:NnV \l__stex_notations_seq
6450         {\tl_to_str:n{x}} \l__stex_notations_str

```

```

6451     }
6452   }{
6453     \tl_set:No \l__stex_notations_opprec_tl { 0 }
6454   }
6455   \int_step_inline:nn \l_stex_get_symbol_arity_int {
6456     \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
6457       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_str
6458     }{
6459       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_opprec_tl
6460     }
6461   }
6462 }

```

Experimental; named arguments:

```

6463 \cs_new_protected:Nn \__stex_notations_do_argnames: {
6464   \tl_clear:N \l_stex_notation_args_tl
6465   \stex_map_args:N \__stex_notations_do_argname:nn
6466 }
6467
6468 \cs_new_protected:Nn \__stex_notations_do_argname:nn {
6469   \clist_if_empty:NNTF \l_stex_key_intent_args_clist {
6470     \tl_put_right:Ne \l_stex_notation_args_tl {
6471       #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}{
6472         \str_if_empty:NF \l_stex_key_intent_str {#1}
6473       }
6474     }
6475   }{
6476     \tl_put_right:Ne \l_stex_notation_args_tl {
6477       #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}
6478       {\c_dollar_str\clist_item:Nn \l_stex_key_intent_args_clist 1}
6479     }
6480     \clist_pop:NN \l_stex_key_intent_args_clist \l_tmpa_tl
6481   }
6482 }

```

inserts hidden arguments that don't occur in the body of the notation:

```

6483 \cs_new:Nn \__stex_notations_do_missing_args:n {
6484   \str_set:Nn \l__stex_notations_missing_str {#1}
6485   \exp_args:NNe \seq_set_split:NnV \l_tmpa_seq {\c_hash_str\c_hash_str\c_hash_str\c_hash_str}
6486   \str_clear:N \l__stex_notations_missing_str
6487   \seq_map_inline:Nn \l_tmpa_seq {
6488     \tl_put_right:Nn \l__stex_notations_missing_str { ##1 }
6489   }
6490   \tl_clear:N \l__stex_notations_missing_tl
6491   \stex_map_args:N \__stex_notations_add_missing_args:nn
6492 }
6493
6494 \cs_new:Nn \__stex_notations_add_missing_args:nn {
6495   % TODO this skips arguments if e.g. ##1 occurs rather than #1!!
6496   \exp_args:NNe \str_if_in:NnF \l__stex_notations_missing_str {\c_hash_str\c_hash_str#1}{
6497     \tl_put_right:Nn \l__stex_notations_missing_tl{\STEXinvisible{## #1}}
6498   }
6499 }

```

(End of definition for `\stex_notation_parse:n`. This function is documented on page 153.)

sfunction

_stex_notation_add:

```

6500 \cs_new_protected:Nn \_stex_notation_add: {
6501   \stex_add_notation:ooeoo
6502   {\l_stex_get_symbol_uri}
6503   \l_stex_key_variant_str
6504   {\int_use:N \l_stex_get_symbol_arity_int}
6505   \l_stex_notation_macrocode_cs
6506   \l_stex_key_op_tl
6507 }

```

(End of definition for _stex_notation_add:. This function is documented on page 153.)

sfunction

\stex_add_notation:nnnnn

\stex_add_notation:ooeoo

```

6508 \cs_new:Nn \__stex_notations_macro:nn {
6509   l_stex_notation_ #1?#2 _cs
6510 }
6511 \cs_new:Nn \__stex_notations_op_macro:nn {
6512   l_stex_notation_ #1?#2 _op_cs
6513 }
6514
6515 \cs_new_protected:Nn \stex_add_notation:nnnnn {
6516   \exp_args:Nne \stex_debug:nn{notations}{Adding~notation:^^J
6517     #1~\c_hash_str#2~#3^^J
6518     \tl_to_str:n{#4}^^J\tl_to_str:n{#5}^^J
6519     to~\stex_use_module_uri:N \l_stex_current_module_uri
6520   }
6521   \prop_gput:cen{ \_stex_notations_macro:N \l_stex_current_module_uri }
6522   {\stex_use_symbol_uri:n{#1}?#2}{\{#1\}{#2\}{#3\}{#4\}{#5}}
6523   \stex_do_up_to_module:n {
6524     \__stex_notations_set_macro:nnnnn{#1}{#2}{#3}{#4}{#5}
6525     %\__stex_notations_activate_not:nn{#1}{#2}
6526   }
6527 }
6528 \cs_generate_variant:Nn \stex_add_notation:nnnnn {ooeoo}
6529
6530 \cs_new_protected:Nn \_stex_activate_notations: {
6531   \stex_debug:nn{activating}{All~notations:~\cs_meaning:c{ \_stex_notations_macro:N \l_stex
6532   \prop_map_inline:cn{ \_stex_notations_macro:N \l_stex_current_module_uri }{
6533     \__stex_notations_set_macro:nnnnn ##2
6534   }
6535 }
6536
6537 \cs_new_protected:Nn \__stex_notations_activate_not:nn {
6538   \bool_set_false:N \l_tmpa_bool
6539   \stex_debug:nn{notations}{activating~\stex_use_symbol_uri:n{#1}?#2}
6540   \prop_map_inline:cn{ \_stex_notations_macro:N \l_stex_current_module_uri }{
6541     %\stex_debug:nn{notations}{##1?}
6542     \exp_args:Ne \str_if_eq:nnT{\stex_use_symbol_uri:n{#1}?#2}{##1}{
6543       \prop_map_break:n{
6544         %\stex_debug:nn{notations}{Found:~\tl_to_str:n{##2}}
6545         \bool_set_true:N \l_tmpa_bool

```

```

6546         \_stex_notations_set_macro:nnnnn ##2
6547     }
6548 }
6549 }
6550 \bool_if:NF \l_tmpa_bool {
6551     \errmessage{Woop}
6552 }
6553 }
6554
6555 \cs_new_protected:Nn \_stex_notations_set_macro:nnnnn {
6556     \tl_set:cn {\_stex_notations_macro:nn{\stex_use_symbol_uri:n{#1}}{#2}}{#4}
6557     \cs_if_exist:cF{\_stex_notations_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{
6558         \tl_set:cn {\_stex_notations_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{#4}
6559     }
6560     \tl_if_empty:nF{#5}{
6561         \tl_set:cn{\_stex_notations_op_macro:nn{\stex_use_symbol_uri:n{#1}}{#2}}{#5}
6562         \cs_if_exist:cF{\_stex_notations_op_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{
6563             \tl_set:cn{\_stex_notations_op_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{#5}
6564         }
6565     }
6566 }

```

(End of definition for \stex_add_notation:nnnnn. This function is documented on page 153.)

sfunction

_stex_map_args:N
_stex_map_notation_args:N

```

6567 \cs_new:Nn \_stex_map_args:N {
6568     \tl_if_empty:NF \l_stex_get_symbol_args_tl {
6569         \exp_after:wN \_stex_notations_map_args_i:w \exp_after:wN
6570         #1 \l_stex_get_symbol_args_tl \_stex_notations_args_end:
6571     }
6572 }
6573 \cs_new:Npn \_stex_notations_map_args_i:w #1 #2 #3 #4 \_stex_notations_args_end: {
6574     #1 {#2} {#3}
6575     \tl_if_empty:NF{#4}{
6576         \_stex_notations_map_args_i:w #1 #4 \_stex_notations_args_end:
6577     }
6578 }
6579
6580 \cs_new:Nn \_stex_map_notation_args:N {
6581     \tl_if_empty:NF \l_stex_notation_args_tl {
6582         \exp_after:wN \_stex_notations_map_args_ii:w \exp_after:wN
6583         #1 \l_stex_notation_args_tl \_stex_notations_args_end:
6584     }
6585 }
6586 \cs_new:Npn \_stex_notations_map_args_ii:w #1 #2 #3 #4 #5 #6 \_stex_notations_args_end: {
6587     #1 {#2} {#3} {#4} {#5}
6588     \tl_if_empty:NF{#6}{
6589         \_stex_notations_map_args_ii:w #1 #6 \_stex_notations_args_end:
6590     }
6591 }

```

(End of definition for _stex_map_args:N and _stex_map_notation_args:N. These functions are documented on page 153.)

sfunction

_stex_var_notation_macro:

```

6592 \cs_new_protected:Nn \_stex_var_notation_macro: {
6593   \tl_set_eq:cN {\_stex_notations_macro:nn{\l_stex_key_name_str}{\l_stex_key_variant_str}}
6594   \cs_if_exist:cF {\_stex_notations_macro:nn{\l_stex_key_name_str}{}}{
6595     \tl_set_eq:cN{\_stex_notations_macro:nn{\l_stex_key_name_str}{}}
6596     \l_stex_notation_macrocode_cs
6597   }
6598   \tl_if_empty:NF \l_stex_key_op_tl {
6599     \tl_set_eq:cN {\_stex_notations_op_macro:nn{\l_stex_key_name_str}{\l_stex_key_variant_
6600     \cs_if_exist:cF{\_stex_notations_op_macro:nn{\l_stex_key_name_str}{}}{
6601       \cs_set_eq:cN{\_stex_notations_op_macro:nn{\l_stex_key_name_str}{}}
6602       \l_stex_key_op_tl
6603     }
6604   }
6605 }

```

(End of definition for _stex_var_notation_macro:. This function is documented on page 153.)

sfunction

\setnotation

_stex_notation_set_default:n

```

6606 \cs_new_protected:Nn \_stex_notation_set_default:n{
6607   \stex_if_in_module:TF{
6608     \stex_add_notation:ooeo{#1}{
6609       {\int_use:N \l_stex_get_symbol_arity_int}
6610       \l_stex_notation_macrocode_cs
6611       \l_stex_key_op_tl
6612     }{
6613       \cs_set_eq:cN {\_stex_notations_macro:nn {\stex_use_symbol_uri:N \l_stex_get_symbol_ur
6614       \l_stex_notation_macrocode_cs
6615       \tl_if_empty:NF \l_stex_key_op_tl {
6616         \cs_set_eq:cN{\_stex_notations_op_macro:nn{\stex_use_symbol_uri:N \l_stex_get_symbol
6617         \l_stex_key_op_tl
6618       }
6619     }
6620   }
6621
6622   \cs_new_protected:Npn \setnotation #1 #2 {
6623     \stex_get_symbol:n{#1}
6624     \cs_if_exist:cTF{\_stex_notations_macro:nn{\stex_use_symbol_uri:N \l_stex_get_symbol_uri
6625     \tl_set_eq:Nc \l_stex_notation_macrocode_cs {\_stex_notations_macro:nn{\stex_use_symbol
6626     \cs_if_exist:cTF{\_stex_notations_op_macro:nn{\stex_use_symbol_uri:N \l_stex_get_symbol
6627     \tl_set_eq:Nc \l_stex_key_op_tl {\_stex_notations_op_macro:nn{\stex_use_symbol_uri:N
6628     }{
6629     \tl_clear:N \l_stex_key_op_tl
6630   }
6631   \_stex_notation_set_default:n{
6632     \l_stex_get_symbol_uri
6633   }
6634   }{
6635     \msg_error:nnxx{stex}{error/unknownnotation}{#2}{
6636       \stex_use_symbol_uri:N \l_stex_get_symbol_uri
6637     }
6638   }
6639 }

```

(End of definition for `\setnotation` and `_stex_notation_set_default:n`. These functions are documented on page 153.)

sfunction

`\stex_iterate_notations:nn`

```

6640 \cs_new_protected:Nn \stex_iterate_notations:nn {
6641   \seq_clear:N \l__stex_notations_mods_seq
6642   \stex_pseudogroup_with:nn{\__stex_notations_not_cs:nnnnn}{
6643     \cs_set:Npn \__stex_notations_not_cs:nnnnn
6644       ##1 ##2 ##3 ##4 ##5 { #2 }
6645     \clist_map_function:nN {#1} \__stex_notations_it_not_i:n
6646   }
6647 }
6648
6649 \cs_new_protected:Nn \__stex_notations_it_not_i:n {
6650   \seq_if_in:NnF \l__stex_notations_mods_seq {#1} {
6651     \seq_put_left:Nn \l__stex_notations_mods_seq {#1}
6652     \prop_map_inline:cn{\_stex_notations_macro:n{#1}}{
6653       \__stex_notations_not_cs:nnnnn ##2
6654     }
6655     \prop_map_inline:cn{\_stex_morphisms_macro:n{#1}}{
6656       \__stex_notations_it_not_check:nnnn ##2
6657     }
6658   }
6659 }
6660 \cs_new_protected:Nn \__stex_notations_it_not_check:nnnn {
6661   \tl_if_empty:nT{#1}{
6662     \__stex_notations_it_not_i:n {#2}
6663   }
6664 }

```

(End of definition for `\stex_iterate_notations:nn`. This function is documented on page 154.)

sfunction

`\stex_use_notation:nnTF`
`\stex_use_op_notation:nnTF`

```

6665 \cs_new_protected:Npn \stex_use_notation:nnTF #1 #2 #3 #4 {
6666   \cs_if_exist:cTF{\_stex_notations_macro:nn{#1}{#2}}{
6667     \cs_set_eq:Nc \l_stex_notation_cs {\_stex_notations_macro:nn{#1}{#2}}
6668     #3
6669   }{#4}
6670 }
6671 \cs_new_protected:Npn \stex_use_op_notation:nnTF #1 #2 #3 #4 {
6672   \cs_if_exist:cTF{\_stex_notations_op_macro:nn{#1}{#2}}{
6673     \cs_set_eq:Nc \l_stex_notation_cs {\_stex_notations_op_macro:nn{#1}{#2}}
6674     #3
6675   }{#4}
6676 }

```

(End of definition for `\stex_use_notation:nnTF` and `\stex_use_op_notation:nnTF`. These functions are documented on page 154.)

sfunction

`_stex_notation_check:`

```

6677 \cs_new_protected:Nn \_stex_notation_check: {

```

```

6678 \stex_check_term:nn{notation}{
6679   \cs_set:Npn \comp ##1 {##1}
6680   \stex_debug:nn{check}{Checking~notation:~\cs_meaning:N \l_stex_notation_macrocode_cs ^^
6681   \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}}{
6682     \stex_notation_make_args:
6683   }
6684 }
6685 }

```

(End of definition for `_stex_notation_check:`. This function is documented on page 154.)

sfunction

`_stex_notation_do_html:n`

```

6686 \cs_new_protected:Nn \_stex_notation_do_html:n {
6687   \stex_annotate_invisible:n{\_stex_annotate_force_break:n{\hbox{\stex_annotate:nn {
6688     data-ftml-notation={#1},
6689     data-ftml-notationfragment={\l_stex_key_variant_str},
6690     data-ftml-precedence={\l__stex_notations_opprec_tl},
6691     data-ftml-argprec={\seq_use:Nn \l__stex_notations_precs_seq ,}
6692   }}{
6693     \cs_set_protected:Npn \argsep ##1 ##2 {
6694       \stex_annotate:nn{data-ftml-argsep={}}{
6695         ##1 \tl_if_empty:nTF{##2}{\!\,}{##2}
6696       }
6697     }
6698     \cs_set_protected:Npn \argmap ##1 ##2 ##3 {
6699       \cs_set:Npn \__stex_notations_map_cs: #####1 { ##2 }
6700       \stex_annotate:nn{data-ftml-argmap={}}{
6701         \__stex_notations_map_cs:{##1} \stex_annotate:nn{data-ftml-argmap-sep={}}{##3}
6702       }
6703     }
6704     \cs_set_protected:Npn \maincomp {
6705       \do_comp:nnNn {maincomp}{}\compemph@uri
6706     }
6707     \stex_debug:nn{notation}{Doing~notation~HTML:\meaning\l_stex_get_symbol_args_tl}
6708     $
6709     \tl_clear:N \l_stex_current_full_tl
6710     \stex_annotate:nn{data-ftml-notationcomp={}}{
6711       \exp_args:Nne \use:nn {
6712         \l_stex_notation_macrocode_cs {}
6713       }{
6714         \stex_map_args:N \__stex_notations_make_arg_html:nn
6715       }
6716     }
6717     $
6718     \tl_if_empty:NF \l_stex_key_op_tl {
6719       $
6720       \tl_clear:N \l_stex_current_full_tl
6721       \stex_annotate:nn{data-ftml-notationopcomp={}}{
6722         \stex_term_oms:nn{\l_stex_key_variant_str}{\l_stex_key_op_tl}
6723       }
6724       $
6725     }
6726   }}

```

```

6727   }}
6728 }
6729
6730 \cs_new:Nn \__stex_notations_make_arg_html:nn {
6731   {\stex_annotate:nn{data-ftml-argnum=#1}{x}}
6732 }

```

(End of definition for `_stex_notation_do_html:n`. This function is documented on page 154.)

sfunction

`_stex_notation_make_args:`

```

6733 \cs_new:Nn \_stex_notation_make_args: {
6734   \_stex_map_notation_args:N \_stex_notations_make_arg:nnnn
6735 }
6736
6737 \cs_new:Nn \__stex_notations_make_arg:nnnn {
6738   \str_case:nnF #2 {
6739     a {{
6740       a\c_math_subscript_token{#1,1},
6741       a\c_math_subscript_token{#1,2}
6742     }}
6743     B {{
6744       B\c_math_subscript_token{#1,1},
6745       B\c_math_subscript_token{#1,2}
6746     }}
6747   }{{
6748     \_stex_term_arg:nnnn{#1}{#2}{#3}{#4}
6749     {{#2}\c_math_subscript_token{#1}}
6750   }}
6751 }

```

(End of definition for `_stex_notation_make_args:.` This function is documented on page 154.)

sfunction

`\stex_do_default_notation:`

`\stex_do_default_notation_op:`

```

6752 \cs_new_protected:Nn \stex_do_default_notation: {
6753   \stex_do_default_notation_op:
6754   \tl_if_empty:NTF \l_stex_current_args_tl {
6755     \tl_clear:N \l__stex_notations_args_tl
6756   }\_stex_notations_default_args:
6757   \tl_set:Ne \l_stex_default_notation {\STEXInternalNotation{}{0}{}}{\l__stex_notations_args_tl}
6758   \exp_args:No \exp_not:n \l_stex_default_notation
6759   }}
6760 }
6761
6762 \cs_new_protected:Nn \stex_do_default_notation_op: {
6763   \_stex_notations_make_name:
6764   \tl_set:Ne \l_stex_default_notation {\exp_not:N \maincomp{ \exp_not:N \mathrm {\l__stex_notations_make_name}}
6765 }
6766
6767 \cs_new_protected:Nn \__stex_notations_default_args: {
6768   \_stex_notations_make_name:
6769   \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
6770   \tl_set:Ne \l__stex_notations_args_tl {

```

```

6771 \stex_map_args:N \__stex_notations_augment_arg:nn
6772 }
6773 \tl_put_right:Nn \l_stex_default_notation {\comp{}}
6774 \seq_clear:N \l_tmpa_seq
6775 \int_step_inline:nn \l_stex_current_arity_str {
6776   \seq_put_right:Nn \l_tmpa_seq {#### #1}
6777 }
6778 \tl_put_right:Ne \l_stex_default_notation {
6779   \seq_use:Nn \l_tmpa_seq {\mathpunct{\comp{,}}}}
6780 }
6781 \tl_put_right:Nn \l_stex_default_notation {\comp{}}
6782 }
6783
6784 \cs_new:Nn \__stex_notations_augment_arg:nn {
6785   #1#2{0}{}}
6786 }
6787
6788 \cs_new_protected:Nn \__stex_notations_make_name: {
6789   \exp_args:NNNe \seq_set_split:Nnn \l_tmpa_seq / \l_stex_current_display_tl
6790   \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
6791 }

```

(End of definition for `\stex_do_default_notation:` and `\stex_do_default_notation_op:`. These functions are documented on page 154.)

sfunction

\STEXInternalNotation

```

6792 \cs_new_protected:Npn \STEXInternalNotation #1 #2 #3 #4 #5 #6 {
6793   \__stex_notations_process:nnnnnn{#1}{#2}{#3}{#4}{#5}{
6794     \l__stex_notations_code_tl
6795     #6
6796   }
6797 }
6798
6799 \cs_new_protected:Npn \__stex_notations_process:nnnnnn #1 #2 #3 #4 {
6800   \tl_if_empty:nTF{#4}{
6801     \__stex_notations_simple:nnnnn{#1}{#2}{#3}
6802   }{
6803     \__stex_notations_complex:nnnnnn{#1}{#2}{#3}{#4}
6804   }
6805 }
6806
6807 \cs_new_protected:Nn \__stex_notations_simple:nnnnn {
6808   \stex_debug:nn{Notation~code}{\tl_to_str:n{#4}}
6809   \tl_set:Nn \l__stex_notations_code_tl {
6810     \cs_set:Npn \l__stex_notations_cs {
6811       \stex_maybe_brackets:nn{#2}{
6812         \stex_term_oms_or_omv:nn{#1}{#4}
6813       }
6814     }
6815     \l__stex_notations_cs
6816   }
6817   #5
6818 }

```

```

6819
6820 \cs_new_protected:Nn \__stex_notations_complex:nnnnnn {
6821   \stex_debug:nn{Notation~code}{\tl_to_str:n{#5}}
6822   \int_zero:N \l_tmpa_int
6823   \tl_set:Nn \l__stex_notations_pre_tl {\cs_set_eq:NN \stex_term_oma_or_omb:nn \stex_term_
6824   \tl_set:Nn \l__stex_notations_code_tl {
6825     \cs_generate_from_arg_count:NNnn \l__stex_notations_cs \cs_set:Npn \l_tmpa_int
6826     {
6827       \stex_maybe_brackets:nn{#2}{
6828         \stex_term_oma_or_omb:nn{#1}{
6829           \bool_set_false:N \l_stex_brackets_dones_bool
6830           #5
6831         }
6832       }
6833     }
6834     \l__stex_notations_cs
6835   }
6836   \tl_set:Nn \l__stex_notations_after_tl{
6837     \exp_args:NNo
6838     \tl_put_left:Nn \l__stex_notations_code_tl \l__stex_notations_pre_tl
6839     \tl_put_left:Ne \l__stex_notations_code_tl {
6840       \int_set:Nn \l_tmpa_int {\int_use:N \l_tmpa_int}
6841     }
6842     #6
6843   }
6844   \__stex_notations_parse_args:nnnnw #4 \__stex_notations_args_end:
6845 }
6846
6847 \cs_new_protected:Npn \__stex_notations_parse_args:nnnnw #1 #2 #3 #4 #5 \__stex_notations_a
6848   \tl_if_empty:nTF{#5}{
6849     \__stex_notations_add_last:nnnnn{#1}{#2}{#3}{#4}
6850   }{
6851     \__stex_notations_add_next:nnnnnn{#1}{#2}{#3}{#4}{#5}
6852   }
6853 }
6854
6855 \cs_new_protected:Nn \__stex_notations_add_next:nnnnnn {
6856   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#6}
6857   \__stex_notations_parse_args:nnnnw #5 \__stex_notations_args_end:
6858 }
6859
6860 \cs_new_protected:Nn \__stex_notations_add_last:nnnnn {
6861   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#5}
6862   \stex_debug:nn{sequences}{Executing~notation:~\meaning\l__stex_notations_code_tl}
6863   \l__stex_notations_after_tl
6864 }
6865
6866 \cs_new_protected:Nn \__stex_notations_add:nnnnn {
6867   \int_incr:N \l_tmpa_int
6868   \str_case:nn{#2}{
6869     i {
6870       \tl_put_right:Nn \l__stex_notations_code_tl {
6871         {\stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
6872       }

```



```

6873 }
6874 b {
6875   \tl_set:Nn \l__stex_notations_pre_tl {
6876     \cs_set_eq:NN \_stex_term_oma_or_omb:nn \_stex_term_omb:nn
6877   }
6878   \tl_put_right:Nn \l__stex_notations_code_tl {
6879     {\_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
6880   }
6881 }
6882 a {
6883   \tl_put_right:Nn \l__stex_notations_code_tl {
6884     {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
6885   }
6886 }
6887 B {
6888   \tl_set:Nn \l__stex_notations_pre_tl {
6889     \cs_set_eq:NN \_stex_term_oma_or_omb:nn \_stex_term_omb:nn
6890   }
6891   \tl_put_right:Nn \l__stex_notations_code_tl {
6892     {\_stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
6893   }
6894 }
6895 }
6896 }

```

(End of definition for \STEXInternalNotation. This function is documented on page 154.)

sfunction

\argsep

```

6897 \cs_new_protected:Npn \argsep #1 #2 {
6898   \__stex_notations_check_aB_arg:Nn\argsep{#1}
6899   \stex_pseudogroup_with:nn{\_stex_term_do_aB_clist:}{
6900     \tl_set:Nn \_stex_term_do_aB_clist: {
6901       \seq_use:Nn \l_stex_aB_args_seq {#2}
6902     }
6903     #1
6904   }
6905 }
6906
6907 \cs_new_protected:Nn \__stex_notations_check_aB_arg:Nn {
6908   \exp_args:Ne \cs_if_eq:NNF {\tl_head:n{#2}}
6909   \_stex_term_arg_aB:nnnnn {
6910     \msg_error:nnx{stex}{error/assocarg}{\tl_to_str:n{#1}}
6911   }
6912 }

```

(End of definition for \argsep. This function is documented on page 154.)

sfunction

\argmap

```

6913 \cs_new_protected:Npn \argmap #1 #2 #3 {
6914   \__stex_notations_check_aB_arg:Nn\argmap{#1}
6915   \stex_pseudogroup_with:nn{
6916     \_stex_term_do_aB_clist:

```

```

6917 \_stex_notations_map_cs:
6918 }{
6919 \cs_set:Npn \_stex_notations_map_cs: ##1 { #2 }
6920 \tl_set:Nn \_stex_term_do_aB_clist: {
6921 \seq_clear:N \l_tmpa_seq
6922 \seq_map_inline:Nn \l_stex_aB_args_seq {
6923 \tl_if_eq:nnTF{##1}{\ellipses}{
6924 \seq_put_right:Nn \l_tmpa_seq \ellipses
6925 }{
6926 \seq_put_right:Nx \l_tmpa_seq {
6927 \exp_after:wN \exp_not:n \exp_after:wN { \_stex_notations_map_cs: {##1} }
6928 }
6929 }
6930 }
6931 \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
6932 \seq_use:Nn \l_stex_aB_args_seq {#3}
6933 }
6934 #1
6935 }
6936 }

```

(End of definition for \argmap. This function is documented on page 154.)

sfunction

\argarraymap

```

6937 \int_new:N \l_stex_notations_clist_count_int
6938 \cs_new_protected:Npn \argarraymap #1 #2 #3 #4 {
6939 \_stex_notations_check_aB_arg:Nn\argarraymap{#1}
6940 \stex_pseudogroup_with:nn{
6941 \_stex_term_do_aB_clist:
6942 \_stex_notations_map_cs:
6943 }{
6944 \cs_set:Npn \_stex_notations_map_cs: ##1 { #3 }
6945 \int_set:Nn \l_stex_notations_clist_count_int {\exp_args:No\clist_count:n{\tl_to_str:n
6946 \tl_set:Nn \_stex_term_do_aB_clist: {
6947 \tl_clear:N \l_tmpa_tl
6948 \int_zero:N \l_tmpa_int
6949 \seq_map_inline:Nn \l_stex_aB_args_seq {
6950 \int_incr:N \l_tmpa_int
6951 \int_compare:nNnT \l_tmpa_int > \l_stex_notations_clist_count_int {
6952 \int_set:Nn \l_tmpa_int 1
6953 }
6954 \tl_put_right:Ne \l_tmpa_tl {
6955 \exp_after:wN \exp_not:n \exp_after:wN { \_stex_notations_map_cs: {##1} }
6956 \clist_item:nn{#4}\l_tmpa_int
6957 }
6958 }
6959 \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
6960 \begin{array}{#2}
6961 \l_tmpa_tl
6962 \end{array}
6963 }
6964 #1
6965 }
6966 }

```

(End of definition for `\argarraymap`. This function is documented on page 154.)
sfunction

33.1 Automated Bracketing

Variables for current (downwards) precedence, set of parentheses etc.:

```

6967 \int_new:N \l_stex_notation_downprec
6968 \tl_set:Nn \l__stex_notations_left_bracket_str (
6969 \tl_set:Nn \l__stex_notations_right_bracket_str )
6970 \bool_new:N \l_stex_brackets_dones_bool

\infprec
\neginfprec
6971 \tl_const:Ne \infprec {\int_use:N \c_max_int}
6972 \tl_const:Ne \neginfprec {-\int_use:N \c_max_int}
6973
6974 \int_set:Nn \l_stex_notation_downprec \infprec

```

(End of definition for `\infprec` and `\neginfprec`. These variables are documented on page 154.)

`_stex_maybe_brackets:nn`

```

6975 \cs_new_protected:Nn \_stex_maybe_brackets:nn {
6976   \bool_if:NTF \l_stex_brackets_dones_bool {
6977     \bool_set_false:N \l_stex_brackets_dones_bool
6978     #2
6979   } {
6980     \stex_debug:nn{brackets}{#1>\int_eval:n \l_stex_notation_downprec?}
6981     \int_compare:nNnTF { #1 } > \l_stex_notation_downprec {
6982       %\bool_if:NTF \l_stex_inarray_bool { #2 }{
6983         \dobrackets {
6984           #2
6985         }
6986       %}
6987     }{
6988       #2
6989     }
6990   }
6991 }

```

(End of definition for `_stex_maybe_brackets:nn`. This function is documented on page 154.)
sfunction

`\dobrackets`

```

6992 \cs_new_protected:Npn \dobrackets #1 {
6993   \group_begin:
6994     \bool_set_true:N \l_stex_brackets_dones_bool
6995     \mathopen{\tl_if_exist:NT\l_stex_current_symbol_uri\comp
6996       \l__stex_notations_left_bracket_str
6997     }
6998     #1
6999   \group_end:
7000   \mathclose{\tl_if_exist:NT\l_stex_current_symbol_uri\comp
7001     \l__stex_notations_right_bracket_str
7002   }
7003 }

```

(End of definition for \dobrackets. This function is documented on page 155.)

sfunction

\withbrackets

```
7004 \cs_new_protected:Npn \withbrackets #1 #2 #3 {  
7005   \stex_pseudogroup:nn{  
7006     \tl_set:Nn \l__stex_notations_left_bracket_str { #1 }  
7007     \tl_set:Nn \l__stex_notations_right_bracket_str { #2 }  
7008     #3  
7009   }{  
7010     \stex_pseudogroup_restore:N \l__stex_notations_left_bracket_str  
7011     \stex_pseudogroup_restore:N \l__stex_notations_right_bracket_str  
7012   }  
7013 }
```

(End of definition for \withbrackets. This function is documented on page 155.)

sfunction

\dowithbrackets

```
7014 \cs_new_protected:Npn \dowithbrackets #1 #2 #3 {  
7015   \withbrackets{#1}{#2}{\dobrackets{#3}}  
7016 }
```

(End of definition for \dowithbrackets. This function is documented on page 155.)

sfunction

Chapter 34

Structures

```
7017 <@@=stex_structures>

Keys:
7018 \stex_keys_define:nnnn{mathstructure}{
7019   \tl_clear:N \l_stex_current_this_tl
7020   \str_clear:N \l__stex_structures_name_str
7021 }{
7022   this .tl_set:N = \l_stex_current_this_tl ,
7023   unknown .code:n = {
7024     \str_if_empty:NTF \l_keys_key_str {
7025       \str_set:Ne \l__stex_structures_name_str {\l_keys_key_tl}
7026     }{
7027       \str_set_eq:NN \l__stex_structures_name_str \l_keys_key_str
7028     }
7029   }
7030 }{}
```

`mathstructure (env.)`

```
7031 \stex_new_stylable_env:nnnnnnn {mathstructure}{m 0{}}{
7032   \__stex_structures_begin:nn{#1}{#2}
7033   \stex_smsmode_do:
7034 }{
7035   \stex_structural_feature_module_end:
7036   \__stex_structures_do externals:
7037 }{}{}{}
7038
7039 \stex_sms_allow_env:n{mathstructure}
7040 \stex_deactivate_macro:Nn \mathstructure {module~environments}
7041 \stex_every_module:n {\stex_reactivate_macro:N \mathstructure}
```

Shared code for `mathstructure` and `extstructure`:

```
7042 \cs_new_protected:Nn \__stex_structures_begin:nn {
7043   \stex_keys_set:nn {mathstructure}{#2}
7044   \str_if_empty:NT \l__stex_structures_name_str {
7045     \str_set:Nn \l__stex_structures_name_str {#1}
7046   }
7047   \def\comp{\_comp}
7048
7049   \tl_set:Ne \l__stex_structures_struct_uri {
```

```

7050 \exp_args:No \stex_new_module_uri:n \l__stex_structures_name_str
7051 }
7052
7053 \exp_args:Nne \use:nn { \stex_add_symbol:nnnnnnN }
7054 { {#1}{\l__stex_structures_name_str}{0}{\defed}{\l__stex_structures_struct_uri}}
7055 {} \stex_invoke_structure:
7056 \str_set:Ne \l_stex_macroname_str {#1}
7057
7058 \exp_args:No \stex_structural_feature_module:nn
7059 {\l__stex_structures_name_str}{structure}
7060 }
7061
7062 \cs_new_protected:Nn \__stex_structures_doexternals: {
7063 \tl_set:Nn \l__stex_structures_replace_this_tl {####1}
7064 }
7065

```

extstructure (env.)

```

7066 \stex_new_stylable_env:nnnnnn {extstructure}{m O{} m}{
7067 \seq_clear:N \l__stex_structures_imports_seq
7068 \clist_map_inline:nn{#3}{
7069 \stex_get_mathstructure:n{##1}
7070 \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
7071 \stex_if_module_exists:nT{####1}{
7072 \seq_put_right:Nn \l__stex_structures_imports_seq{####1}
7073 }
7074 }
7075 }
7076 \__stex_structures_begin:nn{#1}{#2}
7077 \seq_map_inline:Nn \l__stex_structures_imports_seq{
7078 \stex_if_do_html:T {
7079 \hbox{\stex_annotate_invisible:nn
7080 {data-ftml-import={\stex_use_module_uri:n{##1}}}{}}
7081 }
7082 \stex_add_morphism:nonn
7083 {}{##1}{import}{}
7084 \stex_execute_in_module:e{
7085 \stex_activate_module:n {##1}
7086 }
7087 }
7088 \stex_smsmode_do:
7089 }{
7090 \stex_structural_feature_module_end:
7091 \__stex_structures_doexternals:
7092 }{}{}
7093
7094 \stex_sms_allow_env:n{extstructure}
7095 \stex_deactivate_macro:Nn \extstructure {module~environments}
7096 \stex_every_module:n {
7097 \stex_reactivate_macro:N \extstructure
7098 }
7099
7100 \stex_new_stylable_env:nnnnnn {extstructure*}{m}{
7101 \__stex_structures_new_extstruct_name:

```

```

7102 \seq_clear:N \l__stex_structures_imports_seq
7103 \stex_get_mathstructure:n{#1}
7104
7105 \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
7106   \stex_if_module_exists:nT{##1}{
7107     \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
7108   }
7109 }
7110
7111 \stex_module_uri_split_name:NNN \l__stex_structures_parent_uri \l__stex_structures_last_n
7112
7113 \stex_execute_in_module:e{
7114   \__stex_structures_extend_structure:nnnn{\l__stex_structures_parent_uri}{\l__stex_structures_last_n}
7115 }
7116
7117 \exp_args:No \__stex_structures_begin:nn\l__stex_structures_exstruct_name_str{
7118
7119   \seq_map_inline:Nn \l__stex_structures_imports_seq {
7120     \stex_if_do_html:T {
7121       \stex_annotate_invisible:nn
7122       {data-ftml-import= {\stex_use_module_uri:n{##1}}}{ }
7123     }
7124     %\stex_add_morphism:nonn
7125     % {}{##1}{import}{ }
7126     %\stex_execute_in_module:e{
7127     % \stex_activate_module:n {##1}
7128     %}
7129     \stex_activate_module:n {##1}
7130   }
7131
7132   \stex_smsmode_do:
7133 }{
7134   \prop_map_inline:cn{\_stex_symbol_macro:N \l_stex_current_module_uri }{
7135     \__stex_structures_check_def:nnnnnnn ##2
7136   }
7137   \stex_structural_feature_module_end:
7138   %\exp_args:Ne \stex_do_up_to_module:n {
7139   % \stex_activate_module:n {\exp_args:No \stex_new_module_uri:n {\l__stex_structures_exst
7140   %}
7141   \__stex_structures_do externals:
7142 }{ }{ }{ }
7143
7144 \stex_sms_allow_env:n{extstructure*}
7145 \exp_after:wN \stex_deactivate_macro:Nn
7146 \cs:w extstructure*\cs_end: {module~environments}
7147 \stex_every_module:n {
7148   \exp_after:wN \stex_reactivate_macro:N \cs:w extstructure*\cs_end:
7149 }
7150
7151 \cs_new_protected:Nn \__stex_structures_extend_structure:nnnn {
7152   \stex_debug:nn{ext}{Extending~\stex_use_module_uri:n {#1} / #2-by~\stex_use_module_uri:n
7153   \exp_args:Nne \tl_put_right:cn{\_stex_module_code_macro:n{#4}}{
7154     \stex_activate_module:n{#3}
7155   }

```

```

7156 \exp_args:Nne \prop_put:cnn{\_stex_morphisms_macro:n{#4}}{\stex_use_module_uri:n{#3}}{
7157   {}{#3}{extstruct}}{
7158 }
7159 \exp_args:Nne \use:nn {\_stex_structures_extend_ii:nnnn}{
7160   \exp_args:Nne \use:nn \_stex_structures_extend_i:nnnnnnnnNn {
7161     \prop_item:cn{\_stex_symbol_macro:n {#1} } { #2 }
7162     } { #3 }
7163   }{#1}{#2}
7164 }
7165
7166 \cs_new_protected:Nn \_stex_structures_extend_ii:nnnn {
7167   \prop_put:cnn{\_stex_symbol_macro:n {#3} } { #4 }{#2}
7168   \tl_set:ce{#1}{
7169     \stex_invoke_symbol:nnnnnnN {\stex_new_symbol_uri:nn {#3}{#4}}
7170     \use_none:nn #2
7171   }
7172 }
7173
7174 \cs_new:Nn \_stex_structures_extend_i:nnnnnnnnNn {
7175   {#1}{#{#1}{#2}{#3}{#4}{#5}{#6,#9}{#7}#8}
7176 }
7177
7178 \cs_new_protected:Nn \_stex_structures_check_def:nnnnnnnn {
7179   \tl_if_empty:nT{#5}{
7180     \msg_error:nnnn{stex}{error/needsdefiniens}{#2}{extstructure*}
7181   }
7182 }
7183
7184 \cs_new_protected:Nn \_stex_structures_new_extstruct_name: {
7185   \stex_do_up_to_module:n {
7186     \str_set:Ne \l__stex_structures_extname_count {\int_eval:n{\l__stex_structures_extname_
7187   }
7188   \str_set:Ne \l__stex_structures_exstruct_name_str {EXTSTRUCT_\l__stex_structures_extname_
7189 }
7190
7191 \stex_every_module:n{ \str_set:Nn \l__stex_structures_extname_count 0}

```

\this

```

7192 \bool_new:N \l_stex_in_this_bool
7193
7194 \cs_new_protected:Npn \stex_current_this: {
7195   { \bool_set_true:N \l_stex_allow_semantic_bool
7196     \tl_if_empty:NTF \l_stex_current_this_tl {} }{
7197     \tl_set:Nn \l_stex_current_full_tl {
7198       \exp_args:Ne \stex_use_symbol_uri:n {
7199         \exp_args:No \stex_new_symbol_uri:n \l__stex_structures_name_str
7200       }
7201     }
7202     \str_set_eq:NN \l_stex_current_display_tl \l__stex_structures_name_str
7203     \bool_set_true:N \l_stex_in_this_bool
7204     \maincomp{\l_stex_current_this_tl}
7205     \bool_set_false:N \l_stex_in_this_bool
7206   }
7207 }

```



```

7208 }
7209
7210 \let \this \stex_current_this:

```

(End of definition for \this. This function is documented on page 156.)

```

sfunction

```

\stex_get_mathstructure:n

```

7211 \cs_new_protected:Nn \stex_get_mathstructure:n {
7212   \stex_get_mathstructure_maybe:nTF{#1}{#{
7213     \msg_error:nnn{stex}{error/unknownstructure}{#1}
7214   }
7215 }
7216 \cs_new_protected:Npn \stex_get_mathstructure_maybe:nTF #1 #2 #3 {
7217   \tl_clear:N \l_stex_get_structure_module_uri
7218   \stex_get_symbol:nTF{#1}{
7219     \exp_args:No \tl_if_eq:NNTF \l_stex_get_symbol_invoke_cs \stex_invoke_structure: {
7220       \tl_set:Ne \l_stex_get_structure_module_uri { \exp_after:wN \__stex_structures_get:w
7221         #2
7222       }{
7223         #3
7224       }
7225     }{
7226       #3
7227     }
7228   }
7229
7230   \cs_new:Npn \__stex_structures_get:w #1 , #2 \stex_end: { #1 }

```

(End of definition for \stex_get_mathstructure:n. This function is documented on page 156.)

```

sfunction

```

\usestructure

```

7231 \cs_new_protected:Npn \usestructure #1 {
7232   \stex_if_smsmode:TF{
7233     \stex_get_mathstructure_maybe:nTF{ #1 }{
7234       \stex_debug:nn{usestructure}{Using~structure~#1:~\stex_use_module_uri:N \l_stex_get_str
7235       \__stex_structures_do_usestructure:
7236     }{
7237       \stex_debug:nn{usestructure}{Structure~#1~not~found~in~sms~mode}
7238     }
7239   }{
7240     \stex_get_mathstructure:n{ #1 }
7241     \stex_debug:nn{usestructure}{Using~structure~#1:~\stex_use_module_uri:N \l_stex_get_str
7242     \__stex_structures_do_usestructure:
7243   }
7244   \stex_smsmode_do:
7245 }
7246 \stex_sms_allow_escape:N \usestructure
7247
7248 \cs_new_protected:Nn \__stex_structures_do_usestructure: {
7249   \seq_clear:N \l__stex_structures_imports_seq
7250   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
7251     \stex_if_module_exists:nT{##1}{

```

```

7252     \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
7253   }
7254 }
7255 \seq_map_inline:Nn \l__stex_structures_imports_seq {
7256   \stex_if_do_html:T {
7257     \hbox{\stex_annotate_invisible:nn
7258       {data-ftml-usemodule=\stex_use_module_uri:n {##1}} {}}
7259   }
7260   \stex_activate_module:n {##1}
7261 }
7262 }

```

(End of definition for \usestructure. This function is documented on page 156.)
 sfunction

Chapter 35

Statements

7263 `<@@=stex_statements>`

Keys:

```
7264 \stex_keys_define:nnnn{statement}{  
7265   \str_clear:N \l_stex_key_name_str  
7266   \str_clear:N \l_stex_key_macroname_str  
7267   \clist_clear:N \l_stex_key_for_clist  
7268   \str_clear:N \l_stex_key_args_str  
7269   \tl_clear:N \l_stex_key_type_tl  
7270   \tl_clear:N \l_stex_key_def_tl  
7271   \tl_clear:N \l_stex_key_return_tl  
7272   \clist_clear:N \l_stex_key_argtypes_clist  
7273 }{  
7274   name      .str_set:N = \l_stex_key_name_str ,  
7275   for       .clist_set:N = \l_stex_key_for_clist ,  
7276   macro     .str_set:N = \l_stex_key_macroname_str ,  
7277   % start   .str_set:N = \l_stex_key_title_str , % TODO remove  
7278   type      .tl_set:N = \l_stex_key_type_tl ,  
7279   judgment  .code:n = {},  
7280   from      .code:n= {}, % TODO remove  
7281   to        .code:n={} % TODO remove  
7282 }{id,title,style,symargs}
```

`\stex_do_for_list:`

```
7283 \cs_new_protected:Npn \stex_do_for_list: {  
7284   \seq_clear:N \l_stex_fors_seq  
7285   \clist_map_inline:Nn \l_stex_key_for_clist {  
7286     \exp_args:Ne\stex_get_symbol:n{\tl_to_str:n{##1}}  
7287     \seq_put_right:No \l_stex_fors_seq  
7288     {\l_stex_get_symbol_uri}  
7289   }  
7290 }
```

(End of definition for \stex_do_for_list:. This function is documented on page 157.)

sfunction

`\stex_new_statement:nnn`

```
7291 \cs_new_protected:Nn \stex_new_statement:nnn {  
7292   \stex_new_stylable_env:nnnnnnn {#1}{0}}{
```

```

7293 \stex_keys_set:nn{statement}{##1}
7294 #3
7295
7296 \stex_if_smsmode:F {
7297   \exp_args:Nne \begin{stex_annotate_env}{
7298     \__stex_statements_html_keyvals:nn{#1}{false}
7299   }
7300   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
7301   \str_set_eq:NN \thisname \l_stex_key_name_str
7302   \clist_set_eq:NN \thisfor \l_stex_key_for_str
7303   \stex_if_html_backend:TF {
7304     \noindent
7305     \stex_annotate:nn{data-ftml-title={}, style:display={none}}{\_stex_annotate_force_b
7306     \tl_if_empty:NF \l_stex_key_title_tl{~}
7307     \l_stex_key_title_tl
7308   }}
7309   }
7310   \stex_style_apply:
7311 }
7312 \_stex_do_id:
7313 \stex_smsmode_do:
7314 }{
7315   \stex_if_smsmode:F {
7316     \stex_if_html_backend:F \stex_style_apply:
7317     \end{stex_annotate_env}
7318   }
7319 }{}{}{s}
7320 \stex_sms_allow_env:n{s#1}
7321
7322 \tl_if_empty:NF{#2}{\__stex_statements_make_macro:nnn{#1}{#2}{#3}}
7323 }
7324
7325 \cs_new_protected:Nn \__stex_statements_make_macro:nnn {
7326   \exp_after:wN \NewDocumentCommand \cs:w inline#2\cs_end: { 0{} m}{
7327     \group_begin:
7328     \stex_keys_set:nn{statement}{##1}
7329     #3
7330     \_stex_do_id:
7331     \stex_if_smsmode:F{
7332       \exp_args:Ne \stex_annotate:nn{\__stex_statements_html_keyvals:nn{#1}{true}}{
7333         \_stex_annotate_force_break:n{##2}
7334       }
7335     }
7336     \group_end:
7337     %\stex_if_smsmode:TF \stex_smsmode_do: \stex_ignore_spaces_and_pars:
7338     \stex_smsmode_do:
7339   }
7340   \exp_after:wN \stex_sms_allow_escape:N\cs:w inline#2\cs_end:
7341 }
7342
7343 \cs_new:Nn \__stex_statements_html_keyvals:nn {
7344   data-ftml-#1={},
7345   data-ftml-inline={#2},
7346   \seq_if_empty:NF \l_stex_fors_seq {,

```

```

7347     data-ftml-fors={\seq_map_indexed_function:Nn \l_stex_fors_seq \_stex_comma_sep_uri:nn }
7348   }
7349   \str_if_empty:NF \l_stex_key_id_str {,
7350     data-ftml-id={\l_stex_key_id_str}%{\stex_uri_use:N \l_stex_current_doc_uri ? \l_stex_ke
7351   }
7352   \clist_if_empty:NF \l_stex_key_style_clist {,
7353     data-ftml-styles={\l_stex_key_style_clist}
7354   }
7355 }
7356
7357 \cs_new:Nn \_stex_comma_sep_uri:nn {
7358   \int_compare:nNnF{#1}=1 {,}
7359   \stex_use_symbol_uri:n{#2}
7360 }

```

(End of definition for `\stex_new_statement:nnn`. This function is documented on page 157.)

sfunction

`_stex_statements_setup:n` sets up a statement that may declare a new symbol with the given `role` by processing the optional arguments, calling `\stex_symdecl_do:`, etc.

```

7361 \cs_new_protected:Nn \_stex_statements_setup:n {
7362   \str_if_empty:NF \l_stex_key_macroname_str {
7363     \str_if_empty:NT \l_stex_key_name_str {
7364       \str_set_eq:NN \l_stex_key_name_str \l_stex_key_macroname_str
7365     }
7366   }
7367   \stex_do_for_list:
7368   \tl_clear:N \l_stex_current_def_uri
7369
7370   \str_if_empty:NTF \l_stex_key_name_str \_stex_statements_setup_noname: {
7371     \_stex_statements_setup_named:n{#1}
7372   }
7373 }
7374
7375 \cs_new_protected:Nn \_stex_statements_setup_named:n {
7376   \_stex_statements_force_id:
7377   \seq_put_right:Ne \l_stex_fors_seq {
7378     \l_stex_current_module_uri {\l_stex_key_name_str}
7379   }
7380   \str_set_eq:NN \l_stex_macroname_str \l_stex_key_macroname_str
7381   \str_set:Nn \l_stex_key_role_str {#1}
7382   \stex_symdecl_do:
7383   \exp_args:Nne \use:nn {\stex_add_symbol:nnnnnnN}{
7384     {\l_stex_key_macroname_str}{\l_stex_key_name_str}
7385     {\int_use:N \l_stex_get_symbol_arity_int}
7386     {\l_stex_get_symbol_args_tl}
7387     {#1}{}}{\stex_invoke_symbol:
7388   }
7389   \stex_if_do_html:T \stex_symdecl_html:
7390   \tl_set:Ne \l_stex_current_def_uri {\exp_args:No \stex_new_symbol_uri:nn \l_stex_current_
7391   \stex_debug:nn{statement}{name:~\stex_use_symbol_uri:N \l_stex_current_def_uri}
7392 }
7393
7394 \cs_new_protected:Nn \_stex_statements_setup_noname: {

```

```

7395 \stex_debug:nn{statement}{no~name}
7396 \int_compare:nNnTF {\seq_count:N \l_stex_fors_seq} = 1 {
7397   \tl_set:Ne \l_stex_current_def_uri {\seq_item:Nn \l_stex_fors_seq 1}
7398   \stex_debug:nn{statement}{for:~\stex_use_symbol_uri:N \l_stex_current_def_uri}
7399 }{
7400   \stex_debug:nn{statement}{no~for}
7401 }
7402 }
7403
7404 \cs_new_protected:Nn \__stex_statements_force_id: {
7405   %\str_if_empty:NT \l_stex_key_id_str {
7406   %   \stex_ref_new_id:n{ }
7407   %   \str_set_eq:NN \l_stex_key_id_str \l__stex_refs_str
7408   %}
7409 }

```

(End of definition for __stex_statements_setup:n.)

__stex_statements_setup_def: Sets up all the macros allowed in definition-like environments:

```

7410 \cs_new_protected:Nn \__stex_statements_setup_def: {
7411   \stex_if_smsmode:F{
7412     \seq_map_inline:Nn \l_stex_fors_seq {
7413       \exp_args:Ne \stex_ref_new_sym_target:n{\stex_use_symbol_uri:n{##1}}
7414     }
7415   }
7416   \stex_reactivate_macro:N \definiendum
7417   \stex_reactivate_macro:N \defnotation
7418   \stex_reactivate_macro:N \definame
7419   \stex_reactivate_macro:N \Definame
7420   \stex_reactivate_macro:N \varbind
7421   \stex_reactivate_macro:N \definiens
7422 }

```

(End of definition for __stex_statements_setup_def:.)

sdefinition (env.)

```

7423 \stex_new_statement:nnn{definition}{def}{
7424   \__stex_statements_force_id:
7425   \__stex_statements_setup:n{ }
7426   \__stex_statements_setup_def:
7427 }

```

sassertion (env.)

```

7428 \stex_new_statement:nnn{assertion}{ass}{
7429   \__stex_statements_setup:n{assertion}
7430   \stex_if_smsmode:F{
7431     \seq_map_inline:Nn \l_stex_fors_seq {
7432       \exp_args:Ne \stex_ref_new_sym_target:n{\stex_use_symbol_uri:n{##1}}
7433     }
7434   }
7435   \stex_reactivate_macro:N \varbind
7436   \stex_reactivate_macro:N \conclusion
7437   \stex_reactivate_macro:N \premise
7438   \stex_reactivate_macro:N \definiendum

```

```

7439 \stex_reactivate_macro:N \defnotation
7440 \stex_reactivate_macro:N \definame
7441 \stex_reactivate_macro:N \Definame
7442 }

```

sexample (*env.*)

```

7443 \stex_new_statement:nnn{example}{ex}{
7444   \stex_if_smsmode:F {\_stex_statements_setup:n{example}}
7445 }

```

sparagraph (*env.*)

```

7446 \stex_new_statement:nnn{paragraph}{}{
7447   \clist_if_in:NnTF \l_stex_key_style_clist {symdoc}{
7448     \_stex_statements_force_id:
7449     \_stex_statements_setup:n{
7450       \_stex_statements_setup_def:
7451     }{
7452       \_stex_statements_setup:n{
7453     }
7454 }

```

definiens

```

7455 \NewDocumentCommand \definiens { 0{} m }{
7456   \group_begin:
7457   \_stex_definiens_impl:nn{#1}{#2}
7458   \group_end:
7459   \stex_smsmode_do:
7460 }
7461
7462 \cs_new_protected:Nn \_stex_definiens_impl:nn {
7463   \tl_if_empty:nF {#1} {
7464     \stex_get_symbol:n { #1 }
7465     \tl_set_eq:NN \l_stex_current_def_uri \l_stex_get_symbol_uri
7466   }
7467   \tl_if_empty:NT \l_stex_current_def_uri {
7468     \msg_error:nn{stex}{error/definiensfor}
7469   }
7470   \stex_debug:nn{definiens}{Checking-\stex_use_symbol_uri:N \l_stex_current_def_uri }
7471
7472   \exp_args:No \_stex_add_definiens:nn \l_stex_current_def_uri {#2}
7473 }
7474
7475 \stex_deactivate_macro:Nn \definiens {definition~environments}
7476 \stex_sms_allow_escape:N \definiens

```

(End of definition for definiens. This function is documented on page 157.)

sfunction

_stex_add_definiens:nn

```

7477 \cs_new_protected:Nn \_stex_add_definiens:nn {
7478   \stex_if_do_html:TF {
7479     \stex_annotate:nn{data-ftml-definiens={\stex_use_symbol_uri:n{#1}}}{
7480       #2 %\_stex_annotate_force_break:n{ #2 }
7481     }

```

```

7482   }{ \stex_if_smsmode:F{#2} }
7483   \stex_if_in_module:T {
7484     \exp_args:Ne \str_if_eq:noT {\stex_symbol_uri_module:n{#1}}\l_stex_current_module_uri{
7485       \__stex_statements_add_definiens:n{#1}
7486     }
7487   }
7488 }
7489
7490 \cs_new_protected:Nn \__stex_statements_add_definiens:n {
7491   \stex_debug:nn{definiens}{Adding~definiens~to~\stex_use_symbol_uri:n{#1}}
7492   \exp_args:Nne \prop_gput:cne{\exp_args:Ne \stex_symbol_macro:n {\stex_symbol_uri_module:
7493     {\stex_symbol_uri_name:n {#1}} {
7494       \exp_args:Nne \use:nn { \__stex_statements_definiens_inner:nnnnnnN }{ \exp_args:Nne
7495     }
7496   }
7497
7498   \cs_new:Nn \__stex_statements_definiens_inner:nnnnnnN {
7499     {#1}{#2}{#3}{#4}{defed}{#6}{#7}{#8}
7500   }

```

(End of definition for `\stex_add_definiens:nn`. This function is documented on page 157.)

sfunction

`\varbind`

```

7501 \NewDocumentCommand \varbind {m} {
7502   \clist_map_inline:nn {#1} {
7503     \stex_get_var:n {##1}
7504     \stex_if_do_html:T {
7505       \stex_annotate_invisible:nn {data-ftml-bind=\l_stex_get_variable_str}{ }
7506     }
7507   }
7508 }
7509 \stex_deactivate_macro:Nn \varbind {definition~or~assertion~environments}

```

(End of definition for `\varbind`. This function is documented on page 157.)

sfunction

`\conclusion`

```

7510 \NewDocumentCommand \conclusion { 0{ } m } {
7511   \group_begin:
7512   \tl_if_empty:nF {#1} {
7513     \stex_get_symbol:n { #1 }
7514     \tl_set_eq:NN \l_stex_current_def_uri \l_stex_get_symbol_uri
7515   }
7516   \tl_if_empty:NT \l_stex_current_def_uri {
7517     \msg_error:nn{stex}{error/conclusionfor}
7518   }
7519   \stex_annotate:nn{ data-ftml-conclusion={\stex_use_symbol_uri:N \l_stex_current_def_uri}}
7520   #2 %\stex_annotate_force_break:n{ #2 }
7521 }
7522 \group_end:
7523 }
7524 \stex_deactivate_macro:Nn \conclusion {assertion~environments}

```


(End of definition for `\conclusion`. This function is documented on page 157.)
`sfunction`

`\premise`

```

7525 \NewDocumentCommand \premise {0{} m} {
7526   \tl_if_empty:nF {#1} {
7527     \stex_debug:nn{Here:}{Variable~#1}
7528     \exp_args:Nne\use:nn{\vardef}{v#1}[name=#1]{#1}}
7529   }
7530   \stex_annotate:nn{data-ftml-premise={#1}}{#2}
7531 }
7532 \stex_deactivate_macro:Nn \premise {assertion~environments}

```

(End of definition for `\premise`. This function is documented on page 157.)
`sfunction`

Chapter 36

Proofs

```
7533 <@@=stex_proof>
      Keys:
7534
7535 \stex_keys_define:nnnn{ spfcommon }{
7536   \tl_clear:N \l_stex_key_for_clist
7537   \tl_clear:N \l_stex_key_term_tl
7538   \tl_clear:N \l_stex_key_method_tl
7539   \tl_clear:N \l_stex_key_just_tl
7540 }{
7541   for      .clist_set:N = \l_stex_key_for_clist ,
7542   term     .tl_set:N    = \l_stex_key_term_tl,
7543   method   .tl_set:N    = \l_stex_key_method_tl,
7544   just     .tl_set:N    = \l_stex_key_just_tl,
7545 }{id,style,title}
7546
7547 \bool_set_false:N \l_stex_key_showname_bool
7548
7549 \stex_keys_define:nnnn{ spfnamed }{
7550   \str_clear:N \l_stex_key_name_str
7551   \str_clear:N \l_stex_key_numname_str
7552   \bool_set_false:N \l_stex_key_showname_bool
7553 }{
7554   name      .str_set_x:N = \l_stex_key_name_str,
7555   numname   .str_set_x:N = \l_stex_key_numname_str,
7556   showname  .bool_set:N  = \l_stex_key_showname_bool
7557 }{spfcommon}
7558
7559 \cs_new_protected:Nn \__stex_proof_do_spf_name: {
7560   \str_if_empty:NTF \l_stex_key_numname_str {
7561     \str_if_empty:NTF \l_stex_key_name_str {
7562       \bool_set_false:N \l_stex_key_showname_bool
7563     } {
7564       \exp_args:Nne\use:nn{\vardef}{\v\l_stex_key_name_str}[name=\l_stex_key_name_str]{\l_s
7565     }
7566   }{
7567     \bool_set_false:N \l_stex_key_showname_bool
7568     \__stex_proof_number_as_string:N \l__stex_proof_counter_str
7569     \exp_args:Nne \use:nn {\vardef}{\{\l_stex_key_numname_str}[name=\l__stex_proof_counter_s
```

```

7570 }
7571 }
7572
7573 \cs_new_protected:Nn \__stex_proof_do_spf_name_i: {
7574   \str_if_empty:NTF \l__stex_proof_key_numname_str {
7575     %\str_if_empty:NF \l__stex_proof_key_name_str {
7576       \exp_args:Nne\use:nn{\vardef}{\v\l__stex_proof_key_name_str}[name=\l__stex_proof_key_
7577     %}
7578   }{
7579     \exp_args:Nne \use:nn {\vardef}{\{\l__stex_proof_key_numname_str}[name=\l__stex_proof_co
7580   }
7581 }
7582
7583 \cs_new_protected:Nn \__stex_proof_do_spf_varname: {
7584   \bool_if:NT \l_stex_key_showname_bool {
7585     ~($\csname v\l_stex_key_name_str\endcsname$)~
7586   }
7587 }
7588
7589 \stex_keys_define:nnnn{ spftop }{
7590   \tl_set_eq:NN \l_stex_key_proofend_tl \__stex_proof_proof_box_tl
7591   \bool_set_false:N \l_stex_key_hide_bool
7592   \tl_clear:N \l_stex_key_continues_tl
7593   \tl_clear:N \l_stex_key_functions_tl
7594   \tl_clear:N \l_stex_key_from_tl
7595 }{
7596   proofend      .tl_set:N      = \l_stex_key_proofend_tl,
7597   hide          .code:n        = {
7598     \str_if_eq:eeT{#1}{true}{
7599       \bool_set_true:N \l_stex_key_hide_bool
7600     }
7601   },
7602   continues     .tl_set:N      = \l_stex_key_continues_tl, % <- unused
7603   functions     .tl_set:N      = \l_stex_key_functions_tl, % <- unused
7604   from          .tl_set:N      = \l_stex_key_from_tl % <- unused
7605 }{spfcommon}
7606
7607 \stex_keys_define:nnnn{ subproof }{ }{ }{spftop,spfnamed}
7608
7609 \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.

```

7610 \intarray_new:Nn\l__stex_proof_counter_intarray{50}
7611
7612 \cs_new_protected:Npn \__stex_proof_insert_number: {
7613   \int_set:Nn \l_tmpa_int {1}
7614   \bool_while_do:nn {

```

```

7615     \int_compare_p:nNn {
7616         \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7617     } > 0
7618 }{
7619     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .
7620     \int_incr:N \l_tmpa_int
7621 }
7622 }
7623 \cs_new_protected:Nn \__stex_proof_number_as_string:N {
7624     \str_clear:N #1
7625     \int_set:Nn \l_tmpa_int {1}
7626     \bool_while_do:nn {
7627         \int_compare_p:nNn {
7628             \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7629         } > 0
7630     }{
7631         \str_put_right:Ne #1 {\intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .}
7632         \int_incr:N \l_tmpa_int
7633     }
7634 }
7635
7636 \cs_new_protected:Npn \__stex_proof_inc_counter: {
7637     \int_set:Nn \l_tmpa_int {1}
7638     \bool_while_do:nn {
7639         \int_compare_p:nNn {
7640             \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7641         } > 0
7642     }{
7643         \int_incr:N \l_tmpa_int
7644     }
7645     \int_compare:nNnF \l_tmpa_int = 1 {
7646         \int_decr:N \l_tmpa_int
7647     }
7648     \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int {
7649         \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int + 1
7650     }
7651 }
7652
7653 \cs_new_protected:Npn \__stex_proof_add_counter: {
7654     \int_set:Nn \l_tmpa_int {1}
7655     \bool_while_do:nn {
7656         \int_compare_p:nNn {
7657             \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7658         } > 0
7659     }{
7660         \int_incr:N \l_tmpa_int
7661     }
7662     \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 1 }
7663 }
7664
7665 \cs_new_protected:Npn \__stex_proof_remove_counter: {
7666     \int_set:Nn \l_tmpa_int {1}
7667     \bool_while_do:nn {
7668         \int_compare_p:nNn {

```

```

7669     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7670   } > 0
7671   ){
7672     \int_incr:N \l_tmpa_int
7673   }
7674   \int_decr:N \l_tmpa_int
7675   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 0 }
7676 }

sproof (env.)

7677 \bool_set_false:N \l__stex_proof_in_spfblock_bool
7678
7679 \cs_new_protected:Nn \__stex_proof_begin_proof:nn {\par
7680   \intarray_gzero:N \l__stex_proof_counter_intarray
7681   \intarray_gset:Nnn \l__stex_proof_counter_intarray 1 1
7682   \stex_keys_set:nn{spf}top}{#1}
7683   \stex_do_for_list:
7684   \stex_if_do_html:T {
7685     \__stex_proof_html_env_proof:
7686   }
7687   \seq_map_inline:Nn \l_stex_fors_seq {
7688     \stex_debug:nn{definiens}{Adding~definiens~to~##1}
7689     \stex_if_do_html:TF{
7690       \STEXinvisible{\hbox{\stex_add_definiens:nn {##1}{proven}}}}
7691     ){
7692       \stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7693     }
7694   }
7695   \stex_if_do_html:F\stex_style_apply:
7696   %\stex_do_id:
7697   \stex_reactivate_macro:N \subproof
7698   \stex_reactivate_macro:N \spfstep
7699   \stex_reactivate_macro:N \conclude
7700   \stex_reactivate_macro:N \assumption
7701   \stex_reactivate_macro:N \eqstep
7702   \stex_reactivate_macro:N \yield
7703   \stex_reactivate_macro:N \spfescape
7704   \stex_reactivate_macro:N \spfblock
7705   \stex_reactivate_macro:N \spfjust
7706   \stex_reactivate_macro:N \spfby
7707   \stex_reactivate_macro:N \spfarg
7708   \noindent\stex_annotate:nn{data-ftml-title={}}{\stex_annotate_force_break:n{#2}}\par
7709   \stex_if_do_html:TF{
7710     \use:c{stex_annotate_env}{data-ftml-proofbody={}}
7711     \stex_annotate_force_break:n{}}
7712   ){
7713     \bool_if:NT \l_stex_key_hide_bool{\setbox0\vbox\bgroup}
7714   }
7715 }

7716
7717 \cs_new_protected:Nn \__stex_proof_html_env_proof: {
7718   \exp_args:Nne \use:c{stex_annotate_env}{
7719     data-ftml-proof={}}
7720   \seq_if_empty:NF \l_stex_fors_seq {,

```

```

7721     data-ftml-fors={\seq_map_indexed_function:NN \l_stex_fors_seq \_stex_comma_sep_uri:nn
7722   }
7723   \bool_if:NT \l_stex_key_hide_bool {,
7724     data-ftml-proofhide=true
7725   }
7726 }
7727 \__stex_proof_html:
7728 }
7729
7730 \cs_new_protected:Nn \__stex_proof_html_env_subproof: {
7731   \str_if_empty:NF \l_stex_key_numname_str {
7732     \__stex_proof_number_as_string:N \l__stex_proof_counter_str
7733   }
7734   \exp_args:Nne \use:c{stex_annotate_env}{
7735     data-ftml-subproof={ }
7736     \seq_if_empty:NF \l_stex_fors_seq {,
7737       data-ftml-fors={\seq_map_indexed_function:NN \l_stex_fors_seq \_stex_comma_sep_uri:nn
7738     }
7739     \bool_if:NT \l_stex_key_hide_bool {,
7740       data-ftml-proofhide=true
7741     }
7742     \str_if_empty:NTF \l_stex_key_numname_str {
7743       \str_if_empty:NF \l_stex_key_name_str {,
7744         data-ftml-stepname={\l_stex_key_name_str}
7745       }
7746     }{
7747       , data-ftml-stepname={\l__stex_proof_counter_str}
7748     }
7749   }
7750   \__stex_proof_html:
7751 }
7752
7753 \cs_new_protected:Nn \__stex_proof_html: {
7754   \stex_annotate_force_break:n{ }
7755   \tl_set:N\l_tmpa_tl {\l_stex_key_term_tl\l_stex_key_method_tl\l_stex_key_just_tl}
7756   \tl_if_empty:NF\l_tmpa_tl{
7757     \stex_annotate_invisible:n{\hbox{
7758       \tl_if_empty:NF \l_stex_key_term_tl {
7759         $\stex_annotate:nn{data-ftml-proofterm={}}{\l_stex_key_term_tl}$
7760       }
7761       \tl_if_empty:NF \l_stex_key_just_tl {
7762         $\stex_annotate:nn{data-ftml-spfjust={}}{\l_stex_key_just_tl}$
7763       }
7764       \tl_if_empty:NF \l_stex_key_method_tl {
7765         \stex_annotate:nn{data-ftml-proofmethod={}}{\l_stex_key_method_tl}
7766       }
7767     }}}
7768 }
7769 }
7770
7771 \cs_new_protected:Nn \__stex_proof_start_comment: {
7772   \bool_if:NT \l__stex_proof_in_spfblock_bool {
7773     \par
7774     \group_begin:\stexcommentfont

```

```

7775 }
7776 }
7777 \cs_new_protected:Nn \__stex_proof_end_comment: {
7778   \bool_if:NT \l__stex_proof_in_spfblock_bool {
7779     \par
7780     \group_end:
7781   }
7782 }
7783 \cs_new_protected:Npn \spfescape #1{
7784   \__stex_proof_end_comment: #1 \__stex_proof_start_comment:
7785 }
7786 \stex_deactivate_macro:Nn \spfescape {sproof~environments}
7787
7788 \stex_new_stylable_env:nnnnnnn{proof}{0}{ m}{
7789   \__stex_proof_begin_proof:nn{#1}{#2}
7790   \bool_set_true:N\l__stex_proof_in_spfblock_bool
7791   %\__stex_proof_start_list:n{}
7792   \__stex_proof_start_comment:
7793 }{
7794   \stex_if_do_html:TF{
7795     %\__stex_proof_end_list:
7796     \use:c{endstex_annotate_env}\use:c{endstex_annotate_env}
7797   }\stex_style_apply:
7798 }{
7799   \emph{\sproofautorefname :}~
7800 }{
7801   \sproofend
7802 }{s}
7803
7804 \AddToHook{env/sproof/end}{
7805   \__stex_proof_end_comment:
7806   \stex_if_do_html:F{\bool_if:NT \l_stex_key_hide_bool\egroup}
7807 }
7808
7809
7810 \stex_new_stylable_env:nnnnnnn{proof*}{0}{}{
7811   \__stex_proof_begin_proof:nn{#1}{}
7812   \bool_set_false:N\l__stex_proof_in_spfblock_bool
7813 }{
7814   \stex_if_do_html:TF{\use:c{endstex_annotate_env}\use:c{endstex_annotate_env}}\stex_style_
7815 }{
7816   \emph{Proof:}~
7817 }{
7818   \sproofend
7819 }{s}
7820
7821 \cs_new_protected:Nn \__stex_proof_start_list:n {
7822   %\par
7823   \group_begin:\list{}{
7824     \setlength\topsep{0pt}
7825     \setlength\parsep{0pt}
7826     \setlength\rightmargin{0pt}
7827   }\item[#1]
7828 }

```

```

7829
7830 \cs_new_protected:Nn \__stex_proof_end_list: {
7831   %\par
7832   \endlist\group_end:
7833 }

```

subproof (*env*.)

```

7834 \str_set_eq:NN \subproofautorefname \spfstepautorefname
7835
7836 \cs_new_protected:Nn \__stex_proof_do_after_subproof:N {
7837   \bgroup
7838   \expandafter \__stex_proof_do_after_subproof_i:nn #1
7839   \egroup
7840   \__stex_proof_do_spf_name_i:
7841 }
7842 \cs_new_protected:Nn \__stex_proof_do_after_subproof_i:nn {
7843   \exp_args:Nno \stex_keys_set:nn{subproof}{#1}
7844   \str_gset_eq:NN \l__stex_proof_key_name_str \l_stex_key_name_str
7845   \str_gset_eq:NN \l__stex_proof_key_numname_str \l_stex_key_numname_str
7846   \bool_gset_eq:NN \l__stex_proof_key_showname_bool \l_stex_key_showname_bool
7847   \str_gset:Nn \l__stex_proof_counter_str {#2}
7848 }
7849 \cs_new_protected:Npn \__stex_proof_set_after_subproof:n {
7850   \str_if_empty:NTF \l_stex_key_numname_str {
7851     \str_if_empty:NTF \l_stex_key_name_str \use_none:n \__stex_proof_set_after_subproof_i:n
7852   }\__stex_proof_set_after_subproof_i:n
7853 }
7854 \cs_new_protected:Npn \__stex_proof_set_after_subproof_with_number:n {
7855   \str_if_empty:NTF \l_stex_key_numname_str {
7856     \str_if_empty:NTF \l_stex_key_name_str \use_none:n \__stex_proof_set_after_subproof_wit
7857   }\__stex_proof_set_after_subproof_with_number_i:n
7858 }
7859 \cs_new_protected:Nn \__stex_proof_set_after_subproof_i:n {
7860   \tl_gset:cn{subproof_arg_ \int_use:N \currentgrouplevel}{#{1}{}}
7861
7862   \__stex_proof_set_aftergroup: % \bool_if:NTF \l__stex_proof_in_spfblock_bool \__stex_proo
7863 }
7864 \cs_new_protected:Nn \__stex_proof_set_after_subproof_with_number_i:n {
7865   \__stex_proof_number_as_string:N \l__stex_proof_counter_str
7866   \tl_gset:ce{subproof_arg_ \int_use:N \currentgrouplevel}{\exp_not:n{#1}}{\l__stex_proof_
7867   \__stex_proof_set_aftergroup:
7868 }
7869 \cs_new_protected:Nn \__stex_proof_set_aftergroup: {
7870   \aftergroup \__stex_proof_do_after_subproof:N
7871   \expandafter \aftergroup \csname subproof_arg_ \int_use:N \currentgrouplevel\endcsname
7872 }
7873
7874 \stex_new_stylable_env:nnnnnn{subproof}{s O{} m}{
7875   \stex_keys_set:nn{subproof}{#2}
7876   \stex_do_for_list:
7877   \stex_if_do_html:T {
7878     \__stex_proof_html_env_subproof:
7879   }
7880   \seq_map_inline:Nn \l_stex_fors_seq {

```



```

7881 \stex_debug:nn{definiens}{Adding-definiens-to-##1}
7882 \stex_if_do_html:TF{
7883   \STEXinvisible{\hbox{\_stex_add_definiens:nn {##1}{proven}}}}
7884 }{
7885   \_stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7886 }
7887 }
7888
7889 \IfBooleanTF #1 {
7890   \bool_set_false:N \l__stex_proof_in_spfblock_bool
7891   \__stex_proof_set_after_subproof:n{#2}
7892   \stex_if_do_html:F \stex_style_apply:
7893   \str_if_empty:NF \l_stex_key_id_str {
7894     \__stex_proof_number_as_string:N \@currentlabel
7895     %\str_set:Ne \@currentHref{subproof.\@currentlabel}
7896     %\_stex_do_id:
7897   }
7898
7899
7900   \stex_annotate:nn{data-ftml-title={}}{#3}
7901 }{
7902   \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7903
7904     \__stex_proof_set_after_subproof_with_number:n{#2}
7905
7906     %\str_if_empty:NF \l_stex_key_id_str {
7907       %\_stex_proof_number_as_string:N \@currentlabel
7908       %\str_set:Ne \@currentHref{subproof.\@currentlabel}
7909       %\_stex_do_id:
7910     %}
7911     \use:c{stex_annotate_env}{data-ftml-title={}} \__stex_proof_start_list:n{\__stex_proo
7912       \__stex_proof_add_counter:
7913       \stex_if_do_html:F \stex_style_apply:
7914     }{
7915       \noindent \stex_annotate:nn{data-ftml-title={}}{#3} \par
7916       \stex_if_do_html:F \stex_style_apply:
7917
7918       \__stex_proof_set_after_subproof:n{#2}
7919
7920       %\_stex_do_id:
7921     }
7922   }
7923   \stex_if_do_html:TF{
7924     \use:c{stex_annotate_env}{data-ftml-proofbody={}}
7925     \_stex_annotate_force_break:n{ }
7926   }{\bool_if:NT \l_stex_key_hide_bool{\setbox0\vbox\bgroup}}
7927   \__stex_proof_start_comment:
7928 }{
7929   \bool_if:NT\l__stex_proof_in_spfblock_bool {
7930     \__stex_proof_remove_counter:
7931     \__stex_proof_inc_counter:
7932   }
7933
7934   \stex_if_do_html:TF{

```

```

7935     \use:c{endstex_annotate_env} %proofbody
7936     \bool_if:NT\l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
7937     \use:c{endstex_annotate_env} % subproof
7938   }{
7939     \stex_style_apply:
7940     \bool_if:NT \l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
7941   }
7942 }{}{}{}
7943
7944 \AddToHook{env/subproof/before}{
7945   \__stex_proof_end_comment:
7946 }
7947
7948 \AddToHook{env/subproof/after}{
7949   \__stex_proof_start_comment:
7950 }
7951
7952 \AddToHook{env/subproof/end}{
7953   \__stex_proof_end_comment:
7954   \stex_if_do_html:F{\bool_if:NT \l_stex_key_hide_bool\egroup}
7955 }
7956
7957 \stex_deactivate_macro:Nn \subproof {sproof~environments}
7958

```

spfsketchenv (env.) TODO:

```

7959 \newenvironment{spfsketchenv}{}{}

```

\spfsketch

```

7960 \stex_new_stylable_cmd:nnnn{spfsketch}{0}{ m}{\par
7961   \begin{spfsketchenv}
7962   \stex_keys_set:nn{spftop}{#1}
7963   \_stex_do_for_list:
7964   \_stex_do_id:
7965   \par
7966   \exp_args:Nne \use:c{stex_annotate_env}{
7967     data-ftml-proofsketch={
7968       \seq_if_empty:NF \l_stex_fors_seq {
7969         \seq_map_function:NN \l_stex_fors_seq \stex_use_symbol_uri:N
7970       }
7971     }
7972   }
7973   \stex_style_apply:
7974   #2\par
7975   \use:c{endstex_annotate_env}
7976 \end{spfsketchenv}
7977 }{
7978   \noindent\emph{\spfsketchenvautorefname :}~
7979 }

```

(End of definition for \spfsketch. This function is documented on page 158.)

sfunction

Keys for proof step macros:

```

7980 \stex_keys_define:nnnn { spfsteps } {

```

```

7981   \%str_clear:N \l_stex_key_name_str
7982 }{
7983   \%name          .str_set_x:N = \l_stex_key_name_str
7984 }{spfcommon,spfnamed}

Common setup for all the proof step macros:

7985 \cs_new_protected:Nn \__stex_proof_make_step_macro:Nnnnn {
7986   \NewDocumentCommand #1 {s O{}} +m {
7987     \__stex_proof_end_comment:
7988     \stex_keys_set:nn{spfsteps}{##2}
7989     \%str_if_empty:NF \l_stex_key_name_str {
7990       \% \exp_args:Nne\use:nn{\vardef}{\v\l_stex_key_name_str}[name=\l_stex_key_name_str]{\l_
7991       \%}
7992     \__stex_proof_do_spf_name:
7993
7994     \spfstepenv
7995     \str_if_empty:NF \l_stex_key_id_str {
7996       \%__stex_proof_number_as_string:N \@currentlabel
7997       \%str_set:Ne \@currentHref{spfstep.\@currentlabel}
7998       \% \stex_do_id:
7999     }
8000
8001     \bool_if:NTF \l__stex_proof_in_spfblock_bool {
8002       \IfBooleanTF ##1 {
8003         \__stex_proof_step_html_i:nn{#2}{##3}
8004       }{
8005         \__stex_proof_step_html:nn{#2}{\__stex_proof_start_list:n{#3} ##3 \__stex_proof_end
8006         #5
8007       }
8008       \endspfstepenv
8009       \__stex_proof_start_comment:
8010     }{
8011       \__stex_proof_step_html_i:nn{#2}{##3}
8012       \endspfstepenv
8013     }
8014   }
8015   \stex_deactivate_macro:Nn #1 {sproof~environments}
8016 }

8017
8018 \cs_new_protected:Nn \__stex_proof_step_html:nn {
8019   \str_if_empty:NF \l_stex_key_numname_str {
8020     \__stex_proof_number_as_string:N \l__stex_proof_counter_str
8021   }
8022   \stex_if_do_html:TF{
8023     \%group_begin:
8024     \exp_args:Nne \use:c{stex_annotate_env}{
8025       data-ftml-spf#1={
8026         \seq_if_empty:NF \l_stex_fors_seq {
8027           \seq_map_function:NN \l_stex_fors_seq \stex_use_symbol_uri:N
8028         }
8029       }
8030     \str_if_empty:NTF \l_stex_key_numname_str {
8031       \str_if_empty:NF \l_stex_key_name_str {,
8032         data-ftml-stepname={\l_stex_key_name_str}
8033       }

```

```

8034     }{
8035     , data-ftml-stepname={\l__stex_proof_counter_str}
8036     }
8037   }
8038   \__stex_proof_html:
8039   #2
8040   \par
8041   \use:c{endstex_annotate_env}
8042   %\group_end:
8043   }{ #2 }
8044 }
8045 \cs_new_protected:Nn \__stex_proof_step_html_i:nn {
8046   \stex_if_do_html:TF{
8047     \noindent
8048     %\group_begin:
8049     \exp_args:Ne \stex_annotate:nn{
8050       data-ftml-spf#1={
8051         \seq_if_empty:NF \l_stex_fors_seq {
8052           \seq_map_function:NN \l_stex_fors_seq \stex_use_symbol_uri:N
8053         }
8054       }
8055       \str_if_empty:NTF \l_stex_key_numname_str {
8056         \str_if_empty:NF \l_stex_key_name_str {,
8057           data-ftml-stepname={\l_stex_key_name_str}
8058         }
8059       }{
8060         , data-ftml-stepname={\l__stex_proof_counter_str}
8061       }
8062     }{
8063       \__stex_proof_html:
8064       #2
8065     }
8066     %\group_end:
8067   }{ #2 }
8068 }

```

spfstepenv (env.) TODO:

```

8069 \newenvironment{spfstepenv}{
8070   \str_set_eq:NN \spfstepenvautorefname \spfstepautorefname
8071 }{}

```

spfblock (env.)

```

8072 \NewDocumentEnvironment{spfblock}{}{
8073   \bool_set_false:N \l__stex_proof_in_spfblock_bool
8074 }{
8075   \aftergroup\__stex_proof_start_comment:
8076 }
8077
8078 \stex_deactivate_macro:Nn \spfblock {sproof~environments}
8079 \AddToHook{env/spfblock/before}{
8080   \__stex_proof_end_comment:
8081 }

```

\spfstep

```

8082 \__stex_proof_make_step_macro:Nnnnn \spfststep {step} {\__stex_proof_insert_number:\__stex_pr
(End of definition for \spfststep. This function is documented on page 158.)
sfunction

\assumption
8083 \__stex_proof_make_step_macro:Nnnnn \assumption {assumption} {\__stex_proof_insert_number:\
(End of definition for \assumption. This function is documented on page 158.)
sfunction

\conclude
8084 \__stex_proof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
(End of definition for \conclude. This function is documented on page 158.)
sfunction

\eqstep
8085 \NewDocumentCommand \eqstep {s m}{
8086 \__stex_proof_end_comment:
8087 \spfststepenv
8088
8089 \bool_if:NTF \l__stex_proof_in_spfblock_bool {
8090 \IfBooleanTF #1 {
8091 \__stex_proof_step_html_i:nn{eqstep}{\$= #2$}
8092 }{
8093 \__stex_proof_step_html:nn{eqstep}{\__stex_proof_start_list:n{\$=$} \$#2$ \__stex_proof
8094 }
8095 \endspfststepenv
8096 \__stex_proof_start_comment:
8097 }{
8098 \__stex_proof_step_html_i:nn{eqstep}{\$= #2$}
8099 \endspfststepenv
8100 }
8101 }
8102 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
(End of definition for \eqstep. This function is documented on page 158.)
sfunction

\yield
8103 \NewDocumentCommand \yield {+m}{
8104 \stex_annotate:nn{data-ftml-proofterm={}}{ #1 }
8105 }
8106 \stex_deactivate_macro:Nn \yield {sproof~environments}
(End of definition for \yield. This function is documented on page 158.)
sfunction

\spfjust
8107 \newcommand\spfjust[1]{
8108 \stex_annotate:nn{data-ftml-spfjust={}}{ #1 }
8109 }
8110 \stex_deactivate_macro:Nn \spfjust {sproof~environments}

```

(End of definition for `\spfjust`. This function is documented on page 158.)

sfunction

`\spfby`

```
8111 \newcommand\spfby[1]{
8112   \stex_annotate:nn{data-ftml-proofmethod={}}{ #1 }
8113 }
8114 \stex_deactivate_macro:Nn \spfby {sproof~environments}
```

(End of definition for `\spfby`. This function is documented on page 158.)

sfunction

`\spfarg`

```
8115 \newcommand\spfarg[2][]{
8116   \stex_annotate:nn{data-ftml-spfarg={#1}}{ #2 }
8117 }
8118 \stex_deactivate_macro:Nn \spfarg {sproof~environments}
```

(End of definition for `\spfarg`. This function is documented on page 159.)

sfunction

`\sproofend`

```
8119 \tl_set:Nn \__stex_proof_proof_box_tl {
8120   \ltx@ifpackageloaded{amssymb}{\square$}{
8121     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
8122   }
8123 }
8124
8125 \tl_set:Nn \sproofend {
8126   \tl_if_empty:NF \l_stex_key_proofend_tl {
8127     \hfil\null\nobreak\hfill\l_stex_key_proofend_tl\par\smallskip
8128   }
8129 }
```

(End of definition for `\sproofend`. This function is documented on page 159.)

sfunction

`\stexcommentfont`

```
8130 \cs_new_protected:Npn \stexcommentfont {
8131   \small\itshape
8132 }
```

(End of definition for `\stexcommentfont`. This function is documented on page 159.)

sfunction

Chapter 37

Metatheory

```
8133 <@@=stex_meta>

      Setup:
8134 \group_begin:
8135 \cs_set:Npn \__stex_modules_persist_module: {}
8136 \cs_set:Npn \stex_check_term:nn #1 #2 {}
8137 \cs_set:Npn \stex_sref_do_aux:n #1 { #1 }
8138 \bool_set_false:N \c_stex_check_terms_bool
8139 \bool_set_false:N \l__stex_annotate_do_output_bool
8140 \stex_set_meta_archive:
8141 \tl_set:Nx \l_stex_current_document_uri {
8142   {\tl_to_str:n{FTML/meta}}{}
8143   {\tl_to_str:n{Metatheory}}{\tl_to_str:n{en}}{}
8144 }
8145 \stex_module_setup_top_nosig:n{Metatheory}
8146 \g_stex_every_module_tl

8147
8148 \symdef{typeof}[name=type~of,args=1]{\maincomp{\mathrm{tp}}\dobrackets{#1}}
8149
8150 \symdef{commented}[args=2]{#2 \; (#1)}
8151 \notation{commented}[inv]{#2}
8152
8153 \symdef{of~type}[args=ii,invisible]{#1}
8154 \notation{of~type}[colon]{#1 \mathbin{\comp{:}} #2}
8155
8156 \symdef{apply}[args=ia,prec=0;\infpref x\infpref]{#1\mathopen{\comp{}} #2 \mathclose{\comp{}}}
8157 \notation{apply}[lambda]{#1\; \argsep{#2}{\;}}
8158 \notation{apply}[infixop]{\argsep{#2}{\mathbin{#1}}\STEXinvisible{#1}}
8159 \notation{apply}[infixrel]{\argsep{#2}{\mathrel{#1}}\STEXinvisible{#1}}
8160
8161 \symdef{autoprove}[name=prove automatically]{\mathrm{automatically\ proved}}
8162
8163 % structures
8164 \symdef{module~type}[args=i,op=\mathtt{MOD}]
8165   {\mathopen{\comp{\mathtt{MOD}}}{}#1\mathclose{\comp{}}}
8166 \symdef{module~type~merge}[args=a,op=\oplus]
8167   {\argsep{#1}{\mathbin{\comp{\oplus}}}}
8168 \symdef{anonymous~record}[args=a]
```

```

8169   {\mathopen{\comp{[[]}\#1\mathclose{\comp{[]}}}}
8170 \symdef{record~field}[args=2]{\#1\comp{.}\#2}
8171 \symdecl*{record~type}
8172
8173 \symdecl{mathstruct}[name=mathematical~structure,args=a] % TODO
8174 \notation{mathstruct}[angle,prec=nobrackets]
8175   {\mathopen{\comp{\langle} \#1 \mathclose{\comp{\rangle}}}
8176 \notation{mathstruct}[parens,prec=nobrackets]
8177   {\mathopen{\comp{() \#1 \mathclose{\comp{}}}
8178
8179 % sequences
8180 \symdef{ellipses}[ldots]{\ldots}
8181 \symdef{sequence~expression}[comma,args=a]{\#1}
8182 \symdef{sequence~type}[args=1]{\#1^{\comp{ast}}}
8183 \symdef{ranged~sequence~type}[args=ia]{\#1\c_math_subscript_token{\dobrackets{\#2}}}
8184 \symdef{sequence~range}[inv,args=a]{\#1}
8185 \symdef{sequence~map}[args=ai]{
8186   \comp{\mathrm{map}}\mathopen{\comp{()}\#1\mathpunct{\comp{,}}}
8187   \#2\mathclose{\comp{()}}
8188 }
8189 \symdef{fold~right}[args=aibbi]{
8190   \maincomp{\mathrm{fold}}\dobrackets{\#2\mathpunct{\comp{;}}\#1\mathpunct{\comp{;}}
8191   \dobrackets{\#3,\#4} \mathrel{\comp{\mapsto}} \#5
8192 }
8193 }
8194
8195 \symdef{fold}[args=abbi]{
8196   \maincomp{\mathrm{fold}}\dobrackets{\#1\mathpunct{\comp{;}}
8197   \dobrackets{\#2,\#3} \mathrel{\comp{\mapsto}} \#4
8198 }
8199 }
8200
8201 \symdecl{bind}[args=Bi,assoc=pre]
8202 \notation{bind}[defun,prec=nobrackets,op=(\cdot)\;\to\;\cdot]
8203   {\mathopen{\comp{() \#1 \mathclose{\comp{()}\mathbin{\comp{\to}}}} \#2}
8204 \notation{bind}[forall]{\comp{\forall} \#1.\;\#2}
8205 \notation{bind}[Pi]{\mathop{\comp{\prod}}\c_math_subscript_token{\#1}\#2}
8206
8207 \symdef{implicit~bind}[args=Bi,assoc=pre]{\mathopen{\comp{\}} \#1 \mathclose{\comp{\}}\c_math_
8208 \symdef{apply~implicits}[args=ia,prec=nobrackets]{
8209   \underbrace{\#1}_{\#2}
8210 }
8211
8212 \symdecl{arrow}[args=ai,assoc=pre]
8213 \notation{arrow}[single]{
8214   \argsep{\#1}{\mathbin{\comp{\times}}}
8215   \mathrel{\maincomp{\to}} \#2
8216 }
8217 \notation{arrow}[double]{
8218   \argsep{\#1}{\mathbin{\comp{\times}}}
8219   \mathrel{\maincomp{\Rrightarrow}} \#2
8220 }
8221
8222 \symdef{intsecttp}[intersection type,args=a,sqcap,prec=0;-10]{ \argsep{\#1}{\maincomp{\mathr

```



```

8223
8224 \symdef{bindin}[args=Bi]{ \mathop{\maincomp{\amalg}}_{\#1}\#2 }
8225
8226 \symdecl*{integer~literal}
8227 \notation{integer~literal}{\mathbb Z}
8228
8229 \symdecl*{ordinal}
8230 \notation{ordinal}{\mathtt{Ord}}
8231
8232 % propositions
8233 \symdef{prop}[name=proposition]{\mathtt{Prop}}
8234 \symdef{judgment~holds}[args=i,role=judgment]{\comp\vdash\;#1}
8235
8236 % any object
8237 \symdef{object}{\mathtt{Obj}}
8238
8239 \symdef{letin}[name=let in,args=Bi,role=let]{\maincomp{\mathrm{let}}\ #1\ \comp{\mathrm{in}}}
8240
8241 % TODO DELETE
8242 \symdef{aseqdots}[args=a,prec=nobrackets]
8243   {\#1\comp{\, \ldots}}\%{\##1\comp,\##2}
8244 \symdef{aseqfromto}[args=ai,prec=nobrackets]
8245   {\#1\comp{\, \ldots,}\#2}\%{\##1\comp,\##2}
8246 \symdef{aseqfromtovia}[args=aai,prec=nobrackets]
8247   {\#1\comp{\, \ldots,}\#2\comp{\, \ldots,}\#3}\%{\##1\comp,\##2}

Closing:
8248 \global \let \l_stex_metatheory_uri \l_stex_current_module_uri
8249 \global \let \c_stex_default_metatheory \l_stex_metatheory_uri
8250 \stex_close_module:
8251 \group_end:

```

Chapter 38

Others

```
8252 <@@=stex_others>
8253
8254 \cs_new_protected:Npn \MSC #1 {}
8255
8256 \_stex_persist_read_now:
8257 \cs_new_protected:Nn \__stex_others_newlabel:n {
8258   \tl_gset:cn{__stex_sref_aux_sym_ \tl_to_str:n{#1}}{X}
8259   \exp_args:Nc\__stex_others_old_newlabel:{\tl_to_str:n{#1}}
8260 }
8261 \AtBeginDocument{
8262   \iow_now:Nn \@auxout {
8263     \ExplSyntaxOn
8264     \let\__stex_others_old_newlabel:\newlabel
8265     \let\newlabel\__stex_others_newlabel:n
8266     \ExplSyntaxOff
8267   }
8268 }
```

Inference Rules

```
8269 \NewDocumentCommand \FTMLrule {m m}{
8270   \tl_if_empty:nTF{#2}{
8271     \seq_clear:N \l_tmpa_seq
8272   }{
8273     \seq_set_split:Nnn \l_tmpa_seq , {#2}
8274   }
8275   \int_zero:N \l_tmpa_int
8276   \stex_annotate_invisible:n{\hbox{
8277     $
8278     \stex_annotate:nn{data-ftml-inferencerule={#1}}{
8279       \_stex_annotate_force_break:n{~
8280         \seq_if_empty:NF \l_tmpa_seq {
8281           \seq_map_inline:Nn \l_tmpa_seq {
8282             \int_incr:N \l_tmpa_int
8283             \stex_annotate:nn{
8284               data-ftml-argmode=i,
8285               data-ftml-arg={\int_use:N \l_tmpa_int}
8286             }{ ##1 }
8287           }
8288         }
8289       }
8290     }
8291   }
```

```

8289         ~}
8290     }$
8291 }}
8292 }
8293 \stex_deactivate_macro:Nn \FTMLrule {module~environments}
8294 \stex_every_module:n{\stex_reactivate_macro:N \FTMLrule}

```

Chapter 39

Additional Packages

39.1 Implementation: The notesslides Package

39.1.1 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.

```
8295 \*cls)
8296 @@=notesslides)
8297 \ProvidesExplClass{notesslides}{2026/06/24}{4.1.0}{notesslides Class}
8298 \RequirePackage{13keys2e}
8299
8300 \str_const:Nn \c__notesslides_class_str {article}
8301
8302 \keys_define:nn{notesslides / cls}{
8303   class .str_set_x:N = \c__notesslides_class_str,
8304   notes .bool_set:N = \c__notesslides_notes_bool ,
8305   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
8306   %docopt .str_set_x:N = \c__notesslides_docopt_str,
8307   unknown .code:n = {
8308     \PassOptionsToClass{\CurrentOption}{beamer}
8309     \PassOptionsToClass{\CurrentOption}{\c__notesslides_class_str}
8310     \PassOptionsToPackage{\CurrentOption}{notesslides}
8311     \PassOptionsToPackage{\CurrentOption}{stex}
8312   }
8313 }
8314 \ProcessKeysOptions{ notesslides / cls }
8315
8316 \RequirePackage{stex}
8317 \stex_if_html_backend:T {
8318   \bool_set_true:N \c__notesslides_notes_bool
8319 }
8320
8321 \bool_if:NTF \c__notesslides_notes_bool {
8322   \PassOptionsToPackage{notes=true}{notesslides}
8323   \message{notesslides.cls:~Formatting~document~in~notes~mode}
```

```

8324 }{
8325   \PassOptionsToPackage{notes=false}{notesslides}
8326   \message{notesslides.cls:~Formatting~document~in~slides~mode}
8327 }
8328
8329 \bool_if:NTF \c_notesslides_notes_bool {
8330   \LoadClass{\c_notesslides_class_str}
8331 }{
8332   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
8333   %\newcounter{Item}
8334   %\newcounter{paragraph}
8335   %\newcounter{subparagraph}
8336   %\newcounter{Hfootnote}
8337 }
8338 \RequirePackage{notesslides}
8339 </cls>

```

now we do the same for the notesslides package.

```

8340 (*package)
8341 \ProvidesExplPackage{notesslides}{2026/06/24}{4.1.0}{notesslides Package}
8342 \RequirePackage{l3keys2e}
8343
8344 \keys_define:nn{notesslides / pkg}{
8345   notes .bool_set:N = \c_notesslides_notes_bool ,
8346   slides .code:n = { \bool_set_false:N \c_notesslides_notes_bool },
8347   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
8348   topsect .str_set_x:N = \c_notesslides_topsect_str,
8349   unknown .code:n = {
8350     \PassOptionsToPackage{\CurrentOption}{stex}
8351     \PassOptionsToPackage{\CurrentOption}{tikzinput}
8352   }
8353 }
8354 \ProcessKeysOptions{ notesslides / pkg }
8355
8356 \RequirePackage{stex}
8357 \stex_if_html_backend:T {
8358   \bool_set_true:N \c_notesslides_notes_bool
8359 }
8360
8361 \cs_set:Npn \sectiontitleemph #1 {
8362   \textbf{\Large #1}
8363 }
8364
8365 \newif\ifnotes
8366 \bool_if:NTF \c_notesslides_notes_bool {
8367   \notesttrue
8368   \PassOptionsToPackage{dvipsnames,svgnames}{xcolor}
8369   \RequirePackage[noamsthm,hyperref]{beamerarticle}
8370   \RequirePackage{mdframed}
8371   \str_if_empty:NTF \c_notesslides_topsect_str{
8372     %\setsectionlevel{section}
8373   } {
8374     \exp_args:No \setsectionlevel \c_notesslides_topsect_str
8375   }

```

```

8376 }{
8377 \notesfalse
8378
8379 \cs_new_protected:Nn \__notesslides_do_sectocframes: {
8380 \cs_set_protected:Nn \__notesslides_do_label:n {
8381 \str_case:nnF{##1}{
8382 {part} {
8383 \tl_set:Nx\l__notesslides_num{\thepart}
8384 \tl_set:cx{@ @ label}{
8385 \cs_if_exist:NTF\parttitlename{\exp_not:N\parttitlename}{\exp_not:N\partname}{
8386 }
8387 {chapter} {
8388 \tl_set:Nx\l__notesslides_num{\thechapter}
8389 \tl_set:cx{@ @ label}{
8390 \cs_if_exist:NTF\chaptertitlename{\exp_not:N\chaptertitlename}{\exp_not:N\chaptername}{
8391 }
8392 {section} {
8393 \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesectionname}
8394 \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
8395 }
8396 {subsection} {
8397 \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesectionname}
8398 \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
8399 }
8400 {subsubsection} {
8401 \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesectionname}
8402 \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
8403 }
8404 {paragraph} {
8405 \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesectionname}
8406 \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
8407 }
8408 }{
8409 \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesectionname}
8410 \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
8411 }
8412 }
8413 \cs_set_protected:Nn \_sfragment_do_level:nn {
8414 \tl_if_exist:cT{c@##1}{\stepcounter{##1}}
8415 \addcontentsline{toc}{##1}{\protect\numberline{\use:c{the##1}}##2}
8416 \__notesslides_do_label:n{##1}
8417 \pdfbookmark[\int_use:N \l_stex_current_section_level_int]{\l__notesslides_num\ ##2}{
8418 \begin{frame}[noframenumbering]
8419 \vfill\centering
8420 \sectiontitleemph{
8421 \use:c{@ @ label} ##2
8422 }
8423 \end{frame}
8424 \int_incr:N \l_stex_current_section_level_int
8425 \str_set:Nn \l_stex_current_section_level_str{##1}
8426 }
8427 }
8428
8429 \cs_set_protected:Nn \_stex_titlefragment: {

```

```

8430     \begin{frame}[noframenumbering]\maketitle\end{frame}
8431     \let\maketitle\relax
8432 }
8433
8434 \AtBeginDocument{
8435     \str_if_empty:NTF \c_notesslides_topsect_str {
8436         \setsectionlevel{section}
8437     } {
8438         \exp_args:No \setsectionlevel \c_notesslides_topsect_str
8439         \exp_args:No \str_if_eq:nnTF \c_notesslides_topsect_str {chapter} {
8440             \__notesslides_define_chapter:
8441         }{
8442             \exp_args:No \str_if_eq:nnT \c_notesslides_topsect_str {part} {
8443                 \__notesslides_define_chapter:
8444                 \__notesslides_define_part:
8445             }
8446         }
8447     }
8448 }
8449
8450 \bool_if:NT \c_notesslides_sectocframes_bool {
8451     \__notesslides_do_sectocframes:
8452 }
8453 }
8454
8455 \cs_new_protected:Nn \__notesslides_define_chapter: {
8456     \cs_if_exist:NF \chaptername {
8457         \cs_set_protected:Npn \chaptername {Chapter}
8458     }
8459     \cs_if_exist:NF \chapter {
8460         \cs_set_protected:Npn \chapter {INVALID}
8461     }
8462     \cs_if_exist:NF \c@chapter {
8463         \newcounter{chapter}\counterwithin*{section}{chapter}
8464     }
8465 }
8466
8467 \cs_new_protected:Nn \__notesslides_define_part: {
8468     \cs_if_exist:NF \partname {
8469         \cs_set_protected:Npn \partname {Part}
8470     }
8471     \cs_if_exist:NF \part {
8472         \cs_set_protected:Npn \part {INVALID}
8473     }
8474     \cs_if_exist:NF \c@part {
8475         \newcounter{part}\counterwithin*{chapter}{part}
8476     }
8477 }

```

\prematuarestop We initialize \afterprematuarestop, and provide \prematuarestop@endsfragment which looks up \sfragment@level and recursively ends enough {sfragment}s.

```

8478 \def \c__notesslides_document_str{document}
8479 \newcommand\afterprematuarestop{}
8480 \def\prematuarestop@endsfragment{

```

```

8481 \unless\ifx\@currenvir\c__notesslides_document_str
8482 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
8483 \expandafter\prematurestop@endsfragment
8484 \fi
8485 }
8486 \providecommand\prematurestop{
8487 \stex_if_html_backend:F{
8488 \message{Stopping~sTeX~processing~prematurely}
8489 \prematurestop@endsfragment
8490 \afterprematurestop
8491 \end{document}}
8492 }
8493 }

```

(End of definition for \prematurestop. This function is documented on page 111.)

39.1.2 Notes and Slides

For the notes case, we also provide the \usetheme macro that would otherwise come from the the beamer class.

```

8494 \bool_if:NT \c_notesslides_notes_bool {
8495 \renewcommand\usetheme[2][\usepackage[#1]{beamertheme#2}]
8496 }
8497 \NewDocumentCommand \libusetheme {0} m {
8498 \libusepackage[#1]{beamertheme#2}
8499 }

```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```

8500 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
8501 \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.

```

8502 \ifnotes
8503
8504 \newlength{\slideframewidth}
8505 \setlength{\slideframewidth}{1.5pt}

```

frame (env.) We first define the keys.

```

8506 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
8507 \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
8508 \bool_set_true:N #1
8509 }{
8510 \bool_set_false:N #1
8511 }
8512 }
8513
8514 \stex_keys_define:nnnn{notesslides / frame}{
8515 \str_clear:N \l__notesslides_frame_label_str
8516 \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
8517 \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
8518 \bool_set_true:N \l__notesslides_frame_fragile_bool
8519 \bool_set_true:N \l__notesslides_frame_shrink_bool

```



```

8520 \bool_set_true:N \l__notesslides_frame_squeeze_bool
8521 \bool_set_true:N \l__notesslides_frame_t_bool
8522 }{
8523   label .str_set_x:N = \l__notesslides_frame_label_str,
8524   allowframebreaks .code:n = {
8525     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
8526   },
8527   allowdisplaybreaks .code:n = {
8528     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
8529   },
8530   fragile .code:n = {
8531     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
8532   },
8533   shrink .code:n = {
8534     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
8535   },
8536   squeeze .code:n = {
8537     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
8538   },
8539   t .code:n = {
8540     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
8541   },
8542   unknown .code:n = {}
8543 }{}

```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```

8544 \cs_new_protected:Nn \__notesslides_setup_itemize: {
8545   \def\itemize@level{outer}
8546   \def\itemize@outer{outer}
8547   \def\itemize@inner{inner}
8548   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
8549   \renewenvironment{itemize}{
8550     \ifx\itemize@level\itemize@outer
8551       \def\itemize@label{$\rhd$}
8552     \fi
8553     \ifx\itemize@level\itemize@inner
8554       \def\itemize@label{$\scriptstyle\rhd$}
8555     \fi
8556     \begin{list}
8557     {\itemize@label}
8558     {\setlength{\labelsep}{.3em}
8559      \setlength{\labelwidth}{.5em}
8560      \setlength{\leftmargin}{1.5em}
8561     }
8562     \edef\itemize@level{\itemize@inner}
8563   }{
8564     \end{list}
8565   }
8566 }

```

We create the box with the `mdframed` environment from the `equinymous` package.

```

8567 \stex_if_html_backend:TF {
8568   %\stex_css_literal:n{
8569   % .stex-frame {

```

```

8570 % background-color:#fff;
8571 % margin:5px ~ 0;
8572 % border:2px ~ solid ~ #000;
8573 % border-radius:5px;
8574 % padding:5px;
8575 % padding-right:10px;
8576 % width: var(--rustex-curr-width);
8577 % display:block;
8578 % }
8579 %}

8580
8581 \cs_new_protected:Nn \__notesslides_frame_box_begin: {
8582   \vbox\bgroup
8583   \begin{stex_annotate_env}{data-ftml-slide={}}%{class:stex-frame={}}
8584     \mdf@patchamsthm\notesslidesfont
8585   }
8586 \cs_new_protected:Nn \__notesslides_frame_box_end: {
8587   %^A \notesslides@slidelabel
8588   \medskip\par\hbox to \textwidth{\tiny\notesslidesfooter}
8589   \end{stex_annotate_env}\egroup
8590 }
8591 }{
8592 \cs_new_protected:Nn \__notesslides_frame_box_begin: {
8593   \begin{mdframed}[
8594     linewidth=\slideframewidth,
8595     skipabove=1ex,
8596     skipbelow=1ex,
8597     userdefinedwidth=\slidewidth,
8598     align=center,
8599     %usetwoside=false
8600   ]\notesslidesfont
8601   }
8602 \cs_new_protected:Nn \__notesslides_frame_box_end: {
8603   \medskip\par\noindent\tiny\notesslidesfooter%^A\notesslides@slidelabel
8604   \end{mdframed}
8605   }
8606 }

```

We define the environment, read them, and construct the slide number and label.

```

8607 \renewenvironment{frame}[1][]{
8608   \stex_keys_set:nn{notesslides / frame}{#1}
8609   \stepcounter{framenum}
8610   \renewcommand\newpage{\addtocounter{framenum}{1}}
8611   \def\@currentlabel{\theframenum}
8612   \str_if_empty:NF \l__notesslides_frame_label_str {
8613     \label{\l__notesslides_frame_label_str}
8614   }
8615   \__notesslides_setup_itemize:
8616   \__notesslides_frame_box_begin:
8617 }{
8618   \__notesslides_frame_box_end:
8619 }

```

Now, we need to redefine the frametitle (we are still in course notes mode).


```

8666 %      \rustex_if:T {\par
8667 %      \rustex_direct_HTML:n{</td><td>}
8668 %      }
8669 %  }
8670 % }
8671 \end{lrbox}\usebox\columnbox
8672 }
8673 \fi

```

39.1.3 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.

```

8674 \cs_if_exist:cTF{note}{
8675   \bool_if:NTF \c_notesslides_notes_bool {
8676     \renewenvironment{note}{\ignorespaces}{ }
8677   }{
8678     \renewenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup}{\egroup}
8679   }
8680 }{
8681   \bool_if:NTF \c_notesslides_notes_bool {
8682     \newenvironment{note}{\ignorespaces}{ }
8683   }{
8684     \newenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup}{\egroup}
8685   }
8686 }

```

For other environments we introduce variants prefixed with `n`, which are excluded in `slides` mode.

```

8687 \cs_new_protected:Nn \__notesslides_notes_env:nnnn {
8688   \bool_if:NTF \c_notesslides_notes_bool {
8689     \newenvironment{#1}#2{#3}{#4}
8690   }{
8691     \newenvironment{#1}#2{
8692       \cs_set:Npn \__notesslides_eat: ####1 \end ####2 {
8693         \str_if_eq:nnTF{#1}{####2}{
8694           \end{#1}
8695         }{
8696           \__notesslides_eat:
8697         }
8698       }
8699       \__notesslides_eat:
8700       %\setbox\l_tmpa_box\vbox\bgroup#3
8701     }{
8702       %#4\egroup
8703     }
8704   }
8705 }
8706
8707 \__notesslides_notes_env:nnnn{nparagraph}{[1] []}{\begin{sparagraph}[#1]}{\end{sparagraph}}
8708 \__notesslides_notes_env:nnnn{nfragment}{[2] []}{\begin{sfragment}[#1]{#2}}{\end{sfragment}}
8709 \__notesslides_notes_env:nnnn{ndefinition}{[1] []}{\begin{sdefinition}[#1]}{\end{sdefinition}}

```

```

8710 \_notesslides\_notes\_env:nnnn{nassertion}{[1][ ]}{\begin{sassertion}[#1]}{\end{sassertion}}
8711 \_notesslides\_notes\_env:nnnn{nproof}{[2][ ]}{\begin{sproof}[#1]{#2}}{\end{sproof}}
8712 \_notesslides\_notes\_env:nnnn{nexample}{[1][ ]}{\begin{sexample}[#1]}{\end{sexample}}
8713
8714 \RequirePackage{graphicx}
8715
8716 \NewDocumentCommand\frameimage{s O{} m}{
8717   \IfBooleanTF #1 {
8718     \begin{frame}[plain]
8719   }{
8720     \begin{frame}
8721   }
8722   \bool_if:NTF \c_notesslides\_notes\_bool {
8723     \slidewidth=\dimexpr\slidewidth-(2\slideframewidth)\relax
8724   }{
8725     \slidewidth=\textwidth\relax
8726   }
8727   \def\Gin@ewidth{}\setkeys{Gin}{#2}
8728   \tl_if_empty:NTF \Gin@ewidth {
8729     \mhgraphics[width=\slidewidth,#2]{#3}
8730   }{
8731     \mhgraphics[#2]{#3}
8732   }
8733   \end{frame}
8734 }

```

hacking inputref:

\inputref*

```

8735 \cs_set_eq:NN\_notesslides\_inputref:\inputref
8736 \cs_set_protected:Npn\inputref{\@ifstar\ninputref\_notesslides\_inputref:}
8737 \bool_if:NTF \c_notesslides\_notes\_bool {
8738   \newcommand\ninputref[2][ ]{
8739     \_notesslides\_inputref:[#1]{#2}
8740   }
8741 }{
8742   \newcommand\ninputref[2][ ]{}
8743 }

```

(End of definition for \inputref*. This function is documented on page 110.)

39.1.4 Styling Across Notes/Slides

```

8744 \def\notesslides\titleemph#1{
8745   {\Large\bf\sffamily#1}
8746   \vskip0.1\baselineskip
8747   \leaders\vrule width \textwidth
8748   \vskip0.4pt%
8749   \nointerlineskip
8750 }
8751
8752 \def\notesslides\footer{}
8753
8754 \let\notesslides\font\sffamily

```

39.1.5 Beamer Compatibility

All of this should be removed and made part of a template

```

8755
8756 \stex_if_do_html:TF{
8757   \NewDocumentEnvironment{slideshow}{}{
8758     \begin{stex_annotate_env}{data-ftml-slideshow={}}
8759     \cs_set_protected:Npn \nextslide ##1 {
8760       \begin{stex_annotate_env}{data-ftml-slideshow-slide={}} ##1
8761       \end{stex_annotate_env}
8762     }
8763     \cs_set_protected:Npn \lastslide ##1 {
8764       \begin{stex_annotate_env}{data-ftml-slideshow-slide={}} ##1
8765       \end{stex_annotate_env}
8766     }
8767   }{
8768     \end{stex_annotate_env}
8769   }
8770 }{
8771   \int_new:N \l__notesslides_slideshow_counter_int
8772   \NewDocumentEnvironment{slideshow}{}{
8773     \int_zero:N \l__notesslides_slideshow_counter_int
8774     \cs_set_protected:Npn \nextslide ##1 {
8775       \int_incr:N \l__notesslides_slideshow_counter_int
8776       \only<\int_use:N \l__notesslides_slideshow_counter_int>{##1}
8777     }
8778     \cs_set_protected:Npn \lastslide ##1 {
8779       \int_incr:N \l__notesslides_slideshow_counter_int
8780       \only<\int_use:N \l__notesslides_slideshow_counter_int ->{##1}
8781     }
8782   }{
8783
8784   }
8785 }
8786
8787 \bool_if:NT \c_notesslides_notes_bool {
8788   \def\author{\@dblarg\ns@author}
8789   \long\def\ns@author[#1]#2{%
8790     \tl_if_empty:nTF{#1}{
8791       \def\beamer@shortauthor{#2}
8792     }{
8793       \def\beamer@shortauthor{#1}
8794     }
8795     \def\@author{#2}
8796   }
8797   \def\title{\@dblarg\ns@title}
8798   \long\def\ns@title[#1]#2{%
8799     \tl_if_empty:nTF{#1}{
8800       \def\beamer@shorttitle{#2}
8801     }{
8802       \def\beamer@shorttitle{#1}
8803     }
8804     \def\@title{#2}
8805     \stexdoctitle{#2}

```

```

8806 }
8807 \def\insertshortauthor{
8808   \hbox\bgroup\def\{\}\cs_if_exist:NT\beamer@shortauthor\beamer@shortauthor\egroup
8809 }
8810 \def\insertshorttitle{
8811   \hbox\bgroup\def\{\}\cs_if_exist:NT\beamer@shorttitle\beamer@shorttitle\egroup
8812 }
8813 \stex_if_html_backend:TF{
8814   \def\insertframenumber{\stex_annotate:nn{data-ftml-slide-number={}}{}}
8815 }{
8816   \def\insertframenumber{\@arabic\c@framenumber}
8817 }
8818 \def\insertshortdate{\today}
8819 }

```

39.1.6 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.

```

8820 \gdef\printexcursions{}
8821
8822
8823
8824 \stex_if_html_backend:TF{
8825   \newcommand\excursionref[4][\% repos, label, path, text
8826     \begin{sparagraph}[title=Excursion]
8827       #4~
8828       \stex_in_archive:nn{#1}{
8829         \stex_document_uri_from_archive_file:Nn \l__notesslides_uri {#3}
8830         \exp_args:Ne\href{\stex_use_document_uri:N\l__notesslides_uri}{here}
8831       }
8832     \end{sparagraph}
8833   }
8834   \def\excursion{\excursionref}
8835 }{
8836   \newcommand\excursionref[4][\% repos, label, path, text
8837     \bool_if:NT \c_notesslides_notes_bool {
8838       \begin{sparagraph}[title=Excursion]
8839         #4~\sref[fallback=the appendix]{#2}.
8840       \end{sparagraph}
8841     }
8842   }
8843   \newcommand\activate@excursion[2][\{
8844     \tl_gput_right:Nn\printexcursions{\inputref[#1]{#2}}
8845   }
8846   \newcommand\excursion[4][\% repos, label, path, text
8847     \bool_if:NT \c_notesslides_notes_bool {
8848       \activate@excursion[#1]{#3}
8849       \excursionref[#1]{#2}{#3}{#4}
8850     }
8851   }
8852

```

8853 }

(End of definition for \excursion. This function is documented on page 112.)

\excursiongroup

```

8854 \keys_define:nn{notesslides / excursiongroup }{
8855   id          .str_set_x:N = \l__notesslides_excursion_id_str,
8856   intro       .tl_set:N    = \l__notesslides_excursion_intro_tl,
8857   archive     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
8858 }
8859 \cs_new_protected:Nn \l__notesslides_excursion_args:n {
8860   \tl_clear:N \l__notesslides_excursion_intro_tl
8861   \str_clear:N \l__notesslides_excursion_id_str
8862   \str_clear:N \l__notesslides_excursion_mhrepos_str
8863   \keys_set:nn {notesslides / excursiongroup }{ #1 }
8864 }
8865
8866
8867 \stex_if_html_backend:TF{
8868   \newcommand\excursiongroup[1][]{ }
8869 }{
8870   \newcommand\excursiongroup[1][]{
8871     \l__notesslides_excursion_args:n{ #1 }
8872     \tl_if_empty:NF\printexcursions
8873     {\IfInputref{ }\begin{note}
8874       \begin{sfragment}{Excursions}% TODO pass on id
8875       \ifdefempty\l__notesslides_excursion_intro_tl{ }{
8876         \exp_args:NNe \use:nn \inputref{[\l__notesslides_excursion_mhrepos_str]{
8877           \l__notesslides_excursion_intro_tl
8878         }}
8879       }
8880       \printexcursions%
8881       \end{sfragment}
8882       \end{note}}}
8883   }
8884 }
8885
8886 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi

```

(End of definition for \excursiongroup. This function is documented on page 112.)

```

8887 \prop_new:N \g__notesslides_variables_prop
8888 \cs_set_protected:Npn \setSGvar #1 #2 {
8889   \prop_gput:Nnn \g__notesslides_variables_prop {#1}{#2}
8890 }
8891 \cs_set_protected:Npn \useSGvar #1 {
8892   \prop_item:Nn \g__notesslides_variables_prop {#1}
8893 }
8894 \cs_set_protected:Npn \ifSGvar #1 #2 #3 {
8895   \prop_get:NnNF \g__notesslides_variables_prop {#1} \l__notesslides_tmp {
8896     \PackageError{document-structure}
8897     {The sTeX Global variable #1 is undefined}
8898     {set it with \protect\setSGvar}\TODO better error
8899   }
8900   \tl_if_eq:NnT \l__notesslides_tmp {#2}{ #3 }

```



```

8901 }
8902
8903
8904 </package>

```

39.2 Implementation: The problem Package

39.2.1 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.

```

8905 <*package>
8906 <@@=problems>
8907 \ProvidesExplPackage{problem}{2026/06/24}{4.1.0}{Semantic Markup for Problems}
8908 \RequirePackage{l3keys2e}
8909
8910 \keys_define:nn { problem / pkg }{
8911   notes      .default:n      = { true },
8912   notes      .bool_set:N     = \c__problems_notes_bool,
8913   gnotes     .default:n      = { true },
8914   gnotes     .bool_set:N     = \c__problems_gnotes_bool,
8915   hints      .default:n      = { true },
8916   hints      .bool_set:N     = \c__problems_hints_bool,
8917   solutions  .default:n      = { true },
8918   solutions  .bool_set:N     = \c__problems_solutions_bool,
8919   pts        .default:n      = { true },
8920   pts        .bool_set:N     = \c__problems_pts_bool,
8921   min        .default:n      = { true },
8922   min        .bool_set:N     = \c__problems_min_bool,
8923   %boxed     .default:n      = { true },
8924   %boxed     .bool_set:N     = \c__problems_boxed_bool,
8925   test       .default:n      = { true },
8926   test       .bool_set:N     = \c__problems_test_bool,
8927   unknown    .code:n         = {
8928     \PassOptionsToPackage{\CurrentOption}{stex}
8929   }
8930 }
8931 \newif\ifsolutions
8932
8933 \ProcessKeysOptions{ problem / pkg }
8934 \bool_if:NTF \c__problems_solutions_bool {
8935   \solutionstrue
8936 }{
8937   \solutionsfalse
8938 }
8939 \newif\ifintest
8940 \bool_if:NTF \c__problems_test_bool {
8941   \intesttrue
8942 }{
8943   \intestfalse
8944 }
8945

```

```

8946 \RequirePackage{stex}
8947
8948 \ifintest
8949   \AtBeginDocument{
8950     \def\symrefemph#1{#1}
8951     \def\compemph#1{#1}
8952     \def\varemp#1{#1}
8953   }
8954 \fi
8955
8956 \newcommand\precondition[2]{
8957   \stex_get_symbol:n{#2}
8958   \tl_if_empty:NF \l_stex_get_symbol_uri {
8959     \str_case:nnTF {#1}{
8960       {remember}{}
8961       {understand}{}
8962       {analyze}{}
8963       {evaluate}{}
8964       {apply}{}
8965       {create}{}
8966     }{
8967       \ifstexhtml\else\bool_if:NT\c_stex_metadata_bool {
8968         \marginpar{\tiny \textbf{Precondition:}}~#1~
8969         \exp_args:Ne\symrefemph@uri{\stex_symbol_uri_name:N \l_stex_get_symbol_uri}{\l_stex
8970       }
8971     }\fi
8972     \stex_annotate_invisible:nn{
8973       data-ftml-preconditionsymbol={\stex_use_symbol_uri:N \l_stex_get_symbol_uri},
8974       data-ftml-preconditiondimension={#1}
8975     }{}
8976   }{\errmessage{Unknown~cognitive~dimension~#1}}
8977 }
8978 }
8979 \newcommand\objective[2]{
8980   \stex_get_symbol:n{#2}
8981   \tl_if_empty:NF \l_stex_get_symbol_uri {
8982     \str_case:nnTF {#1}{
8983       {remember}{}
8984       {understand}{}
8985       {analyze}{}
8986       {evaluate}{}
8987       {apply}{}
8988       {create}{}
8989     }{
8990       \ifstexhtml\else\bool_if:NT\c_stex_metadata_bool {
8991         \marginpar{\tiny \textbf{Objective:}}~#1~
8992         \exp_args:Ne\symrefemph@uri{\stex_symbol_uri_name:N \l_stex_get_symbol_uri}{\l_stex
8993       }
8994     }\fi
8995     \stex_annotate_invisible:nn{
8996       data-ftml-objectivesymbol={\stex_use_symbol_uri:N \l_stex_get_symbol_uri},
8997       data-ftml-objectivedimension={#1}
8998     }{}
8999   }{\errmessage{Unknown~cognitive~dimension~#1}}

```

```

9000 }
9001 }

```

\problem@kw@* For multilinguality, we define internal macros for keywords that can be specialized in *.ldf files.

```

9002 \AddToHook{begindocument}{
9003   \ExplSyntaxOn\makeatletter
9004   \input{problem-english.ldf}
9005   \ltx@ifpackageloaded{babel}{
9006     \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
9007     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
9008       \input{problem-ngerman.ldf}
9009     }
9010     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
9011       \input{problem-finnish.ldf}
9012     }
9013     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
9014       \input{problem-french.ldf}
9015     }
9016     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
9017       \input{problem-russian.ldf}
9018     }
9019   }{}
9020   \makeatother\ExplSyntaxOff
9021 }

```

(End of definition for \problem@kw@*. This function is documented on page ??.)

39.2.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```

9022 \bool_new:N \l_stex_key_autogradable_bool
9023 \stex_keys_define:nnnn{ problem }{
9024   \tl_clear:N \l_stex_key_pts_tl
9025   \tl_set:Nn \l_stex_key_min_tl 0
9026   \str_clear:N \l_stex_key_name_str
9027   \str_clear:N \l_stex_key_mhrepos_str
9028   \bool_set_false:N \l_stex_key_autogradable_bool
9029 }{
9030   pts .tl_set:N = \l_stex_key_pts_tl,
9031   min .tl_set:N = \l_stex_key_min_tl,
9032   name .str_set:N = \l_stex_key_name_str,
9033   autogradable .bool_set:N = \l_stex_key_autogradable_bool,
9034   archive .code:n = {},
9035   %archive .str_set:N = \l_stex_key_mhrepos_str,
9036   %creators .code:n = {}
9037   %imports .tl_set:N = \l__problems_prob_imports_tl,
9038   %refnum .int_set:N = \l__problems_prob_refnum_int,
9039 }{id,title,style,uses}

```

Then we set up a counter for problems.

\numberproblemsin

```

9040 \newcounter{sproblem}[section]
9041 \newcommand\numberproblemsin[1]{
9042   \@addtoreset{sproblem}{#1}
9043   \def\thesproblem{\arabic{#1}.\arabic{sproblem}}
9044 }
9045 \numberproblemsin{section}
9046 %\def\theplainsproblem{\arabic{sproblem}}
9047 %\def\thesproblem{\thesection.\theplainsproblem}

```

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

sproblem (env.)

```

9048 \cs_new:Nn \__problems_activate_macros: {
9049   \stex_reactivate_macro:N \solution
9050   \stex_reactivate_macro:N \mcb
9051   \stex_reactivate_macro:N \scb
9052   \stex_reactivate_macro:N \fillinsol
9053   \stex_reactivate_macro:N \hint
9054   \stex_reactivate_macro:N \exnote
9055   \stex_reactivate_macro:N \gnote
9056 }
9057
9058 \fp_new:N \g_problem_total_pts_fp
9059 \fp_new:N \g_problem_total_min_fp
9060
9061 \bool_new:N \l__problems_in_problem_bool
9062 \bool_new:N \l__problems_has_pts_bool
9063 \bool_new:N \l__problems_has_min_bool
9064 \bool_set_false:N \l__problems_in_problem_bool
9065 \stex_new_stylable_env:nnnnnnn {problem} {0{}}{
9066   \bool_if:NT \l__problems_in_problem_bool {
9067     \msg_error:nn{stex}{error/nestedproblem}
9068   }
9069   \cs_if_exist:NTF \l_problem_inputproblem_keys_tl {
9070     \tl_put_left:Nn \l_problem_inputproblem_keys_tl {#1,}
9071     \exp_args:Nno \stex_keys_set:nn{problem}{
9072       \l_problem_inputproblem_keys_tl
9073     }
9074   }{
9075     \stex_keys_set:nn{problem}{#1}
9076   }
9077   \refstepcounter{sproblem}
9078
9079   \stex_if_do_html:T {
9080     \str_if_empty:NT \l_stex_key_name_str {
9081       \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
9082       \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
9083     }
9084     \exp_args:Nne \begin{stex_annotate_env} {
9085       data-ftml-problem={\l_stex_key_name_str},
9086       %data-ftml-language={ \l_stex_current_language_str},
9087       data-ftml-autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
9088       \tl_if_empty:NF \l_stex_key_pts_tl{,data-ftml-problempoints={\l_stex_key_pts_tl}}

```

```

9089     }
9090     \noindent \stex_annotate:nn{data-ftml-title={}}{\_stex_annotate_force_break:n{\l_stex_k
9091     \tl_if_empty:NF \l_stex_key_title_tl {
9092         \exp_args:No \stexdoctitle \l_stex_key_title_tl
9093     }
9094     \_stex_annotate_force_break:n{}}
9095 }
9096 \tl_set_eq:NN \thistitle \l_stex_key_title_tl
9097 \tl_if_empty:NT \l_stex_key_pts_tl { \tl_set:Nn \l_stex_key_pts_tl{0} }
9098
9099
9100 \bool_set_true:N \l__problems_in_problem_bool
9101 \tl_set_eq:NN \l__problems_pts_tl \l_stex_key_pts_tl
9102 \tl_set_eq:NN \l__problems_min_tl \l_stex_key_min_tl
9103 \tl_if_eq:NnTF \l__problems_pts_tl {0}
9104     {\bool_set_false:N \l__problems_has_pts_bool}
9105     {\bool_set_true:N \l__problems_has_pts_bool}
9106 \tl_if_eq:NnTF \l__problems_min_tl {0}
9107     {\bool_set_false:N \l__problems_has_min_bool}
9108     {\bool_set_true:N \l__problems_has_min_bool}
9109 \int_gzero:N \g__problems_subproblem_int
9110
9111 \stex_if_do_html:F\stex_style_apply:
9112 \_stex_do_id:
9113 \__problems_activate_macros:
9114 }{
9115     \fp_gadd:Nn \g_problem_total_pts_fp { \l__problems_pts_tl}
9116     \fp_gadd:Nn \g_problem_total_min_fp { \l__problems_min_tl}
9117     \__problems_record_problem:
9118     \stex_if_do_html:F\stex_style_apply:
9119     \stex_if_do_html:T{ \end{stex_annotate_env} }
9120 }{
9121     \par\noindent\problemheader
9122     \stex_if_do_html:F{
9123         \bool_if:NT \c__problems_pts_bool {
9124             \tl_if_eq:NnF \l__problems_pts_tl {0}{
9125                 \marginpar{\l__problems_pts_tl}{~\problem@kw@pts\smallskip}
9126             }
9127         }
9128         \bool_if:NT \c__problems_min_bool {
9129             \tl_if_eq:NnF \l__problems_min_tl {0} {
9130                 \marginpar{\l__problems_min_tl}{~\problem@kw@minutes\smallskip}
9131             }
9132         }
9133     }
9134     \par
9135     \stex_ignore_spaces_and_pars:
9136 }{
9137     \par\bigskip
9138     % \bool_if:NT \c__problems_test_bool \pagebreak
9139 }{s}
9140
9141 \tl_set:Nn \problemheader {
9142     \stex_if_do_html:TF{

```

```

9143 \tl_if_empty:NF \thistitle {
9144 \stex_annotate:nn{data-ftml-title={}}{\textbf{\thistitle}}
9145 }
9146 }{
9147 \textbf{\sproblemautorefname{~}\thesproblem
9148 \tl_if_empty:NF \thistitle {
9149 {~}(\thistitle)
9150 }
9151 }
9152 }
9153 }
9154
9155 \cs_new_protected:Nn \__problems_record_problem: {
9156 \exp_args:Nne \iow_now:Nn \auxout {
9157 \problem@restore {\thesproblem}{\l__problems_pts_tl}{\l__problems_min_tl}
9158 }
9159 }
9160
9161 \cs_new_protected:Npn \problem@restore #1 #2 #3 {}

```

subproblem (env.)

```

9162 \int_new:N \g__problems_subproblem_int
9163
9164 \stex_new_stylable_env:nnnnnn {subproblem} {0}{}{
9165 \stex_keys_set:nn{problem}{#1}
9166 \bool_if:NF \l__problems_in_problem_bool{
9167 \ifstexhtml\else
9168 \par{\bfseries WARNING~subproblem~to~be~used~in~some~problem}\par
9169 \fi
9170 \__problems_activate_macros:
9171 \bool_set_true:N \l__problems_in_problem_bool
9172 \tl_set:Nn \l__problems_pts_tl{0}
9173 \tl_set:Nn \l__problems_min_tl{0}
9174 }
9175 \str_if_empty:NT \l_stex_key_name_str {
9176 \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
9177 \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
9178 }
9179 \stex_if_do_html:T {
9180 \str_if_empty:NT \l_stex_key_name_str {
9181 \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
9182 \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
9183 }
9184 \exp_args:Nne \begin{stex_annotate_env} {
9185 data-ftml-subproblem={\l_stex_key_name_str},
9186 %data-ftml-language={ \l_stex_current_language_str},
9187 data-ftml-autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
9188 \tl_if_empty:NF \l_stex_key_pts_tl{,data-ftml-problempoints={\l_stex_key_pts_tl}}
9189 }
9190 \noindent \stex_annotate:nn{data-ftml-title={}}{\_stex_annotate_force_break:n{\l_stex_k
9191 \tl_if_empty:NF \l_stex_key_title_tl {
9192 \exp_args:No \stexdoctitle \l_stex_key_title_tl
9193 }
9194 \tl_if_empty:NF \l_stex_key_title_tl {

```

```

9195     \exp_args:No \stexdoctitle \l_stex_key_title_tl
9196   }
9197   \_stex_annotate_force_break:n{}
9198 }
9199 \tl_if_empty:NT \l_stex_key_pts_tl { \tl_set:Nn \l_stex_key_pts_tl{0} }
9200 \int_gincr:N \g__problems_subproblem_int
9201 \bool_if:NF \l__problems_has_pts_bool {
9202   \tl_gset:Ne \l__problems_pts_tl {\fp_to_decimal:n {\l__problems_pts_tl + \l_stex_key_pt
9203 }
9204 \bool_if:NF \l__problems_has_min_bool {
9205   \tl_gset:Ne \l__problems_min_tl {\fp_to_decimal:n {\l__problems_min_tl + \l_stex_key_mi
9206 }
9207 \stex_if_smsmode:F {\stex_if_do_html:F \stex_style_apply: }
9208 }{
9209   \stex_if_do_html:TF{ \end{stex_annotate_env} }{\stex_if_smsmode:F\stex_style_apply:}
9210 }{
9211   \begin{list}{}{
9212     \setlength\topsep{0pt}
9213     \setlength\parsep{0pt}
9214     \setlength\rightmargin{0pt}
9215   }\item[\int_use:N \g__problems_subproblem_int .]
9216   \bool_if:NT \c__problems_pts_bool {
9217     \bool_if:NF \l__problems_has_pts_bool {
9218       \marginpar{\smallskip\l_stex_key_pts_tl{}}~\problem@kw@pts}
9219     }
9220   }
9221   \bool_if:NT \c__problems_min_bool {
9222     \bool_if:NF \l__problems_has_min_bool{
9223       \marginpar{\smallskip\l_stex_key_min_tl{}}~\problem@kw@minutes}
9224     }
9225   }
9226   \ignorespaces
9227 }{
9228   \end{list}
9229 }{}

```

\includeproblem

```

9230 \stex_keys_define:nnnn{ includeproblem }{
9231   \str_clear:N \l_stex_key_mhrepos_str
9232 }{
9233   archive .str_set:N = \l_stex_key_mhrepos_str,
9234   unknown .code:n = {}
9235 }{}
9236
9237 \NewDocumentCommand\includeproblem{0{ } m}{
9238   \group_begin:
9239   \tl_set:Nn \l_problem_inputproblem_keys_tl {#1}
9240   \stex_keys_set:nn{includeproblem}{#1}
9241   \exp_args:Nno \use:nn{\inputref[]\l_stex_key_mhrepos_str}{#2}
9242   \group_end:
9243 }
9244

```

(End of definition for \includeproblem. This function is documented on page 117.)

`solution (env.)`

```

9245 \int_new:N \g_problem_id_counter
9246 \dim_new:N \l_stex_key_testspace_dim
9247 \stex_keys_define:nnnn{ solution }{
9248   \str_clear:N \l_stex_key_answerclass_str
9249   \dim_zero:N \l_stex_key_testspace_dim
9250 }{
9251   testspace .dim_set:N = \l_stex_key_testspace_dim,
9252   answerclass .str_set:N = \l_stex_key_answerclass_str
9253 }{id,title,style}
9254
9255 \cs_new_protected:Nn \__problems_solution_start:n {
9256   \stex_keys_set:nn{ solution }{#1}
9257   \str_if_empty:NT \l_stex_key_id_str {
9258     \int_gincr:N \g_problem_id_counter
9259     \str_set:Nx \l_stex_key_id_str {
9260       SOLUTION_\int_use:N \g_problem_id_counter
9261     }
9262   }
9263   \stex_if_do_html:TF{
9264     \begin{stex_annotate_env}{
9265       data-ftml-solution=\l_stex_key_id_str,
9266       data-ftml-answerclass={\l_stex_key_answerclass_str}
9267     }
9268   }
9269 }
9270
9271 \stex_new_stylable_env:nnnnnnn { solution }{ 0{ } }{
9272   \stex_if_do_html:TF{
9273     \__problems_solution_start:n{#1}
9274   }{
9275     \ifsolutions
9276       \__problems_solution_start:n{#1}
9277       \stex_style_apply:
9278     \else
9279       \stex_keys_set:nn{ solution }{#1}
9280       \testspace{\l_stex_key_testspace_dim}
9281       \setbox\l_tmpa_box\vbox\bgroup
9282       \fi
9283     }
9284   }{
9285     \stex_if_do_html:TF{
9286       \end{stex_annotate_env}
9287     }{
9288       \ifsolutions
9289         \stex_style_apply:
9290         \stex_if_do_html:TF{
9291           \end{stex_annotate_env}
9292         }
9293       \else
9294         \egroup
9295       \fi
9296     }
9297   }{

```



```

9298 \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
9299 \noindent\emph{\problem@kw@solution\tl_if_empty:NF \l_stex_key_title_tl{
9300   {~}\l_stex_key_title_tl \str_if_empty:NF \l_stex_key_answerclass_str {
9301     (Answer Class: \l_stex_key_answerclass_str)
9302   }
9303 } :~}
9304 }{
9305 \par\rule[.3em]{\linewidth}{0.4pt}\newline
9306 }{ }
9307
9308 \stex_deactivate_macro:Nn \solution {sproblem-environments}

```

`\startsolutions`
`\stopsolutions`

```

9309 \cs_new_protected:Npn \startsolutions{
9310   \global\solutionstrue
9311 }
9312 \cs_new_protected:Npn \stopsolutions{
9313   \global\solutionsfalse
9314 }

```

(End of definition for `\startsolutions` and `\stopsolutions`. These functions are documented on page 114.)

`hint (env.)`

```

9315
9316 \stex_keys_define:nnnn{ problemenv }{ }{id,title,style}
9317
9318 \cs_new_protected:Nn \__problems_hint_start:n {
9319   \stex_keys_set:nn{ problemenv }{#1}
9320   \str_if_empty:NT \l_stex_key_id_str {
9321     \int_gincr:N \g_problem_id_counter
9322     \str_set:Nx \l_stex_key_id_str {
9323       HINT_\int_use:N \g_problem_id_counter
9324     }
9325   }
9326   \stex_if_do_html:TF{
9327     \begin{stex_annotate_env}{
9328       data-ftml-problemhint=\l_stex_key_id_str
9329     }
9330   }
9331 }
9332
9333 \stex_new_stylable_env:nnnnnnn { hint }{ 0{ } }{
9334   \stex_if_do_html:TF{
9335     \__problems_hint_start:n{#1}
9336   }{
9337     \bool_if:NTF \c__problems_hints_bool {
9338       \__problems_hint_start:n{#1}
9339       \stex_style_apply:
9340     }{
9341       \setbox\l_tmpa_box\vbox\bgroup
9342     }
9343   }
9344 }{

```

```

9345 \stex_if_do_html:TF{
9346   \end{stex_annotate_env}
9347 }{
9348   \bool_if:NTF \c__problems_hints_bool {
9349     \stex_style_apply:
9350     \stex_if_do_html:T{
9351       \end{stex_annotate_env}
9352     }
9353   }{
9354     \egroup
9355   }
9356 }
9357 }{
9358   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
9359   \noindent\emph{\problem@kw@hint\tl_if_empty:NF \l_stex_key_title_tl{
9360     {~}\l_stex_key_title_tl
9361   } :~}
9362 }{
9363   \par\rule[.3em]{\linewidth}{0.4pt}\newline
9364 }{}
9365 \stex_deactivate_macro:Nn \hint {sproblem-environments}

```

exnote (env.)

```

9366 \cs_new_protected:Nn \__problems_exnote_start:n {
9367   \stex_keys_set:nn{ problemenv }{#1}
9368   \str_if_empty:NT \l_stex_key_id_str {
9369     \int_gincr:N \g_problem_id_counter
9370     \str_set:Nx \l_stex_key_id_str {
9371       EXNOTE_\int_use:N \g_problem_id_counter
9372     }
9373   }
9374   \stex_if_do_html:T{
9375     \begin{stex_annotate_env}{
9376       data-ftml-problemnote=\l_stex_key_id_str
9377     }
9378   }
9379 }
9380
9381 \stex_new_stylable_env:nnnnnn { exnote }{ 0{} }{
9382   \stex_if_do_html:TF{
9383     \__problems_exnote_start:n{#1}
9384   }{
9385     \bool_if:NTF \c__problems_notes_bool {
9386       \__problems_exnote_start:n{#1}
9387       \stex_style_apply:
9388     }{
9389       \setbox\l_tmpa_box\vbox\bgroup
9390     }
9391   }
9392 }{
9393   \stex_if_do_html:TF{
9394     \end{stex_annotate_env}
9395   }{
9396     \bool_if:NTF \c__problems_notes_bool {

```

```

9397     \stex_style_apply:
9398     \stex_if_do_html:T{
9399         \end{stex_annotate_env}
9400     }
9401 }{
9402     \egroup
9403 }
9404 }
9405 }{
9406     \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
9407     \noindent\emph{\problem@kw@note\tl_if_empty:NF \l_stex_key_title_tl{
9408         {~}\l_stex_key_title_tl
9409     } :~}
9410 }{
9411     \par\rule[.3em]{\linewidth}{0.4pt}\newline
9412 }{}
9413 \stex_deactivate_macro:Nn \exnote {sproblem~environments}

```

gnote (env.)

```

9414 \int_new:N \l__problems_anscls_int
9415
9416 \cs_new_protected:Nn \__problems_gnote_start:n {
9417     \stex_keys_set:nn{ problemenv }{#1}
9418     \str_if_empty:NT \l_stex_key_id_str {
9419         \int_gincr:N \g_problem_id_counter
9420         \str_set:Nx \l_stex_key_id_str {
9421             GNOTE_\int_use:N \g_problem_id_counter
9422         }
9423     }
9424
9425     \stex_if_do_html:TF{
9426         \begin{stex_annotate_env}{
9427             data-ftml-problemgnote=\l_stex_key_id_str
9428         }
9429     }
9430     \stex_style_apply:
9431 }
9432
9433 \stex_new_stylable_env:nnnnnn { gnote }{ 0{} }{
9434     \stex_if_do_html:TF{
9435         \__problems_gnote_start:n{#1}
9436     }{
9437         \bool_if:NTF \c__problems_gnotes_bool {
9438             \__problems_gnote_start:n{#1}
9439         }{
9440             \setbox\l_tmpa_box\vbox\bgroup
9441         }
9442     }
9443     \stex_reactivate_macro:N \anscls
9444 }{
9445     \stex_if_do_html:TF{
9446         \end{stex_annotate_env}
9447     }{
9448         \bool_if:NTF \c__problems_gnotes_bool {

```

```

9449     \stex_style_apply:
9450     \stex_if_do_html:T{
9451         \end{stex_annotate_env}
9452     }
9453     }{
9454         \egroup
9455     }
9456 }
9457 }{
9458     \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
9459     \noindent\emph{\problem@kw@grading\str_if_empty:NF \l_stex_key_title_tl{
9460         {~}\l_stex_key_title_tl
9461     } :~}
9462 }{
9463     \par\rule[.3em]{\linewidth}{0.4pt}\newline
9464 }{}
9465 \stex_deactivate_macro:Nn \gnote {sproblem-environments}
9466
9467
9468 \stex_keys_define:nnnn{ anscls }{
9469     \str_clear:N \l_stex_key_pts_str
9470     \tl_clear:N \l_stex_key_feedback_tl
9471 }{
9472     pts      .str_set:N = \l_stex_key_pts_str,
9473     feedback .tl_set:N = \l_stex_key_feedback_tl,
9474     update   .code:n    = {}
9475 }{id}
9476 \newcommand \anscls [2] [] {
9477     \stex_keys_set:nn{ anscls }{#1}
9478     \str_if_empty:NT \l_stex_key_id_str {
9479         \int_incr:N \l__problems_anscls_int
9480         \str_set:Nx \l_stex_key_id_str {
9481             AC\int_use:N \l__problems_anscls_int
9482         }
9483     }
9484     \stex_if_do_html:TF{
9485         \begin{list}{}{
9486             \setlength\topsep{0pt}
9487             \setlength\parsep{0pt}
9488             \setlength\rightmargin{0pt}
9489         }\item[] \exp_args:Ne \stex_annotate:nn{
9490             data-ftml-answerclass={\l_stex_key_id_str}
9491             \str_if_empty:NF \l_stex_key_pts_str{
9492                 ,data-ftml-answerclass-pts={\l_stex_key_pts_str}
9493             }
9494         }{
9495             #2~
9496             \tl_if_empty:NF \l_stex_key_feedback_tl{
9497                 \stex_annotate:nn{
9498                     data-ftml-answerclass-feedback={}
9499                 }{\l_stex_key_feedback_tl}
9500             }
9501         }
9502     } \end{list}

```

```

9503 }{
9504   \begin{list}{}{
9505     \setlength\topsep{0pt}
9506     \setlength\parsep{0pt}
9507     \setlength\rightmargin{0pt}
9508   }\item[\l_stex_key_id_str] #2
9509     \str_if_empty:NF \l_stex_key_pts_str {\par
9510       ~ \problem@kw@points :~\l_stex_key_pts_str
9511     }
9512     \tl_if_empty:NF \l_stex_key_feedback_tl {\par
9513       ~ \problem@kw@feedback :~\l_stex_key_feedback_tl
9514     }
9515   \end{list}
9516 }
9517 }
9518 \stex_deactivate_macro:Nn \anscls {gnote-environments}

```

The margin pars are reader-visible, so we need to translate

```

9519 \def\pts#1{
9520   \bool_if:NT \c__problems_pts_bool {
9521     \stex_annotate:nn{data-ftml-problempoints={#1}}{\marginpar{#1~\problem@kw@pts}}
9522   }\hbox_unpack:N\c_empty_box
9523 }
9524 \def\min#1{
9525   \bool_if:NT \c__problems_min_bool {
9526     \stex_annotate:nn{data-ftml-problemminutes={}}{\marginpar{#1~\problem@kw@minutes}}
9527   }\hbox_unpack:N\c_empty_box
9528 }

```

mcb (*env.*)

```

9529 \stex_new_stylable_env:nnnnnnn{mcb}{0}{0}{
9530   \stex_keys_set:nn{style}{#1}
9531   \cs_set:Nn \__problems_mccline:n{
9532     \begin{list}{}{
9533       \setlength\topsep{0pt}
9534       \setlength\parsep{0pt}
9535       \setlength\rightmargin{0pt}
9536     }\item[\problem_mcc_box_tl] ##1 \end{list}
9537   }
9538
9539   \stex_if_do_html:T{
9540     \tl_set:Nn\problem_mcc_box_tl{}
9541     \__problems_maybe_inline:n{
9542       \exp_args:Nne \begin{stex_annotate_env}{
9543         data-ftml-multiple-choice-block={}
9544         \clist_if_empty:NF \l_stex_key_style_clist {,
9545           data-ftml-styles={\l_stex_key_style_clist}
9546         }
9547       }
9548     }
9549     \clist_if_in:NnTF \l_stex_key_style_clist {inline} {
9550       \cs_set:Nn \__problems_mccline:n {##1}
9551     }{
9552       \clist_if_in:NnT \l_stex_key_style_clist {dropdown} {

```

```

9553     \cs_set:Nn \__problems_mccline:n {##1}
9554   }
9555 }
9556 }
9557 \stex_deactivate_macro:Nn \mcb {sproblem~environments}
9558 \stex_deactivate_macro:Nn \scb {sproblem~environments}
9559 \stex_deactivate_macro:Nn \solution {sproblem~environments}
9560 \stex_deactivate_macro:Nn \hint {sproblem~environments}
9561 \stex_deactivate_macro:Nn \exnote {sproblem~environments}
9562 \stex_deactivate_macro:Nn \gnote {sproblem~environments}
9563 \stex_reactivate_macro:N \mcc
9564 \stex_if_do_html:F \stex_style_apply:
9565 }{
9566   \stex_if_do_html:TF{
9567     \__problems_maybe_inline:n{\end{stex_annotate_env}}
9568   }\stex_style_apply:
9569 }{\par}{\}{}
9570
9571 \stexstylemcb[inline]{
9572   {\Large[]\cs_set:Nn \__problems_mccline:n{\problem_mcc_box_tl{~} #1}
9573 }{\{\Large[]}}
9574
9575 \stexstylemcb[dropdown]{
9576   {\Large[]\cs_set:Nn \__problems_mccline:n{\problem_mcc_box_tl{~} #1}
9577 }{\{\Large[]}}
9578
9579 \stex_deactivate_macro:Nn \mcb {sproblem~environments}
we define the keys for the mcc macro
9580 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
9581   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
9582     \bool_set_true:N #1
9583   }{
9584     \bool_set_false:N #1
9585   }
9586 }
9587 \stex_keys_define:nnnn{mcc}{
9588   \tl_clear:N \l_stex_key_feedback_tl
9589   \bool_set_false:N \l_stex_key_T_bool
9590   \tl_clear:N \l_stex_key_Ttext_tl
9591   \tl_clear:N \l_stex_key_Ftext_tl
9592 }{
9593   feedback .tl_set:N      = \l_stex_key_feedback_tl ,
9594   T         .code:n       = {\bool_set_true:N \l_stex_key_T_bool} ,
9595   F         .code:n       = {\bool_set_false:N \l_stex_key_T_bool} ,
9596   Ttext     .tl_set:N     = \l_stex_key_Ttext_tl ,
9597   Ftext     .tl_set:N     = \l_stex_key_Ftext_tl ,
9598 }{id}
9599
\mcc
9600
9601 \cs_set:Nn \__problems_maybe_newline: { \\\ }
9602

```

```

9603 \stex_if_html_backend:TF{
9604   \tl_set:Nn \problem_mcc_box_tl {
9605     \ltx@ifpackageloaded{amssymb}{\square$}{
9606       \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
9607     }
9608   }
9609
9610   \newcommand\mcc[2][]{
9611     \stex_keys_set:nn{mcc}{#1}
9612     \__problems_mccline:n{
9613       \stex_if_do_html:T{
9614         \stex_annotate:nn{data-ftml-problem-choice={
9615           \bool_if:NTF \l_stex_key_T_bool {true}{false}
9616         }}{
9617           #2~
9618           \stex_annotate:nn{data-ftml-problem-choice-verdict={}}{
9619             \bool_if:NTF \l_stex_key_T_bool {
9620               \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
9621             }{
9622               \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
9623             }
9624           }
9625           \tl_if_empty:NF \l_stex_key_feedback_tl {
9626             \stex_annotate:nn{data-ftml-problem-choice-feedback={}}{
9627               \l_stex_key_feedback_tl
9628             }
9629           }
9630         }
9631       }
9632     }
9633   }
9634 }{
9635   \tl_set:Nn \problem_mcc_box_default_tl {
9636     \ltx@ifpackageloaded{amssymb}{\square$}{
9637       \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
9638     }
9639   }
9640   \tl_set:Nn \problem_mcc_box_tl {
9641     \ifsolutions
9642       \bool_if:NTF \l_stex_key_T_bool {
9643         \makebox[0pt][l]{\problem_mcc_box_default_tl}
9644         \hspace{0.1em}$\cs_if_exist:NTF\checkmark{
9645           \raisebox{.15ex}{\checkmark}
9646         } X$
9647       }{\problem_mcc_box_default_tl}
9648     \else
9649       \problem_mcc_box_default_tl
9650     \fi
9651   }
9652   \newcommand\mcc[2][]{
9653     \stex_keys_set:nn{mcc}{#1}
9654     \ifsolutions
9655       \tl_set:Nx \l_tmpa_tl{
9656         \bool_if:NTF \l_stex_key_T_bool {

```

```

9657     \tl_if_empty:NF \l_stex_key_Ttext_tl {\exp_not:N \l_stex_key_Ttext_tl}
9658   }{
9659     \tl_if_empty:NF \l_stex_key_Ftext_tl {\exp_not:N \l_stex_key_Ftext_tl}
9660   }
9661   \tl_if_empty:NF \l_stex_key_feedback_tl {
9662     \exp_not:n{\|\emph{\l_stex_key_feedback_tl}}
9663   }
9664 }
9665 \fi
9666
9667 \__problems_mccline:n{ #2
9668   \ifsolutions
9669     \tl_if_empty:NF \l_tmpa_tl { \footnote{\l_tmpa_tl} }
9670   \fi
9671 }
9672 }
9673 }
9674 \stex_deactivate_macro:Nn \mcc {mcb~environments}

```

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

scb (env.)

```

9675
9676 \cs_new_protected:Nn \__problems_maybe_inline:n {
9677   \clist_if_in:NnTF \l_stex_key_style_clist {inline} {
9678     \let\__problems_oldpar:\stex_par:
9679     \cs_set:Nn\stex_par:{~}
9680     \cs_set:Nn\__problems_maybe_newline:{}
9681     #1
9682     \let\stex_par:\__problems_oldpar:
9683   }{
9684     \clist_if_in:NnTF \l_stex_key_style_clist {dropdown} {
9685       \let\__problems_oldpar:\stex_par:
9686       \cs_set:Nn\stex_par:{~}
9687       \cs_set:Nn\__problems_maybe_newline:{}
9688       #1
9689       \let\stex_par:\__problems_oldpar:
9690     }{\par #1}
9691   }
9692 }
9693
9694 \stex_new_stytable_env:nnnnnn{scb}{0}{}{
9695   \stex_keys_set:nn{style}{#1}
9696   \cs_set:Nn \__problems_sccline:n{
9697     \begin{list}{}{
9698       \setlength\topsep{0pt}
9699       \setlength\parsep{0pt}
9700       \setlength\rightmargin{0pt}
9701     }\item[\problem_scc_box_tl] ##1 \end{list}
9702   }
9703
9704   \stex_if_do_html:T{
9705     \__problems_maybe_inline:n{
9706       \exp_args:Ne\stex_annotate_env{

```



```

9707     data-ftml-single-choice-block={}
9708     \clist_if_empty:NF \l_stex_key_style_clist {,
9709         data-ftml-styles={\l_stex_key_style_clist}
9710     }
9711 }
9712 }
9713 \tl_set:Nn\problem_scc_box_tl{}
9714 \clist_if_in:NnTF \l_stex_key_style_clist {inline} {
9715     \cs_set:Nn \__problems_sccline:n {##1}
9716 }{
9717     \clist_if_in:NnT \l_stex_key_style_clist {dropdown} {
9718         \cs_set:Nn \__problems_sccline:n {##1}
9719     }
9720 }
9721 }
9722 \stex_deactivate_macro:Nn \mcb {sproblem~environments}
9723 \stex_deactivate_macro:Nn \scb {sproblem~environments}
9724 \stex_deactivate_macro:Nn \solution {sproblem~environments}
9725 \stex_deactivate_macro:Nn \hint {sproblem~environments}
9726 \stex_deactivate_macro:Nn \exnote {sproblem~environments}
9727 \stex_deactivate_macro:Nn \gnote {sproblem~environments}
9728 \stex_reactivate_macro:N \scc
9729 \stex_if_do_html:F \stex_style_apply:
9730 }{
9731     \stex_if_do_html:TF{
9732         \__problems_maybe_inline:n { \endstex_annotate_env }
9733     }\stex_style_apply:
9734 }{\par}{\par}
9735
9736 \stexstylescb[inline]{
9737     {\Large[]\cs_set:Nn \__problems_sccline:n{\problem_scc_box_tl{~} #1}
9738 }{\{\Large[]}}
9739
9740 \stexstylescb[dropdown]{
9741     {\Large[]\cs_set:Nn \__problems_sccline:n{\problem_scc_box_tl{~} #1}
9742 }{\{\Large[]}}
9743
9744 \stex_deactivate_macro:Nn \scb {sproblem~environments}

```

\scc

```

9745
9746 \stex_if_html_backend:TF{
9747     \tl_set:Nn\problem_scc_box_tl{${\bigcirc}}
9748     \newcommand\scc[2][]{
9749         \stex_keys_set:nn{mcc}{#1}
9750         \__problems_sccline:n{
9751             \stex_if_do_html:T{
9752                 \stex_annotate:nn{data-ftml-problem-choice={
9753                     \bool_if:NTF \l_stex_key_T_bool {true}{false}
9754                 }}{
9755                     #2~
9756                 \stex_annotate:nn{data-ftml-problem-choice-verdict={}}{
9757                     \bool_if:NTF \l_stex_key_T_bool {
9758                         \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_t

```

```

9759         }{
9760         \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
9761         }
9762     }
9763     \tl_if_empty:NF \l_stex_key_feedback_tl {
9764         \stex_annotate:nn{data-ftml-problem-choice-feedback={}}{
9765             \l_stex_key_feedback_tl
9766         }
9767     }
9768 }
9769 }
9770 }
9771 }
9772 }{
9773 \tl_set:Nn\problem_scc_box_default_tl{${\bigcirc}$}
9774 \tl_set:Nn\problem_scc_box_tl{
9775     \ifsolutions
9776     \bool_if:NTF \l_stex_key_T_bool {
9777         \makebox[Opt][l]{\problem_scc_box_default_tl}
9778         \hspace{0.1em}${\cs_if_exist:NTF\checkmark{
9779             \raisebox{.15ex}{\checkmark}
9780         }} X$
9781     }{\problem_scc_box_default_tl}
9782     \else
9783     \problem_scc_box_default_tl
9784     \fi
9785 }
9786 \newcommand\scc[2][]{
9787     \stex_keys_set:nn{mcc}{#1}
9788     \ifsolutions
9789     \tl_set:Nx \l_tmpa_tl{
9790         \bool_if:NTF \l_stex_key_T_bool {
9791             \tl_if_empty:NF \l_stex_key_Ttext_tl {\exp_not:N \l_stex_key_Ttext_tl}
9792         }{
9793             \tl_if_empty:NF \l_stex_key_Ftext_tl {\exp_not:N \l_stex_key_Ftext_tl}
9794         }
9795         \tl_if_empty:NF \l_stex_key_feedback_tl {
9796             \exp_not:n{ \\emph{\l_stex_key_feedback_tl} }
9797         }
9798     }
9799     \fi
9800
9801     \__problems_sccline:n{ #2
9802     \ifsolutions
9803     \tl_if_empty:NF \l_tmpa_tl { \footnote{\l_tmpa_tl} }
9804     \fi
9805 }
9806 }
9807 }
9808
9809 \stex_deactivate_macro:Nn \scc {scb-environments}
9810
9811
9812 \newcommand\yesTnoF{

```

```

9813 \begin{scb}[style=inline]
9814 \scc[T]{yes}~\scc[F]{no}
9815 \end{scb}
9816 }
9817 \newcommand\yesFnoT{
9818 \begin{scb}[style=inline]
9819 \scc[F]{yes}~\scc[T]{no}
9820 \end{scb}
9821 }
9822 \newcommand\trueTfalseF{
9823 \begin{scb}[style=inline]
9824 \scc[T]{true}~\scc[F]{false}
9825 \end{scb}
9826 }
9827 \newcommand\trueFfalseT{
9828 \begin{scb}[style=inline]
9829 \scc[F]{true}~\scc[T]{false}
9830 \end{scb}
9831 }

```

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

\fillinsol

```

9832 \stex_keys_define:nnnn{fillinsol}{
9833 \tl_clear:N \l__problems_fillin_solution_tl
9834 \dim_zero:N \l_stex_key_testspace_dim
9835 }{
9836 \testspace .dim_set:N = \l_stex_key_testspace_dim,
9837 \exact .code:n = {\__problems_parse_fillin_arg:nnnn{exact}#1 },
9838 \numrange .code:n = {\__problems_parse_fillin_arg:nnnn{numrange}#1 },
9839 \regex .code:n = {\__problems_parse_fillin_arg:nnnn{regex}#1 }
9840 }{}
9841
9842 \tl_const:Nn \c__problems_true_tl{T}
9843 \tl_const:Nn \c__problems_false_tl{F}
9844
9845 \msg_set:nnn{problem}{error/neither-true-nor-false}{
9846 \Fillinsol~verdict~#1~is~neither~T~nor~F!
9847 }
9848
9849 \stex_if_html_backend:TF{
9850 \cs_new:Nn \__problems_parse_fillin_arg:nnnn {
9851 \tl_set:Nn \l_tmpa_tl{#3}
9852 \tl_if_eq:NNF \l_tmpa_tl\c__problems_true_tl{
9853 \tl_if_eq:NNF \l_tmpa_tl\c__problems_false_tl {
9854 \msg_error:nnn{problem}{error/neither-true-nor-false}{#3}
9855 }
9856 }
9857 \tl_put_right:Ne \l__problems_fillin_solution_tl {
9858 \stex_annotate:nn{
9859 data-ftml-fillin-case={#1},
9860 data-ftml-fillin-case-value={\tl_to_str:n{#2}},
9861 data-ftml-fillin-case-verdict={
9862 \tl_if_eq:NNTF\l_tmpa_tl\c__problems_true_tl{true}{false}

```

```

9863     },
9864     }{\stex_annotate_force_break:n{\exp_not:n{#4}}}}
9865   }
9866 }
9867 }{
9868   \cs_new:Nn \__problems_parse_fillin_arg:nnnn {
9869     \tl_put_right:Nn \l__problems_fillin_solution_tl {
9870       #1 & \tl_to_str:n{#2} & #3 & #4 \crr
9871     }
9872   }
9873 }
9874
9875
9876 \newcommand\fillinsol[2][{}{
9877   \stex_if_do_html:F \quad
9878   \mode_if_math:TF{
9879     \hbox{\__problems_fillinsol:nn{#1}{#2}}
9880   }{
9881     \__problems_fillinsol:nn{#1}{#2}
9882   }
9883   \stex_if_do_html:F \quad
9884 }
9885
9886 \stex_if_html_backend:TF{
9887   \cs_new_protected:Nn \__problems_fillinsol:nn {
9888     \stex_keys_set:nn{fillinsol}{#1}
9889     \tl_if_empty:nF{#2}{
9890       \__problems_parse_fillin_arg:nnnn{exact}{#2}{T}{}
9891     }
9892     \hbox_set:Nn \l_tmpa_box{\fbox{\huge{\texttt{\tl_to_str:n{#2}}}\hskip\l_stex_key_testsp
9893     \exp_args:Ne \stex_annotate:nn{
9894       data-ftml-fillinsol={},
9895       data-ftml-fillinsol-width={\dim_to_decimal:n{1.5 \box_wd:N \l_tmpa_box }}
9896     }{
9897       \stex_annotate_force_break:n{
9898         \l__problems_fillin_solution_tl
9899         #2
9900       }
9901     }
9902   }
9903 }{
9904   \cs_new_protected:Nn \__problems_fillinsol:nn {
9905     \stex_keys_set:nn{fillinsol}{#1}
9906     \ifsolutions
9907
9908     \cs_if_exist:NT\textcolor{\textcolor{red}}{\fbox{\texttt{\tl_to_str:n{#2}}}}
9909     \tl_if_empty:NF \l__problems_fillin_solution_tl {
9910       \footnote{
9911         \halign{ ~~~~\hfil & ~~~~\hfil & ~~~~\hfil & ~~~~\hfil \cr
9912           \textbf{type}&\textbf{case}&\textbf{verdict}&\textbf{feedback}\cr
9913           \tl_if_empty:nF{#2}{
9914             exact & #2 & T & \cr
9915           }
9916           \l__problems_fillin_solution_tl

```

```

9917     }
9918   }
9919 }
9920 \else
9921   \fbox{\dim_compare:nNnTF\l_stex_key_testspace_dim={0pt}{
9922     \phantom{\huge\tl_to_str:n{#2}}}
9923   }{
9924     \hspace{\l_stex_key_testspace_dim}\vphantom{\huge{A \tl_to_str:n{#2}}}}
9925   }}
9926 \fi
9927 }
9928 }
9929
9930 \stex_deactivate_macro:Nn \fillinsol {sproblem-environments}

```

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

`\testemptypage`

```

9931 \newcommand\testemptypage[1][\%
9932 \bool_if:NT \c__problems_test_bool {\vfill\begin{center}\hwexam@kw@testemptypage\end{center}}
9933 }

```

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

`\testspace`

```

9934 \newcommand\testspace[1]{\bool_if:NT \c__problems_test_bool {\vspace*{#1}}}
9935 \newcommand\testsmallspace{\testspace{1cm}}
9936 \newcommand\testmedspace{\testspace{2cm}}
9937 \newcommand\testbigspace{\testspace{3cm}}

```

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

`\testnewpage`

```

9938 \newcommand\testnewpage{\bool_if:NT \c__problems_test_bool {\newpage}}

```

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

`\testnewpage`

```

9939 \newcommand{\testnewpageInProblem}%
9940 {\ifintest\vfill\begin{center}\emph{continued-on-next-page}\end{center}\testnewpage\fi}%

```

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

```

9941 \end{package}

```

39.3 Implementation: The hwexam Package

39.3.1 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.

```

9942 \*package
9943 \ProvidesExplPackage{hwexam}{2026/06/24}{4.1.0}{homework assignments and exams}

```

```

9944 \RequirePackage{l3keys2e}
9945
9946 \keys_define:nn {hwexam / pkg}{
9947   multiple .default:n = { false },
9948   multiple .bool_set:N = \c_hwexam_multiple_bool,
9949   qrcode .default:n = { false },
9950   qrcode .bool_set:N = \c_hwexam_qrcode_bool,
9951   unknown .code:n = {
9952     \PassOptionsToPackage{\CurrentOption}{problem}
9953   }
9954 }
9955 \ProcessKeysOptions{ hwexam /pkg }
9956 \RequirePackage{problem}

```

\hwexam_kw_* For multilinguality, we define internal macros for keywords that can be specialized in *.ldf files.

```

9957 \AddToHook{begindocument}{
9958   \ExplSyntaxOn\makeatletter
9959   \input{hwexam-english.ldf}
9960   \ltx@ifpackageloaded{babel}{
9961     \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
9962     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
9963       \input{hwexam-ngerman.ldf}
9964     }
9965     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
9966       \input{hwexam-finnish.ldf}
9967     }
9968     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
9969       \input{hwexam-french.ldf}
9970     }
9971     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
9972       \input{hwexam-russian.ldf}
9973     }
9974   }{}
9975   \makeatother\ExplSyntaxOff
9976 }

```

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

39.3.2 QR Codes

```

9977 \group_begin:
9978   \escapechar=-1
9979   \xdef\__qr_backslash{\string\}
9980 \group_end:
9981
9982 \bool_if:NT \c_hwexam_qrcode_bool {
9983   \RequirePackage{qrcode}
9984   \RequirePackage{marginnote}
9985   \str_new:N \g__qr_json_str
9986   \bool_new:N \g__qr_in_problems_json_bool
9987   \bool_set_false:N \g__qr_in_problems_json_bool
9988   \bool_new:N \g__qr_in_subproblems_json_bool
9989   \bool_set_false:N \g__qr_in_subproblems_json_bool

```

```

9990 \bool_new:N \g_@@_qr_in_anscls_json_bool
9991 \bool_set_false:N \g_@@_qr_in_anscls_json_bool
9992 \bool_if:NTF \c__problems_gnotes_bool {
9993   \gdef \qrjson { \str_gput_right:Nx \g_@@_qr_json_str}
9994 }{
9995   \gdef \qrjson #1 {}
9996 }
9997 \qrjson{[]}
9998 \AtEndDocument{\qrjson{}}
9999 \def\__qr_escape_char:n #1 {
10000   #1\exp_args:Nno\use:nn{\if_charcode:w#1}\__qr_backslash#1\fi
10001 }
10002
10003 \gdef\__qr_escape:n #1{\exp_args:Ne\str_map_function:nN{\tl_to_str:n{#1}}\__qr_escape_
10004 \gdef \__qr_escape:o #1{\exp_args:No\__qr_escape:n#1}
10005
10006 \def\qrschema{T0:D0:\examnumber}
10007
10008 \bool_if:NTF \c__problems_test_bool {
10009   \def\doproblemqr{
10010     \ifstexhtml\else{
10011       \reversemarginpar\marginnote{\qrcline[height=1.5cm]{\qrschema}}
10012       \normalmarginpar
10013     }\fi
10014   }
10015   \def\insertexamnumber{
10016     \ifstexhtml\else
10017       \tl_if_exist:NTF \examnumber {
10018         {\Large\bfseries ID:~\examnumber}
10019       }{
10020         {\color{red}\Large\bfseries!!!~ WARNING:~NO~{\string\examnumber}~SET~!!!}\gdef\examnum
10021       }
10022       \global\def\insertexamnumber{}
10023     \fi
10024   }
10025 }{
10026   \def\doproblemqr{}
10027   \def\insertexamnumber{}
10028 }
10029
10030 \stexstyleproblem[noqr]{
10031   \par\noindent\problemheader \xdef\grid{\thesproblem}
10032   \bool_if:NT \c__problems_pts_bool {
10033     \tl_if_eq:NnF \l__problems_pts_tl {0}{
10034       \marginpar{\l__problems_pts_tl}{~\problem@kw@points\smallskip}
10035     }
10036   }
10037   \bool_if:NT \c__problems_min_bool {
10038     \tl_if_eq:NnF \l__problems_min_tl {0}{
10039       \marginpar{\l__problems_min_tl}{~\problem@kw@minutes\smallskip}
10040     }
10041   }
10042
10043   \bool_if:NTF \g_@@_qr_in_problems_json_bool {

```

```

10044 \qrjson{,}
10045 }{
10046 \bool_gset_true:N \g_@@_qr_in_problems_json_bool
10047 }
10048
10049 \qrjson {
10050 \c_left_brace_str
10051 "id": "\_@@_qr_escape:o\l_stex_key_id_str", "title": "\_@@_qr_escape:o\l_stex_key_title_tl
10052 "number": "\thesproblem"
10053 }
10054
10055 \tl_if_eq:NnF \l__problems_pts_tl {0}{
10056 \qrjson {
10057 , "pts": \l__problems_pts_tl
10058 }
10059 }
10060 \par
10061 \stex_ignore_spaces_and_pars:
10062 }{
10063 \bool_if:NT \g_@@_qr_in_subproblems_json_bool {
10064 \qrjson {}}
10065 }
10066 \bool_gset_false:N \g_@@_qr_in_subproblems_json_bool
10067 \qrjson {\c_right_brace_str}
10068 \par\bigskip
10069 }
10070
10071 \stexstyleproblem{
10072 \par\noindent\problemheader \xdef\grid{\thesproblem}
10073 \doproblemqr
10074 \bool_if:NT \c__problems_pts_bool {
10075 \tl_if_eq:NnF \l__problems_pts_tl {0}{
10076 \marginpar{\l__problems_pts_tl}{~\problem@kw@points\smallskip}
10077 }
10078 }
10079 \bool_if:NT \c__problems_min_bool {
10080 \tl_if_eq:NnF \l__problems_min_tl {0} {
10081 \marginpar{\l__problems_min_tl}{~\problem@kw@minutes\smallskip}
10082 }
10083 }
10084
10085 \bool_if:NTF \g_@@_qr_in_problems_json_bool {
10086 \qrjson{,}
10087 }{
10088 \bool_gset_true:N \g_@@_qr_in_problems_json_bool
10089 }
10090
10091 \qrjson {
10092 \c_left_brace_str
10093 "id": "\_@@_qr_escape:o\l_stex_key_id_str", "title": "\_@@_qr_escape:o\l_stex_key_title_tl
10094 "number": "\thesproblem"
10095 }
10096
10097 \tl_if_eq:NnF \l__problems_pts_tl {0}{

```



```

10098 \qrjson {
10099   , "pts": \l__problems_pts_tl
10100 }
10101 }
10102 \par
10103 \stex_ignore_spaces_and_pars:
10104 }{
10105   \bool_if:NT \g_@@_qr_in_subproblems_json_bool {
10106     \qrjson {}
10107   }
10108   \bool_gset_false:N \g_@@_qr_in_subproblems_json_bool
10109   \qrjson {\c_right_brace_str}
10110   \par\bigskip
10111 }
10112
10113 \stexstylesubproblem[noqr]{
10114   \begin{list}{}{
10115     \setlength\topsep{0pt}
10116     \setlength\parsep{0pt}
10117     \setlength\rightmargin{0pt}
10118   } \item[\int_use:N \g__problems_subproblem_int .]
10119   \xdef\grid{\thesproblem.\int_use:N \g__problems_subproblem_int}
10120   \bool_if:NT \c__problems_pts_bool {
10121     \bool_if:NF \l__problems_has_pts_bool {
10122       \marginpar{\smallskip\l_stex_key_pts_tl}{~\problem@kw@points}
10123     }
10124   }
10125   \bool_if:NT \c__problems_min_bool {
10126     \bool_if:NF \l__problems_has_min_bool{
10127       \marginpar{\smallskip\l_stex_key_min_tl}{~\problem@kw@minutes}
10128     }
10129   }
10130   \bool_if:NTF \g_@@_qr_in_subproblems_json_bool {
10131     \qrjson {,}
10132   }{
10133     \qrjson {
10134       , "subproblems": [
10135       ]
10136       \bool_gset_true:N \g_@@_qr_in_subproblems_json_bool
10137     }
10138     \qrjson {
10139       \c_left_brace_str
10140       "id": "\_@@_qr_escape:o\l_stex_key_id_str", "title": "\_@@_qr_escape:o\l_stex_key_title_tl
10141       "number": "\thesproblem.\int_use:N \g__problems_subproblem_int"
10142     }
10143     \bool_if:NF \l__problems_has_pts_bool {
10144       \qrjson {
10145         , "pts": \l_stex_key_pts_tl
10146       }
10147     }
10148   }{
10149     \qrjson {\c_right_brace_str}
10150   \end{list}
10151 }

```

```

10152
10153 \stexstylesubproblem{
10154   \begin{list}{}{
10155     \setlength\topsep{0pt}
10156     \setlength\parsep{0pt}
10157     \setlength\rightmargin{0pt}
10158   }\item[\int_use:N \g__problems_subproblem_int .]
10159   \xdef\grid{\thesproblem.\int_use:N \g__problems_subproblem_int}
10160   \doproblemqr
10161   \bool_if:NT \c__problems_pts_bool {
10162     \bool_if:NF \l__problems_has_pts_bool {
10163       \marginpar{\smallskip\l_stex_key_pts_tl}{~\problem@kw@points}
10164     }
10165   }
10166   \bool_if:NT \c__problems_min_bool {
10167     \bool_if:NF \l__problems_has_min_bool{
10168       \marginpar{\smallskip\l_stex_key_min_tl}{~\problem@kw@minutes}
10169     }
10170   }
10171   \bool_if:NTF \g_@@_qr_in_subproblems_json_bool {
10172     \qrjson {,}
10173   }{
10174     \qrjson {
10175       , "subproblems": [
10176     }
10177     \bool_gset_true:N \g_@@_qr_in_subproblems_json_bool
10178   }
10179   \qrjson {
10180     \c_left_brace_str
10181     "id": "\_@@_qr_escape:o\l_stex_key_id_str", "title": "\_@@_qr_escape:o\l_stex_key_title_tl"
10182     "number": "\thesproblem.\int_use:N \g__problems_subproblem_int"
10183   }
10184   \bool_if:NF \l__problems_has_pts_bool {
10185     \qrjson {
10186       , "pts": \l_stex_key_pts_tl
10187     }
10188   }
10189 }{
10190 \qrjson {\c_right_brace_str}
10191 \end{list}
10192 }
10193
10194 \stexstylegnote{
10195   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
10196   \noindent\emph{\problem@kw@grading\str_if_empty:NF \l_stex_key_title_tl{
10197     {~}\l_stex_key_title_tl
10198   } :~}
10199   \qrjson {
10200     , "gnote": \c_left_brace_str
10201     "id": "\_@@_qr_escape:o\l_stex_key_id_str", "title": "\_@@_qr_escape:o\l_stex_key_title_tl"
10202     "anscls": [
10203   }
10204 }{
10205   \qrjson {

```

```

10206   ]\c_right_brace_str
10207 }
10208 \par\rule[.3em]{\linewidth}{0.4pt}\newline
10209 }
10210
10211 \renewcommand \anscls [2] [] {
10212   \stex_keys_set:nn{ anscls }{#1}
10213   \str_if_empty:NT \l_stex_key_id_str {
10214     \int_incr:N \l__problems_anscls_int
10215     \str_set:Nx \l_stex_key_id_str {
10216       AC\int_use:N \l__problems_anscls_int
10217     }
10218   }
10219   \bool_if:NTF \g_@@_qr_in_anscls_json_bool {
10220     \qrjson{,}
10221   }{
10222     \bool_set_true:N \g_@@_qr_in_anscls_json_bool
10223   }
10224   \qrjson{
10225     \c_left_brace_str
10226     "id": "\_@@_qr_escape:o\l_stex_key_id_str", "description": "\_@@_qr_escape:n{#2}",
10227     "feedback": "\_@@_qr_escape:o\l_stex_key_feedback_tl",
10228     "pts": "\l_stex_key_pts_str"
10229     \c_right_brace_str
10230   }
10231   \begin{list}{}{
10232     \setlength\topsep{0pt}
10233     \setlength\parsep{0pt}
10234     \setlength\rightmargin{0pt}
10235   }\item[\l_stex_key_id_str]
10236     \stex_if_do_html:TF{
10237       \exp_args:Ne \stex_annotate:nn{
10238         data-ftml-answerclass={\l_stex_key_id_str}
10239         \str_if_empty:NF \l_stex_key_pts_str{
10240           ,data-ftml-answerclass-pts={\l_stex_key_pts_str}
10241         }
10242       }{
10243         #2
10244         \tl_if_empty:NF \l_stex_key_feedback_tl{
10245           \stex_annotate_invisible:nn{
10246             data-ftml-answerclass-feedback={true}
10247           }{\l_stex_key_feedback_tl}
10248         }
10249       }
10250     }{#2}
10251     \str_if_empty:NF \l_stex_key_pts_str {\par
10252       ~ \problem@kw@points :~\l_stex_key_pts_str
10253     }
10254     \str_if_empty:NF \l_stex_key_feedback_tl {\par
10255       ~ \problem@kw@feedback :~\l_stex_key_feedback_tl
10256     }
10257   \end{list}
10258 }
10259 \stex_deactivate_macro:Nn \anscls {gnote-environments}

```

```

10260
10261 \AtEndDocument{
10262   \message{^^J^^J\g_@@_qr_json_str^^J^^J}
10263   \bool_if:NT \c__problems_gnotes_bool {
10264     \iow_new:N \c_@@_qr_json_iow
10265     \iow_open:Nn \c_@@_qr_json_iow {\jobname-vollkorn.json}
10266     \iow_now:Nx \c_@@_qr_json_iow {\g_@@_qr_json_str}
10267     \iow_close:N \c_@@_qr_json_iow
10268   }
10269 }
10270
10271 \renewcommand\testemptypage[1][\%
10272   \bool_if:NT \c__problems_test_bool {\
10273     \xdef\grid{P\thepage}
10274     \doprobblemqr
10275     \vfill\begin{center}\hwexam@kw@testemptypage\end{center}\eject
10276   }
10277 }
10278 }

```

39.3.3 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.)`

```

10279 \stex_keys_define:nnnn{ assignment }{
10280   \tl_clear:N \l_stex_key_number_tl
10281   \tl_clear:N \l_stex_key_given_tl
10282   \tl_clear:N \l_stex_key_due_tl
10283 }{
10284   number .tl_set:N      = \l_stex_key_number_tl,
10285   given .tl_set:N       = \l_stex_key_given_tl,
10286   due .tl_set:N         = \l_stex_key_due_tl,
10287   unknown .code:n = {}
10288 }{id,title,style}
10289
10290 \newcounter{assignment}
10291 \stex_new_stytable_env:nnnnnn {assignment}{0}{}{
10292   \cs_if_exist:NTF \l_hwexam_includeassignment_keys_tl {
10293     \tl_put_left:Nn \l_hwexam_includeassignment_keys_tl {\#1,}
10294     \exp_args:Nno \stex_keys_set:nn{assignment}{
10295       \l_hwexam_includeassignment_keys_tl
10296     }
10297   }{
10298     \stex_keys_set:nn{assignment}{\#1}
10299   }
10300   \tl_if_empty:NF \l_stex_key_number_tl {
10301     \global\setcounter{assignment}{\int_eval:n{\l_stex_key_number_tl-1}}
10302   }
10303   \global\refstepcounter{assignment}
10304   \setcounter{sproblem}{0}
10305   \def\thesproblem{\theassignment.\arabic{sproblem}}

```

```

10306 \stex_if_do_html:T{
10307   \tl_if_empty:NF \l_stex_key_title_tl {
10308     \stexdoctitle \l_stex_key_title_tl
10309   }
10310 }
10311 \stex_style_apply:
10312 \_stex_do_id:
10313 }{
10314 \stex_style_apply:
10315 }{
10316 \par\begin{center}
10317 \textbf{\Large\assignmentautorefname~\theassignment
10318 \tl_if_empty:NF \l_stex_key_title_tl {
10319   {~}---\l_stex_key_title_tl
10320 }
10321 }\par\smallskip
10322 \textbf{
10323   \tl_if_empty:NF \l_stex_key_given_tl {
10324     \hwexam@kw@given :~\l_stex_key_given_tl\quad
10325   }
10326   \tl_if_empty:NF \l_stex_key_due_tl {
10327     \hwexam@kw@due :~\l_stex_key_due_tl\quad
10328   }
10329 }
10330 \end{center}
10331 \par\bigskip
10332 }{
10333 \par\pagebreak
10334 }{}

```

`\includeassignment`

```

10335 \NewDocumentCommand\includeassignment{0{} m}{
10336   \group_begin:
10337   \tl_set:Nn \l_hwexam_includeassignment_keys_tl {#1}
10338   \stex_keys_set:nn{includeproblem}{#1}
10339   \exp_args:Nno \use:nn{\inputref[]\l_stex_key_mhrepos_str}{#2}
10340   \group_end:
10341 }

```

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

Restoring information about problems:

```

10342 \prop_new:N \c_@@_problems_prop
10343 \tl_set:Nn \c_@@_total_mins_tl {0pt}
10344 \tl_set:Nn \c_@@_total_pts_tl {0pt}
10345 \int_new:N \c_@@_total_problems_int
10346 \cs_set_protected:Npn \problem@restore #1 #2 #3 {
10347   \int_gincr:N \c_@@_total_problems_int
10348   \prop_gput:Nnn \c_@@_problems_prop {#1}{#2}{#3}
10349   \tl_gset:Ne \c_@@_total_pts_tl { \dim_eval:n { \c_@@_total_pts_tl + #2pt }}
10350   \tl_gset:Ne \c_@@_total_mins_tl { \dim_eval:n { \c_@@_total_mins_tl + #2pt }}
10351 }

```

`\correction@table` This macro generates the correction table

```

10352 \newcommand\correction@table{

```

```

10353 \int_compare:nNnT \c_@@_total_problems_int = 0 {
10354   \int_incr:N \c_@@_total_problems_int
10355   \prop_put:Nnn \c_@@_problems_prop {~}{~}{~}}
10356 }
10357 \tl_clear:N \l_tmpa_tl
10358 \tl_clear:N \l_tmpb_tl
10359 \tl_clear:N \l_tmpc_tl
10360 \prop_map_inline:Nn \c_@@_problems_prop {
10361   \tl_put_right:Nn \l_tmpa_tl { ##1 & }
10362   \tl_put_right:Nx \l_tmpb_tl { \use_i:nn ##2 & }
10363   \tl_put_right:Nn \l_tmpc_tl { & }
10364 }
10365 \resizebox{\textwidth}{!}{%
10366 \exp_args:Nne \begin{tabular}{|l|*{\int_use:N \c_@@_total_problems_int}{c|}c|l|}\hline
10367 &\exp_args:Ne \multicolumn{\int_eval:n{ \c_@@_total_problems_int + 1}}{c|}{
10368 {\footnotesize\hwexam@kw@forgrading} &\\ \hline
10369 \hwexam@kw@probs & \l_tmpa_tl \hwexam@kw@sum & \hwexam@kw@grade\\ \hline
10370 \hwexam@kw@pts & \l_tmpb_tl \dim_to_decimal:n{\c_@@_total_pts_tl} & \\ \hline
10371 \hwexam@kw@reached & \l_tmpc_tl & \\ [.7cm] \hline
10372 \end{tabular}}

```

(End of definition for \correction@table. This function is documented on page ??.)

\testheading

```

10373 \def\hwexamheader{\input{hwexam-default.header}}
10374
10375 \def\hwexamminutes{
10376   \tl_if_empty:NTF \hwexam@duration {
10377     {\hwexam@min}~\hwexam@minutes@kw
10378   }{
10379     \hwexam@duration
10380   }
10381 }
10382
10383 \stex_keys_define:nnnn{ hwexam / testheading }{
10384   \tl_clear:N \hwexam@min
10385   \tl_clear:N \hwexam@duration
10386   \tl_clear:N \hwexam@reqpts
10387   \tl_clear:N \hwexam@tools
10388 }{
10389   min .tl_set:N = \hwexam@min,
10390   duration .tl_set:N = \hwexam@duration,
10391   reqpts .tl_set:N = \hwexam@reqpts,
10392   tools .tl_set:N = \hwexam@tools
10393 }{}
10394
10395 \newenvironment{testheading}[1][]{
10396   \stex_keys_set:nn { hwexam / testheading}{#1}
10397
10398   \tl_set:Nx \hwexam@totalpts {\dim_to_decimal:n \c_@@_total_pts_tl}
10399   \tl_set:Nx \hwexam@totalmin {\dim_to_decimal:n \c_@@_total_mins_tl}
10400   \tl_set:Nx \hwexam@checktime {\dim_to_decimal:n { \hwexam@min pt - \hwexam@totalmin pt }}
10401
10402   \newif\if@bonuspoints

```

```

10403 \tl_if_empty:NTF \hwexam@reqpts {
10404   \@bonuspointsfalse
10405 }{
10406   \tl_set:Nx \hwexam@bonuspts {
10407     \dim_to_decimal:n{\hwexam@totalpts pt - \hwexam@reqpts pt}
10408   }
10409   \@bonuspointstrue
10410 }
10411
10412 \makeatletter\hwexamheader\makeatother
10413 }{
10414   \newpage
10415 }

```

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

```

examdata
quizdata
homeworkdata
10416 \bool_new:N \g_@@_allow_data_bool
10417 \bool_gset_true:N \g_@@_allow_data_bool
10418 \AtBeginDocument{
10419   \bool_gset_false:N \g_@@_allow_data_bool
10420 }
10421
10422 \msg_set:nnn{stex}{hwexam/doubledata}{
10423   Exam/Homework/Quiz~Data~already~set
10424 }
10425
10426 \msg_set:nnn{stex}{hwexam/nodate}{
10427   Exam/Homework/Quiz~Data~missing~date~value
10428 }
10429
10430 \msg_set:nnn{stex}{hwexam/nocourse}{
10431   Exam/Homework/Quiz~Data~missing~course~value
10432 }
10433
10434 \stex_keys_define:nnnn{ hwexam / commondata }{
10435   % https://docs.rs/dateparser/latest/dateparser/#accepted-date-formats
10436   \str_clear:N \l_@@_key_exam_date_str
10437   \str_clear:N \l_@@_key_exam_course_id_str
10438   \str_clear:N \l_@@_key_exam_term_str
10439   \int_set:Nn \l_@@_key_exam_num_int {1}
10440 }{
10441   date      .str_set:N = \l_@@_key_exam_date_str,
10442   course    .str_set:N = \l_@@_key_exam_course_id_str,
10443   term      .str_set:N = \l_@@_key_exam_term_str,
10444   num       .int_set:N = \l_@@_key_exam_num_int
10445 }{}
10446
10447 \stex_keys_define:nnnn{ hwexam / examdata }{
10448   \bool_set_false:N \l_@@_key_exam_retake_bool
10449 }{
10450   retake    .bool_set:N = \l_@@_key_exam_retake_bool
10451 }{hwexam/commondata}
10452

```

```

10453 \cs_set_protected:Nn \_@@_do_metadata:nnnn {
10454   \bool_if:NF \g_@@_allow_data_bool {
10455     \msg_fatal:nn{stex}{hwexam/doubledata}
10456   }
10457   \bool_gset_false:N \g_@@_allow_data_bool
10458   \stex_keys_set:nn { hwexam / #1}{#2}
10459   \str_set:Nn \g_stex_document_kind_str{#4}
10460   \str_if_empty:NT\l_@@_key_exam_date_str{
10461     \str_set:Ne \l_@@_key_exam_date_str \today
10462     \%msg_fatal:nn{stex}{hwexam/nodate}
10463   }
10464   \str_if_empty:NT\l_@@_key_exam_course_id_str{
10465     \msg_fatal:nn{stex}{hwexam/nocourse}
10466   }
10467   \cs_if_exist:NT \date {
10468     \exp_args:No \date \l_@@_key_exam_date_str
10469   }
10470
10471   \tl_set:Ne \g_stex_document_kind_parameters_tl{
10472     ,data-ftml-document-kind-date={\l_@@_key_exam_date_str}
10473     ,data-ftml-document-kind-num=\int_use:N \l_@@_key_exam_num_int
10474     ,data-ftml-document-kind-course=\l_@@_key_exam_course_id_str
10475     \str_if_empty:NF \l_@@_key_exam_term_str{
10476       ,data-ftml-document-kind-term=\l_@@_key_exam_term_str
10477     }
10478     #3
10479   }
10480 }
10481
10482 \newcommand\examdata[1]{
10483   \_@@_do_metadata:nnnn{examdata}{#1}{
10484     ,data-ftml-document-kind-retake=\bool_if:NTF \l_@@_key_exam_retake_bool{true}{false}
10485   }{exam}
10486 }
10487
10488 \newcommand\homeworkdata[1]{
10489   \_@@_do_metadata:nnnn{commondata}{#1}{}{homework}
10490 }
10491
10492 \newcommand\quizdata[1]{
10493   \_@@_do_metadata:nnnn{commondata}{#1}{}{quiz}
10494 }
10495

```

(End of definition for examdata, quizdata, and homeworkdata. These functions are documented on page ??.)

```

10496 \end{package}

```

39.3.4 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\biertglas{{\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{\biertglas}

```

39.4 Tikzinput Implementation

```

10497 <@=tikzinput>
10498 <*package>
10499
10500 %%%%%%%%%%%%%% tikzinput.dtx %%%%%%%%%%%%%%
10501
10502 \ProvidesExplPackage{tikzinput}{2025/11/11}{4.0.0}{tikzinput package}
10503 \RequirePackage{l3keys2e}
10504
10505 \keys_define:nn { tikzinput } {
10506   image .bool_set:N = \c_tikzinput_image_bool,
10507   image .default:n = false ,
10508   unknown .code:n = {}
10509 }
10510
10511 \ProcessKeysOptions { tikzinput }
10512
10513 \bool_if:NTF \c_tikzinput_image_bool {
10514   \RequirePackage{graphicx}
10515
10516   \providecommand\usetikzlibrary[]{}
10517   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
10518 }{
10519   \RequirePackage{tikz}
10520   \RequirePackage{standalone}
10521
10522   \newcommand \tikzinput [2] [] {
10523     \setkeys{Gin}{#1}
10524     \ifx \Gin@ewidth \Gin@exclamation
10525       \ifx \Gin@eheight \Gin@exclamation
10526         \input { #2 }
10527       \else
10528         \resizebox{!}{ \Gin@eheight }{
10529           \input { #2 }
10530         }
10531       \fi
10532     \else

```

```

10533     \ifx \Gin@eheight \Gin@exclamation
10534         \resizebox{ \Gin@ewidth }{!}{
10535             \input { #2 }
10536         }
10537     \else
10538         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
10539             \input { #2 }
10540         }
10541     \fi
10542 \fi
10543 }
10544 }
10545
10546 \newcommand \ctikzinput [2] [] {
10547     \begin{center}
10548         \tikzinput [#1] {#2}
10549     \end{center}
10550 }
10551 \end{package}

```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
\!	6695
\\$	2170, 2173, 2177
* commands:	
*:n	145
\,	6695
\:	617
\;	8150, 8157, 8202, 8204, 8234
@@ commands:	
\g_@@_allow_data_bool	10416, 10417, 10419, 10454, 10457
_@@_do_metadata:nnnn	10453, 10483, 10489, 10493
\l_@@_key_exam_course_id_str	10437, 10442, 10464, 10474
\l_@@_key_exam_date_str	10436, 10441, 10460, 10461, 10468, 10472
\l_@@_key_exam_num_int	10439, 10444, 10473
\l_@@_key_exam_retake_bool	10448, 10450, 10484
\l_@@_key_exam_term_str	10438, 10443, 10475, 10476
\c_@@_problems_prop	10342, 10348, 10355, 10360
_@@_qr_escape:n	10003, 10004, 10051, 10093, 10140, 10181, 10201, 10226, 10227
_@@_qr_escape_char:n	9999, 10003
\g_@@_qr_in_anscls_json_bool	9990, 9991, 10219, 10222
\g_@@_qr_in_problems_json_bool	9986, 9987, 10043, 10046, 10085, 10088
\g_@@_qr_in_subproblems_json_-bool	9988, 9989, 10063, 10066, 10105, 10108, 10130, 10136, 10171, 10177
\c_@@_qr_json_iow	10264, 10265, 10266, 10267
\g_@@_qr_json_str	9985, 9993, 10262, 10266
\c_@@_total_mins_tl	10343, 10350, 10399
\c_@@_total_problems_int	10345, 10347, 10353, 10354, 10366, 10367
c@@_total_pts_tl	10344, 10349, 10370, 10398
\\	88, 115, 842, 1464, 8385, 8390, 8808, 8811, 9601, 9662, 9796, 9979, 10368, 10369, 10370, 10371
\{	8207
\}	8207
\sqcup	19, 79, 619, 756, 8161, 8239, 8417, 9932, 10272
_	618
_@@_qr_backslash	9979, 10000
_comp	3890, 4648, 5935, 6150, 6302, 7047
_customthiscomp	5008, 5009, 5021, 5026, 5029, 5034
_defcomp	5968, 6134
_thiscomp	4777, 4778, 5939
_varcomp	4335, 4494, 5265, 5286, 5836, 5957, 6065, 6079, 6093
\l	2169, 2172, 2176

A

\activateexcursion	64, 112
\addbibresource	82, 141, 2186
\addcontentsline	8415
\addmhbibresource	82, 141, 2179
\addtocounter	8610
\AddToHook	1793, 7804, 7944, 7948, 7952, 8079, 9002, 9957
\aftergroup	125, 126, 2935, 2957, 3006, 4611, 7870, 7871, 8075
\afterprematurestop	63, 111, 8479, 8490
\amalg	8224
\anscls	9443, 9476, 9518, 10211, 10259
\apply	90
\arabic	9043, 9046, 10305
\arg	40, 95, 5386
\argarraymap	94, 154, 6382, 6937
\argmap	94, 154, 296, 6381, 6698, 6913
\argsep	36, 94, 154, 5769, 6380, 6693, 6897, 8157, 8158, 8159, 8167, 8214, 8218, 8222
\assign	58, 146, 3315, 3574
assignment (env.)	76, 119, 10279
\assignmentautorefname	10317
\assignMorphism	146, 3316, 3679
\assumption	158, 7700, 8083

<code>\ast</code>	8182	9201, 9204, 9216, 9217, 9221, 9222,	
<code>\AtBeginDocument</code>	69, 77,	9337, 9348, 9385, 9396, 9437, 9448,	
	806, 814, 1483, 1510, 1691, 1716,	9520, 9525, 9615, 9619, 9642, 9656,	
	1797, 2953, 8261, 8434, 8949, 10418	9753, 9757, 9776, 9790, 9932, 9934,	
<code>\AtEndDocument</code> ...	627, 1798, 9998, 10261	9938, 9982, 9992, 10008, 10032,	
<code>\AtEndOfPackageFile</code>	2222, 2238, 2253	10037, 10043, 10063, 10074, 10079,	
<code>\author</code>	8788	10085, 10105, 10120, 10121, 10125,	
<code>\autoref</code>	86, 1863, 1879	10126, 10130, 10143, 10161, 10162,	
		10166, 10167, 10171, 10184, 10219,	
		10263, 10272, 10454, 10484, 10513	
B			
<code>\backmatter</code>	210, 1758, 1759, 1761	<code>\bool_if_exist:N</code> TF	339, 353
<code>\baselineskip</code>	8746	<code>\bool_lazy_any:n</code> TF	4224, 4293
<code>\beameritemnestingprefix</code>	8886	<code>\bool_lazy_any_p:n</code>	285
<code>\begin</code>	98, 128, 139, 142–144,	<code>\bool_new:N</code>	27, 685, 1312,
	157, 1470, 1582, 1645, 2081, 2236,		1606, 2387, 2673, 3033, 4312, 4639,
	2251, 2271, 2615, 2773, 2791, 2806,		5562, 5976, 6970, 7192, 9022, 9061,
	3070, 6960, 7297, 7961, 8418, 8430,		9062, 9063, 9986, 9988, 9990, 10416
	8556, 8583, 8593, 8632, 8653, 8661,	<code>\bool_not_p:n</code>	284, 288
	8707, 8708, 8709, 8710, 8711, 8712,	<code>\bool_set_false:N</code>	455,
	8718, 8720, 8758, 8760, 8764, 8826,		582, 693, 701, 975, 998, 1314, 1621,
	8838, 8873, 8874, 9084, 9184, 9211,		2401, 3040, 3340, 3696, 4642, 5992,
	9264, 9327, 9375, 9426, 9485, 9504,		6290, 6538, 6829, 6977, 7205, 7547,
	9532, 9542, 9697, 9813, 9818, 9823,		7552, 7562, 7567, 7591, 7677, 7812,
	9828, 9932, 9940, 10114, 10154,		7890, 8073, 8138, 8139, 8305, 8346,
	10231, 10275, 10316, 10366, 10547		8510, 9028, 9064, 9104, 9107, 9584,
<code>\begingroup</code>	46, 2089, 2594		9589, 9595, 9987, 9989, 9991, 10448
<code>\bf</code>	8745	<code>\bool_set_true:N</code>	341, 355,
<code>\bfseries</code>	9168, 10018, 10020		463, 571, 577, 690, 705, 972, 1352,
<code>\bgroup</code> ..	1900, 1992, 4378, 4551, 7713,		1611, 2399, 2716, 3036, 3324, 3708,
	7837, 7926, 8582, 8678, 8684, 8700,		3718, 4041, 4398, 4640, 4709, 4979,
	8808, 8811, 9281, 9341, 9389, 9440		5011, 5036, 5078, 5079, 5371, 5380,
<code>\bigcirc</code>	9747, 9773		5495, 5596, 5602, 5620, 5638, 5671,
<code>\bigskip</code>	9137, 10068, 10110, 10331		5794, 5884, 5917, 5942, 5943, 5981,
<code>blindfragment (env.)</code>	84, 139, 1628		6217, 6545, 6994, 7195, 7203, 7599,
bool commands:			
	<code>\bool_gset_eq:NN</code>	3126, 7846	
	<code>\bool_gset_false:N</code>		
		10066, 10108, 10419, 10457	
	<code>\bool_gset_true:N</code>		
		10046, 10088, 10136, 10177, 10417	
	<code>\bool_if:N</code> TF ...	473, 581, 600, 601,	
		607, 696, 724, 978, 987, 989, 1083,	
		1626, 1743, 2396, 2448, 2676, 2991,	
		3187, 3327, 3359, 3723, 4393, 4618,	
		5260, 5284, 5565, 5575, 5581, 5591,	
		5595, 5615, 5629, 5633, 5647, 5676,	
		5683, 5697, 6139, 6162, 6220, 6550,	
		6976, 6982, 7584, 7713, 7723, 7739,	
		7772, 7778, 7806, 7862, 7902, 7926,	
		7929, 7936, 7940, 7954, 8001, 8089,	
		8321, 8329, 8366, 8450, 8494, 8675,	
		8681, 8688, 8722, 8737, 8787, 8837,	
		8847, 8934, 8940, 8967, 8990, 9066,	
		9087, 9123, 9128, 9138, 9166, 9187,	
		9201, 9204, 9216, 9217, 9221, 9222,	
		9337, 9348, 9385, 9396, 9437, 9448,	
		9520, 9525, 9615, 9619, 9642, 9656,	
		9753, 9757, 9776, 9790, 9932, 9934,	
		9938, 9982, 9992, 10008, 10032,	
		10037, 10043, 10063, 10074, 10079,	
		10085, 10105, 10120, 10121, 10125,	
		10126, 10130, 10143, 10161, 10162,	
		10166, 10167, 10171, 10184, 10219,	
		10263, 10272, 10454, 10484, 10513	
		<code>\bool_if_exist:N</code> TF	339, 353
		<code>\bool_lazy_any:n</code> TF	4224, 4293
		<code>\bool_lazy_any_p:n</code>	285
		<code>\bool_new:N</code>	27, 685, 1312,
			1606, 2387, 2673, 3033, 4312, 4639,
			5562, 5976, 6970, 7192, 9022, 9061,
			9062, 9063, 9986, 9988, 9990, 10416
		<code>\bool_not_p:n</code>	284, 288
		<code>\bool_set_false:N</code>	455,
			582, 693, 701, 975, 998, 1314, 1621,
			2401, 3040, 3340, 3696, 4642, 5992,
			6290, 6538, 6829, 6977, 7205, 7547,
			7552, 7562, 7567, 7591, 7677, 7812,
			7890, 8073, 8138, 8139, 8305, 8346,
			8510, 9028, 9064, 9104, 9107, 9584,
			9589, 9595, 9987, 9989, 9991, 10448
		<code>\bool_set_true:N</code>	341, 355,
			463, 571, 577, 690, 705, 972, 1352,
			1611, 2399, 2716, 3036, 3324, 3708,
			3718, 4041, 4398, 4640, 4709, 4979,
			5011, 5036, 5078, 5079, 5371, 5380,
			5495, 5596, 5602, 5620, 5638, 5671,
			5794, 5884, 5917, 5942, 5943, 5981,
			6217, 6545, 6994, 7195, 7203, 7599,
			7609, 7790, 8318, 8358, 8508, 8516,
			8517, 8518, 8519, 8520, 8521, 9100,
			9105, 9108, 9171, 9582, 9594, 10222
		<code>\bool_while_do:Nn</code>	973
		<code>\bool_while_do:nn</code>	
			283, 2202, 7614, 7626, 7638, 7655, 7667
		<code>\l_tmpa_bool</code>	3696,
			3708, 3718, 3723, 6538, 6545, 6550
box commands:			
	<code>\box_wd:N</code>	9895	
	<code>\c_empty_box</code>		
		1530, 1540, 3916, 9522, 9527	
	<code>\l_tmpa_box</code>		
		2715, 6372, 8678, 8684, 8700,	
		9281, 9341, 9389, 9440, 9892, 9895	
<code>boxed</code>		113	
C			
<code>\catcode</code>	19, 20, 46,	79, 88, 617, 618, 619, 756, 757, 842,	

1993, 2168, 2169, 2170, 2171, 2172,	8165, 8167, 8169, 8170, 8175, 8177,
2173, 2175, 2176, 2177, 2507, 2508	8182, 8186, 8187, 8190, 8191, 8196,
\cdot 8202	8197, 8203, 8204, 8205, 8207, 8214,
\centering 8419, 8633	8218, 8234, 8239, 8243, 8245, 8247
\chapter 26, 84, 8459, 8460	\comp-macro 151
\chaptername 8390, 8456, 8457	\compemph 105, 151, <u>5927</u> , 8951
\chaptertitlename 8390	\conclude 158, 7699, <u>8084</u>
\checkmark 9644, 9645, 9778, 9779	\conclusion .. 44, 101, 103, 157, 7436, <u>7510</u>
\child 125	\copymod 55–57, 145, 3502, 3504, 3506
\circ 47	copymodule (env.) 145, <u>3442</u>
\clearpage 1752, 1765, 1776, 1787	\copymodule 3477, 3479
clist commands:	\counterwithin 8463, 8475
\clist_clear:N ... 125, 399, 3764,	\cr 9911, 9912, 9914
4710, 4860, 4960, 6271, 7267, 7272	\crrc 9870
\clist_count:N 4839, 4850	cs commands:
\clist_count:n 4943, 6945	\cs:w 483, 486, 517, 520,
\clist_get:NN 508, 546	620, 2452, 2453, 2454, 2460, 2464,
\clist_if_empty:NTF 454, 504, 542, 4057, 4838,	2470, 2729, 7146, 7148, 7326, 7340
4885, 5205, 6469, 7352, 9544, 9708	\cs_argument_spec:N 4176
\clist_if_in:NnTF 104, 107, 131, 7447, 9007, 9010,	\cs_end: 483, 486, 517, 520,
9013, 9016, 9549, 9552, 9677, 9684,	620, 2452, 2453, 2454, 2462, 2466,
9714, 9717, 9962, 9965, 9968, 9971	2470, 2729, 7146, 7148, 7326, 7340
\clist_if_in:nntf 4969	\cs_generate_from_arg_count:NNnn
\clist_item:Nn 5156, 6478 4050,
\clist_item:nn 6956	4407, 4566, 4743, 5504, 5540, 6825
\clist_map_function:NN ... 4711, 4888	\cs_generate_variant:Nn 56,
\clist_map_function:nN 3492, 3558, 4142, 6645	146, 176, 248, 331, 732, 1330, 2546,
\clist_map_inline:Nn 126, 135, 457, 3227, 4059, 4416,	2552, 3030, 4616, 4638, 5499, 6528
4575, 5206, 7070, 7105, 7250, 7285	\cs_if_eq:NNTF .. 65, 73, 802, 810,
\clist_map_inline:nn 363, 412, 5720, 5823, 5852, 7068, 7502	1200, 1205, 1688, 2759, 2773, 2776,
\clist_pop:NN 6480	2791, 2794, 4178, 5753, 5755, 6908
\clist_put_right:Nn 127, 4724, 4728, 4867, 4987, 4989	\cs_if_exist:NTF 675, 1563, 1567, 1610, 1614, 1630,
\clist_set:Nn 42, 123, 4480, 9006, 9961	1633, 1662, 1666, 1700, 1706, 1717,
\clist_set_eq:NN 121, 7302	1745, 1758, 1773, 1784, 1832, 1833,
\l_tmpa_clist 9006, 9007, 9010, 9013,	1862, 1920, 1971, 2047, 2048, 2050,
9016, 9961, 9962, 9965, 9968, 9971	2145, 3918, 4173, 4680, 5235, 5238,
\clstinputmhlisting 85, <u>141</u> , 2238	5246, 5249, 6557, 6562, 6594, 6600,
\cmhgraphics 85, <u>141</u> , <u>2222</u>	6624, 6626, 6666, 6672, 8385, 8390,
\cmhtikzinput 121, <u>141</u> , <u>2253</u>	8393, 8397, 8401, 8405, 8409, 8456,
\color 10020	8459, 8462, 8468, 8471, 8474, 8635,
\columnbox 8651, 8653, 8671	8643, 8655, 8665, 8674, 8808, 8811,
\comp 31–34, 40, 92, 95, 105, 151, 3890, 4335,	9069, 9644, 9778, 9908, 10292, 10467
4494, 4659, 4660, 4778, 4826, 4834,	\cs_meaning:N . 2451, 4084, 6531, 6680
5017, 5018, 5028, 5702, 5836, <u>5927</u> ,	\cs_new:Nn 26, 34, 205, 306, 649, 763, 771,
5979, 5980, 6065, 6079, 6093, 6134,	888, 895, 1098, 1105, 1117, 1120,
6150, 6152, 6302, 6679, 6773, 6779,	1123, 1126, 1137, 1140, 1143, 1146,
6781, 6995, 7000, 7047, 8154, 8156,	1149, 1155, 1161, 1238, 1241, 1244,
	1251, 1254, 1257, 1260, 1264, 1285,
	1288, 1299, 1307, 1407, 1410, 1413,
	1421, 1424, 1427, 1430, 1433, 1436,
	1440, 1443, 1446, 1976, 1988, 2274,
	2277, 2280, 2283, 2286, 2289, 2292,

2295, 2298, 2478, 2483, 2488, 3155,
 3159, 3954, 4117, 4753, 4896, 4903,
 4906, 4916, 5415, 5512, 5699, 6017,
 6042, 6483, 6494, 6508, 6511, 6567,
 6580, 6730, 6733, 6737, 6784, 7174,
 7343, 7357, 7498, 9048, 9850, 9868
 \cs_new:Npn 726,
 727, 735, 736, 881, 884, 891, 1006,
 1152, 1158, 1291, 1984, 2013, 3303,
 5748, 6021, 6181, 6573, 6586, 7230
 \cs_new_nopar:Nn 359, 374
 \cs_new_protected:Nn
 3, 4, 42, 45, 53, 67,
 73, 77, 87, 88, 97, 98, 103, 113, 141,
 153, 169, 177, 192, 208, 215, 245,
 249, 313, 324, 333, 346, 482, 495,
 503, 516, 541, 556, 599, 611, 625,
 630, 654, 699, 729, 740, 741, 779,
 825, 834, 898, 920, 928, 944, 957,
 968, 985, 995, 1010, 1164, 1170,
 1188, 1199, 1218, 1271, 1275, 1313,
 1332, 1359, 1372, 1384, 1392, 1495,
 1504, 1579, 1587, 1591, 1604, 1628,
 1643, 1698, 1744, 1811, 1830, 1850,
 1861, 1872, 1899, 1909, 1918, 1948,
 1990, 2002, 2038, 2064, 2078, 2088,
 2155, 2164, 2190, 2326, 2363, 2372,
 2389, 2395, 2404, 2419, 2431, 2447,
 2457, 2495, 2514, 2518, 2526, 2540,
 2543, 2547, 2565, 2569, 2603, 2613,
 2626, 2636, 2641, 2646, 2660, 2666,
 2681, 2713, 2724, 2733, 2744, 2751,
 2758, 2764, 2786, 2803, 2811, 2820,
 2829, 2838, 2861, 2876, 2891, 2918,
 2937, 2963, 2969, 2989, 3010, 3025,
 3044, 3082, 3091, 3101, 3112, 3120,
 3133, 3163, 3184, 3209, 3217, 3225,
 3235, 3260, 3275, 3280, 3285, 3291,
 3295, 3322, 3345, 3362, 3382, 3386,
 3396, 3416, 3418, 3423, 3466, 3533,
 3586, 3602, 3630, 3642, 3661, 3730,
 3793, 3824, 3913, 3921, 3934, 3948,
 3958, 3969, 4020, 4067, 4083, 4090,
 4096, 4120, 4137, 4146, 4158, 4165,
 4195, 4210, 4223, 4234, 4236, 4248,
 4289, 4313, 4350, 4363, 4376, 4432,
 4440, 4446, 4456, 4508, 4522, 4535,
 4598, 4617, 4646, 4655, 4673, 4677,
 4682, 4701, 4722, 4741, 4755, 4771,
 4776, 4782, 4836, 4859, 4866, 4870,
 4934, 4959, 4985, 5007, 5025, 5033,
 5043, 5060, 5071, 5091, 5101, 5144,
 5152, 5176, 5190, 5204, 5212, 5221,
 5231, 5259, 5292, 5311, 5322, 5369,
 5377, 5431, 5492, 5500, 5516, 5522,
 5559, 5564, 5575, 5580, 5591, 5594,
 5611, 5614, 5629, 5632, 5647, 5651,
 5661, 5664, 5675, 5681, 5697, 5709,
 5713, 5773, 5784, 5816, 5848, 5901,
 5978, 6137, 6158, 6231, 6239, 6242,
 6262, 6323, 6355, 6404, 6416, 6443,
 6463, 6468, 6500, 6515, 6530, 6537,
 6555, 6592, 6606, 6640, 6649, 6660,
 6677, 6686, 6752, 6762, 6767, 6788,
 6807, 6820, 6855, 6860, 6866, 6907,
 6975, 7042, 7062, 7151, 7166, 7178,
 7184, 7211, 7248, 7291, 7325, 7361,
 7375, 7394, 7404, 7410, 7462, 7477,
 7490, 7559, 7573, 7583, 7623, 7679,
 7717, 7730, 7753, 7771, 7777, 7821,
 7830, 7836, 7842, 7859, 7864, 7869,
 7985, 8018, 8045, 8257, 8379, 8455,
 8467, 8506, 8544, 8581, 8586, 8592,
 8602, 8687, 8859, 9155, 9255, 9318,
 9366, 9416, 9580, 9676, 9887, 9904
 \cs_new_protected:Npn
 16, 148, 303, 327,
 422, 587, 709, 713, 716, 720, 721,
 753, 1267, 1400, 1404, 1450, 1456,
 1491, 1561, 1660, 1676, 2015, 2046,
 2359, 2599, 2735, 2843, 2909, 3035,
 3039, 3195, 3205, 3267, 3307, 3408,
 3650, 4171, 4191, 4585, 4635, 4688,
 4708, 4763, 4768, 4803, 4825, 4830,
 4923, 4942, 4955, 5315, 5347, 5409,
 5727, 5905, 5929, 5933, 5935, 5939,
 5951, 5955, 5957, 5960, 5964, 5968,
 5971, 5975, 6622, 6665, 6671, 6792,
 6799, 6847, 6897, 6913, 6938, 6992,
 7004, 7014, 7194, 7216, 7231, 7283,
 7612, 7636, 7653, 7665, 7783, 7849,
 7854, 8130, 8254, 9161, 9309, 9312
 \cs_set:Nn 9531, 9550, 9553,
 9572, 9576, 9601, 9679, 9680, 9686,
 9687, 9696, 9715, 9718, 9737, 9741
 \cs_set:Npe 4100
 \cs_set:Npn 487,
 524, 530, 614, 615, 900, 913, 919,
 2691, 3325, 3339, 3452, 3457, 3518,
 3523, 4051, 4122, 4124, 4127, 4140,
 4197, 4408, 4458, 4567, 4743, 4962,
 4968, 5013, 5037, 5505, 5541, 5850,
 5907, 6375, 6376, 6380, 6381, 6382,
 6643, 6679, 6699, 6810, 6825, 6919,
 6944, 8135, 8136, 8137, 8361, 8692
 \cs_set_eq:NN 54, 74, 315, 319, 1512,
 1518, 1649, 1901, 1994, 3318, 3319,
 3320, 4174, 4649, 4651, 4704, 5266,

5268, 5386, 5389, 5417, 5418, 5578,
 5707, 5818, 5821, 6601, 6613, 6616,
 6667, 6673, 6823, 6876, 6889, 8735
`\cs_set_protected:Nn`
 1040, 8380, 8413, 8429, 10453
`\cs_set_protected:Npn`
 69, 1528, 1538, 3914, 3915,
 5034, 5927, 6693, 6698, 6704, 8457,
 8460, 8469, 8472, 8736, 8759, 8763,
 8774, 8778, 8888, 8891, 8894, 10346
`\cs_set_protected:Npx` 5009
`\cs_undefine:N` 351, 1071, 1834
`\l_tmpa_cs` 5850,
 5868, 5877, 5907, 5912, 6375, 6385
`\csname` ... 349, 663, 2166, 2167, 2168,
 2169, 2170, 2175, 2176, 2177, 2580,
 3483, 3485, 3549, 3551, 7585, 7871
`\ctikzinput` 121, 10546
`\currentgrouplevel` 325,
 328, 334, 335, 337, 339, 341, 347,
 349, 351, 353, 355, 7860, 7866, 7871
`\CurrentOption` 10, 8308, 8309,
 8310, 8311, 8350, 8351, 8928, 9952
`\Currentsectionlevel` 84, 139, 1528
`\currentsectionlevel` 84, 139, 1528

D

`\date` 10467, 10468
`\DeclareOption` 10
`\def` 93, 669,
 673, 676, 678, 1493, 2503, 2529,
 3833, 3890, 4335, 4342, 4494, 4501,
 4659, 4660, 5016, 5021, 5029, 5038,
 5836, 5950, 5980, 6040, 6057, 6065,
 6079, 6093, 6119, 6129, 6134, 6150,
 6302, 6327, 6347, 7047, 8478, 8480,
 8545, 8546, 8547, 8551, 8554, 8611,
 8727, 8744, 8752, 8788, 8789, 8791,
 8793, 8795, 8797, 8798, 8800, 8802,
 8804, 8807, 8808, 8810, 8811, 8814,
 8816, 8818, 8834, 8886, 8950, 8951,
 8952, 9043, 9046, 9047, 9519, 9524,
 9999, 10006, 10009, 10015, 10022,
 10026, 10027, 10305, 10373, 10375
`\defemph` 105, 151, 5960
`\Definame` 102, 152, 6097, 7419, 7441
`\definame` 23, 33, 39, 101,
 102, 105, 152, 305, 6097, 7418, 7440
`\Definames` 6129
`\definames` 6119
`\definiendum` 23, 33,
 39, 101, 102, 105, 152, 6097, 7416, 7438
`\definiens` 41,
 50, 58, 101–103, 7421, 7455, 7475, 7476

`definiens` 157, 7455
`\defnotation`
 ... 33, 39, 102, 152, 6097, 7417, 7439
`\detokenize` 9007, 9010,
 9013, 9016, 9962, 9965, 9968, 9971
 dim commands:
`\dim_compare:nNnTF` 9921
`\dim_eval:n` 10349, 10350
`\dim_new:N` 9246
`\dim_to_decimal:n` 9895,
 10370, 10398, 10399, 10400, 10407
`\dim_zero:N` 9249, 9834
`\dimexpr` 8723
 do commands:
`_do_comp:nNn` 151
`_do_comp:nnNn` 151, 5936,
 5958, 5969, 5976, 5978, 6174, 6705
`\dobracket` 94
`\dobrackets` ... 155, 6983, 6992, 7015,
 8148, 8183, 8190, 8191, 8196, 8197
`\document-URI` 125, 144
`\doproblemqr`
 .. 10009, 10026, 10073, 10160, 10274
`\dowithbrackets` 155, 7014
`\due` 76, 119
`\duration` 77, 119

E

`\edef` 2168, 2169, 2170, 4778, 8562
`\egroup` .. 1904, 1997, 4423, 4582, 7806,
 7839, 7954, 8589, 8678, 8684, 8702,
 8808, 8811, 9294, 9354, 9402, 9454
`\eject` 9932, 10275
`\ellipses`
 98, 4723, 4725, 5824, 5862, 6923, 6924
`\else` . 662, 674, 1469, 2082, 2104, 2119,
 2747, 2754, 5965, 8886, 8967, 8990,
 9167, 9278, 9293, 9648, 9782, 9920,
 10010, 10016, 10527, 10532, 10537
`\emph` 17, 18, 104, 1471,
 5975, 7799, 7816, 7978, 9299, 9359,
 9407, 9459, 9662, 9796, 9940, 10196
`\end` 128, 144,
 1474, 1588, 1657, 2081, 2236, 2251,
 2271, 2630, 2776, 2794, 2814, 3128,
 6962, 7317, 7976, 8423, 8430, 8482,
 8491, 8564, 8589, 8604, 8649, 8663,
 8671, 8692, 8694, 8707, 8708, 8709,
 8710, 8711, 8712, 8733, 8761, 8765,
 8768, 8832, 8840, 8881, 8882, 9119,
 9209, 9228, 9286, 9291, 9346, 9351,
 9394, 9399, 9446, 9451, 9502, 9515,
 9536, 9567, 9701, 9815, 9820, 9825,

9830, 9932, 9940, 10150, 10191, 10257, 10275, 10330, 10372, 10549	<code>\envname</code> 144
<code>\endbackmatter</code> 1760, 1762	<code>\eq</code> 31
<code>\endcsname</code> 349,	<code>\eqstep</code> 158, 7701, 8085
662, 663, 2166, 2167, 2168, 2169, 2170, 2175, 2176, 2177, 2580, 3483, 3485, 3549, 3551, 7585, 7871, 8886	<code>\equal</code> 30
<code>\endfrontmatter</code> 1747, 1748	<code>\errmessage</code> 6551, 8976, 8999
<code>\endgroup</code> 48, 50, 2096, 2596	<code>\escapechar</code> 88, 842, 9978
<code>\endinput</code> 1913, 2005	<code>\everyeof</code> 2725
<code>\endlist</code> 7832	<code>\examdata</code> 10482
<code>\endspfststepenv</code> 8008, 8012, 8095, 8099	<code>examdata</code> 10416
endstex commands:	<code>\examnumber</code> 10006, 10017, 10018, 10020
<code>\endstex_annotate_env</code> 2357, 9732	<code>\excursion</code> 64, 112, 8820
<code>\ensuremath</code> 5597, 5603, 5918	<code>\excursiongroup</code> 64, 112, 8854
environments:	<code>\excursionref</code> 64, 112, 8825, 8834, 8836, 8849
<code>assignment</code> 76, 119, 10279	<code>exnote (env.)</code> 1, 9366
<code>blindfragment</code> 84, 139, 1628	<code>\exnote</code> 9054, 9413, 9561, 9726
<code>copymodule</code> 145, 3442	exp commands:
<code>exnote</code> 1, 9366	<code>\exp_after:wN</code>
<code>extstructure</code> 98, 156, 7066 158, 483, 486, 517, 520, 882,
<code>extstructure*</code> 98	885, 892, 1006, 1030, 1121, 1138,
<code>frame</code> 1, 1, 8506	1141, 1144, 1147, 1150, 1153, 1159,
<code>gnote</code> 1, 9414	1239, 1252, 1255, 1261, 1268, 1293,
<code>hint</code> 1, 9315	1295, 1408, 1428, 1431, 1434, 1444,
<code>interpretmodule</code> 145, 3507	1543, 1961, 2278, 2284, 2290, 2296,
<code>mathstructure</code> 98, 156, 7031	2390, 2391, 2452, 2453, 2454, 2460,
<code>mcb</code> 1, 9529	2464, 2468, 2469, 2529, 2727, 2746,
<code>nassertion</code> 1, 1	2747, 2748, 2753, 2754, 2755, 3178,
<code>ndefinition</code> 1, 1	3245, 3246, 3247, 3483, 3485, 3549,
<code>nexample</code> 1, 1	3551, 3596, 3597, 3598, 3634, 4054,
<code>note</code> 1, 1	4411, 4570, 5093, 5166, 5167, 5172,
<code>nparagraph</code> 1, 1, 1, 1	5330, 5340, 5342, 5385, 5518, 5525,
<code>nsproof</code> 1, 1	5526, 5527, 5535, 5536, 5537, 5550,
<code>problem</code> 1	5732, 5734, 5839, 5840, 5841, 5854,
<code>sassertion</code> 101, 157, 7428	6089, 6145, 6169, 6569, 6582, 6927,
<code>scb</code> 9675	6955, 7145, 7148, 7220, 7326, 7340
<code>sdefinition</code> 101, 157, 7423	<code>\exp_args:Ne</code> 58,
<code>sexample</code> 101, 157, 7443	63, 69, 77, 81, 157, 265, 266, 274,
<code>sfragment</code> 84, 139, 1547	806, 814, 818, 903, 962, 1451, 1814,
<code>smodule</code> 88, 142, 2302	1835, 1837, 1853, 1854, 1966, 2043,
<code>solution</code> 1, 9245	2068, 2327, 2736, 2854, 3169, 3177,
<code>sparagraph</code> 101, 157, 7446	3178, 3228, 3664, 3876, 3945, 4022,
<code>spfblock</code> 158, 8072	4111, 4176, 4178, 4255, 4378, 4536,
<code>spfsketchenv</code> 158, 7959	4777, 4804, 4808, 4887, 4898, 4969,
<code>spfststepenv</code> 158, 8069	5008, 5026, 5156, 5434, 5652, 5753,
<code>sproblem</code> 9048	5755, 5763, 5769, 6033, 6050, 6173,
<code>sproof</code> 103, 158, 7677	6191, 6201, 6542, 6908, 7138, 7198,
<code>stex_annotate_env</code> 128, 709	7286, 7332, 7413, 7432, 7484, 7492,
<code>stex_env_node</code> 128, 709	7494, 8049, 8259, 8830, 8969, 8992,
<code>subproblem</code> 9162	9489, 9706, 9893, 10003, 10237, 10367
<code>subproof</code> 158, 7834	<code>\exp_args:NNe</code> 89, 474, 477,
<code>testheading</code> 77, 119	879, 889, 896, 1903, 1996, 2426,
<code>titlefragment</code> 139, 1606	2530, 3093, 3607, 4217, 4262, 4274,
	4989, 5105, 5119, 5214, 5233, 5466,
	5469, 5739, 5790, 5880, 6053, 6126,
	6389, 6485, 6496, 7160, 8876, 9156

<code>\exp_args:Nne</code>	304,	<code>\exp_not:N</code>	
1065, 1342, 1484, 1596, 1815, 2137,		94, 257, 308, 310, 913, 1080, 1081,	
2158, 2615, 2894, 3026, 3055, 3270,		1086, 1087, 1118, 1127, 1128, 1133,	
3834, 3867, 3935, 4110, 4343, 4742,		1245, 1246, 1414, 1415, 1820, 1905,	
5028, 5045, 5093, 5160, 5162, 5215,		1998, 2474, 2725, 2745, 2753, 3056,	
5503, 5539, 5549, 6328, 6348, 6516,		3671, 4600, 4601, 4604, 4608, 4612,	
6681, 6711, 7053, 7153, 7156, 7159,		4669, 4745, 4748, 4842, 4918, 4992,	
7297, 7383, 7492, 7494, 7528, 7564,		4998, 5018, 5051, 5080, 5083, 5107,	
7569, 7576, 7579, 7718, 7734, 7966,		5113, 5122, 5127, 5136, 5171, 5394,	
7990, 8024, 9084, 9184, 9542, 10366		5400, 5513, 5791, 5796, 5875, 5881,	
<code>\exp_args:NNne</code>	4264, 6789	5885, 5887, 5888, 5910, 6357, 6764,	
<code>\exp_args:NNne</code> 4351, 4509, 5445, 5451		8385, 8390, 9657, 9659, 9791, 9793	
<code>\exp_args:NNno</code>		<code>\exp_not:n</code>	257,
.	1017, 2877, 3617, 4351, 4509	308, 914, 1596, 1823, 1824, 1933,	
<code>\exp_args:NNno</code>	143	1939, 1978, 1980, 1998, 2468, 2479,	
<code>\exp_args:Nnno</code>	3188, 3190, 5919	2534, 3895, 3942, 4103, 4357, 4358,	
<code>\exp_args:NNNx</code>	1017	4359, 4369, 4370, 4371, 4515, 4516,	
<code>\exp_args:NNo</code>		4517, 4528, 4529, 4530, 4602, 4605,	
13, 42, 63, 83, 104, 107, 127, 131,		4609, 4611, 4613, 4753, 4843, 4881,	
156, 222, 348, 750, 800, 1020, 1342,		4912, 4917, 4919, 4991, 4993, 4995,	
1876, 1965, 2206, 2217, 2422, 2894,		4999, 5002, 5012, 5081, 5084, 5108,	
2926, 3614, 3620, 3624, 5242, 5243,		5110, 5114, 5116, 5121, 5125, 5134,	
5253, 5254, 5867, 6436, 6449, 6837		5171, 5526, 5542, 5547, 5552, 5554,	
<code>\exp_args:Nno</code>		5776, 5797, 5834, 5839, 5877, 5889,	
. . 335, 337, 372, 1911, 1943, 2504,		5890, 5912, 6358, 6398, 6399, 6758,	
3172, 4049, 4217, 4262, 4274, 4406,		6927, 6955, 7866, 9662, 9796, 9864	
4555, 4565, 4587, 4765, 4769, 4978,		<code>\expandafter</code>	48,
7843, 9071, 9241, 10000, 10294, 10339		663, 2166, 2167, 2168, 2169, 2170,	
<code>\exp_args:NNx</code>		2580, 5965, 7838, 7871, 8482, 8483	
. . . . 90, 93, 426, 439, 9007, 9010,		<code>\ExplSyntaxOff</code> 2604, 2652, 8266, 9020, 9975	
9013, 9016, 9962, 9965, 9968, 9971		<code>\ExplSyntaxOn</code>	
<code>\exp_args:No</code> 48, 257, 308, 348, 570,		. . . 143, 2600, 2651, 8263, 9003, 9958	
576, 657, 668, 785, 914, 1029, 1521,		<code>\extref</code>	87, 2036
1596, 1615, 1875, 1883, 1884, 1910,		<code>extstructure</code> (env.)	98, 156, 7066
1921, 1924, 1931, 1933, 1939, 1940,		<code>\extstructure</code>	7095, 7097
1941, 1950, 1964, 1971, 1980, 1998,		<code>extstructure*</code> (env.)	98
2003, 2031, 2032, 2267, 2343, 2376,			
2436, 2534, 2566, 2852, 2856, 3088,			
3092, 3175, 3255, 3388, 3409, 3424,			
3432, 3631, 3637, 3638, 3681, 3863,			
3926, 3942, 4102, 4107, 4215, 4264,			
4357, 4358, 4359, 4369, 4370, 4371,			
4515, 4516, 4517, 4528, 4529, 4530,			
4602, 4605, 4609, 4636, 4729, 4843,			
4847, 4854, 4881, 4917, 4919, 4991,			
4993, 4999, 5081, 5108, 5114, 5116,			
5121, 5125, 5134, 5157, 5171, 5232,			
5443, 5471, 5542, 5547, 5552, 5554,			
5753, 5755, 5797, 5877, 5890, 5912,			
6216, 6358, 6758, 6945, 7050, 7058,			
7114, 7117, 7139, 7199, 7219, 7390,			
7472, 8374, 8438, 8439, 8442, 9006,			
9092, 9192, 9195, 9961, 10004, 10468			
<code>\exp_args:Nx</code>	8507, 9581		

F

<code>\fbox</code>	9892, 9908, 9921
<code>\fi</code>	665, 680, 1475, 2084,
2108, 2119, 2748, 2755, 4647, 4678,	
5264, 5965, 6138, 6159, 8484, 8552,	
8555, 8673, 8886, 8954, 8971, 8994,	
9169, 9282, 9295, 9650, 9665, 9670,	
9784, 9799, 9804, 9926, 9940, 10000,	
10013, 10023, 10531, 10541, 10542	
<code>fiboxed</code>	60, 108
file commands:	
<code>\file_if_exist:nTF</code>	612, 631, 997
<code>\fillinsol</code>	117, 9052, 9832
<code>\fn</code>	47
<code>\foldexpr</code>	152, 6188
<code>\foo</code>	16, 47, 90
<code>\fooname</code>	16

\footnote	9669, 9803, 9910
\footnotesize	10368
\foral	40
\forall	40, 8204
fp commands:		
\fp_gadd:Nn	9115, 9116
\fp_new:N	9058, 9059
\fp_to_decimal:n	9202, 9205
frame (env.)	1, 1, 8506
\frameimage	62, 111, 8716
frameimages	60, 108
\frametitle	8620
\frontmatter	210, 1745, 1746, 1749
\FTMLrule	8269, 8293, 8294
\fun	39
\funspace	34
G		
\gdef	1497, 1514, 1519, 8820, 9993, 9995, 10003, 10004, 10020
\given	76, 119
\global	1493, 2167, 8248, 8249, 9310, 9313, 10022, 10301, 10303
gnote (env.)	1, 9414
\gnote	9055, 9465, 9562, 9727
gnotes	76, 113, 118
group commands:		
\group_begin: 18, 78, 87, 613, 755, 841, 2506, 2622, 2686, 2695, 2703, 2944, 2980, 3017, 3053, 3076, 3794, 3825, 3899, 4338, 4497, 4619, 4730, 4734, 5010, 5035, 5153, 5261, 5276, 5494, 5537, 5665, 5710, 5785, 5833, 5849, 5941, 6011, 6028, 6045, 6059, 6071, 6085, 6160, 6233, 6256, 6311, 6343, 6993, 7327, 7456, 7511, 7774, 7823, 8023, 8048, 8134, 9238, 9977, 10336
\group_end:	82, 85, 93, 621, 822, 845, 2520, 2631, 2699, 2706, 2709, 2949, 2986, 3022, 3055, 3129, 3802, 3839, 3904, 4348, 4505, 4680, 4693, 4719, 4731, 4735, 4765, 4926, 4982, 5019, 5040, 5046, 5119, 5172, 5290, 5327, 5338, 5354, 5365, 5373, 5406, 5497, 5518, 5550, 5555, 5673, 5710, 5790, 5841, 5880, 5945, 6156, 6178, 6235, 6256, 6314, 6353, 6999, 7336, 7458, 7522, 7780, 7832, 8042, 8066, 8251, 9242, 9980, 10340
\group_insert_after:N	340, 354
H		
\halign	9911
\hbox	1506, 2940, 3013, 3577, 3592, 3887, 3917, 4022, 4378, 4551, 6687, 7079, 7257, 7690, 7757, 7883, 8121, 8276, 8588, 8808, 8811, 9606, 9637, 9879
hbox commands:		
\hbox_set:Nn	6372, 9892
\hbox_unpack:N 1530, 1540, 3916, 9522, 9527
\HCode	675
\hfil	8127, 991
\hfill	8127
hint (env.)	1, 9315
\hint	9053, 9365, 9560, 9725
hints	76, 113, 118
\hline	..	10366, 10368, 10369, 10370, 10371
\homeworkdata	10488
homeworkdata	10416
\href	1971, 2050, 8830
\hrule	8121, 9606, 9637
\hskip	9892
\hspace	9644, 9778, 9924
\HTML	16, 25
\huge	9892, 9922, 9924
hwexam commands:		
\l_hwexam_includeassignment_	-	keys_tl .. 10292, 10293, 10295, 10337
\hwexam_kw_*	9957
\c_hwexam_multiple_bool	9948
\c_hwexam_qrcode_bool	...	9950, 9982
\hwexamheader	10373, 10412
\hwexamminutes	10375
\hyperlink	2048
I		
\if	2746, 2753
if commands:		
\if_charcode:w	10000
\IfBooleanTF	..	3490, 3556, 3779, 5422, 6305, 6340, 7889, 8002, 8090, 8717
\ifcsname	662, 8886
\ifdefempty	8875
\IfFileExists	6, 21, 1884, 1950, 2065, 2205, 2216, 2897, 2902, 2911
\IfInputref	27, 85, 140, 2112, 8873
\ifinputref	..	27, 85, 140, 2102, 2111, 2119
\ifintest	8939, 8948, 9940
\ifmmode	5965
\ifnotes	61, 109, 8365, 8502
\ifnum	2079
\ifSGvar	63, 112, 8894
\ifsolutions	114,	8931, 9275, 9288, 9641, 9654, 9668, 9775, 9788, 9802, 9906
\ifstexhtml	27, 83, 128, 685, 1469, 8967, 8990, 9167, 10010, 10016

<code>\ifvmode</code>	4647, 4678, 5264, 6138, 6159		1638, 1663, 1667, 1703, 1709, 4011,
<code>\ifx</code>	2166,		4012, 4013, 4014, 5728, 6867, 6950,
	8481, 8550, 8553, 10524, 10525, 10533		7620, 7632, 7643, 7660, 7672, 8282,
<code>\ignorespaces</code>	2935, 2957,		8424, 8775, 8779, 9479, 10214, 10354
	3006, 4348, 4505, 8676, 8682, 9226		
<code>image</code>	120		
<code>\importmodule</code>	35, 43, 48, 96,		
	97, 124, 128, 143, 145, 222, 2954, 3312		
<code>\includeassignment</code>	76, 10335		
<code>\includegraphics</code>	85, 141, 2234, 10517		
<code>\includeproblem</code>	75, 117, 9230		
<code>\indent</code>	4647, 4678, 5264, 6138, 6159		
<code>\infpref</code>	92, 93, 154, 6436, 6971, 8156		
<code>\inline</code>	157		
<code>\inline*</code>	102		
<code>\inlineass</code>	44, 48, 51, 102, 157		
<code>\inlinedef</code>	44, 102, 157		
<code>\inlineex</code>	102, 157		
<code>\input</code>	26, 82, 125, 143, 182, 44, 246,		
	684, 2129, 9004, 9008, 9011, 9014,		
	9017, 9959, 9963, 9966, 9969, 9972,		
	10373, 10526, 10529, 10535, 10539		
<code>\inputassignment</code>	119		
<code>\inputref</code>	26,		
	27, 84, 85, 104, 124, 140, 2054,		
	8735, 8736, 8844, 8876, 9241, 10339		
<code>\inputref*</code>	61, 110, 8735		
<code>\inputreffalse</code>	2107, 2111		
<code>\inputreftrue</code>	2090, 2105		
<code>\insertexnumber</code>	10015, 10022, 10027		
<code>\insertframenumber</code>	8814, 8816		
<code>\insertshortauthor</code>	8807		
<code>\insertshortdate</code>	8818		
<code>\insertshorttitle</code>	8810		
<code>\inset</code>	39		
int commands:			
<code>\int_case:nn</code>	1629		
<code>\int_case:nnTF</code>	1562, 1661, 1699, 1718		
<code>\int_compare:nNnTF</code>			
	328, 1599, 1637, 2135,		
	2157, 3608, 4839, 4850, 4943, 5356,		
	5752, 5762, 5768, 5827, 6363, 6397,		
	6951, 6981, 7358, 7396, 7645, 10353		
<code>\int_compare_p:nNn</code>			
	7615, 7627, 7639, 7656, 7668		
<code>\int_decr:N</code>	7646, 7674		
<code>\int_eval:n</code>	347, 349, 351,		
	5448, 5454, 6980, 7186, 10301, 10367		
<code>\int_gincr:N</code>	5433,		
	9200, 9258, 9321, 9369, 9419, 10347		
<code>\int_gset:Nn</code>	5398		
<code>\int_gzero:N</code>	5383, 9109		
<code>\int_incr:N</code>			
	1564, 1568, 1600, 1631, 1634,		
	1638, 1663, 1667, 1703, 1709, 4011,		
	4012, 4013, 4014, 5728, 6867, 6950,		
	7620, 7632, 7643, 7660, 7672, 8282,		
	8424, 8775, 8779, 9479, 10214, 10354		
<code>\int_new:N</code>	323, 1526,		
	2008, 3967, 4163, 5375, 5712, 6937,		
	6967, 8771, 9162, 9245, 9414, 10345		
<code>\int_set:Nn</code>	1678, 1679, 1680, 1681,		
	1682, 1683, 1685, 3399, 3882, 3976,		
	3980, 3984, 3988, 3992, 3996, 4000,		
	4004, 4008, 4199, 4239, 4300, 4328,		
	4460, 5670, 6840, 6945, 6952, 6974,		
	7613, 7625, 7637, 7654, 7666, 10439		
<code>\int_set_eq:NN</code>	325		
<code>\int_step_function:nN</code>	4746, 5506		
<code>\int_step_inline:nn</code>			
	3960, 6428, 6435, 6455, 6775		
<code>\int_use:N</code>	1582, 1645, 1689, 1692,		
	1713, 3938, 4355, 4367, 4513, 4526,		
	4589, 5398, 5434, 5775, 5793, 5883,		
	6504, 6609, 6840, 6971, 6972, 7385,		
	7860, 7866, 7871, 8285, 8417, 8776,		
	8780, 9215, 9260, 9323, 9371, 9421,		
	9481, 10118, 10119, 10141, 10158,		
	10159, 10182, 10216, 10366, 10473		
<code>\int_zero:N</code>	3970,		
	3971, 5719, 6822, 6948, 8275, 8773		
<code>\c_max_int</code>	6971, 6972		
<code>\l_tmpa_int</code>			
	6822, 6825, 6840, 6867, 6948, 6950,		
	6951, 6952, 6956, 7613, 7616, 7619,		
	7620, 7625, 7628, 7631, 7632, 7637,		
	7640, 7643, 7645, 7646, 7648, 7649,		
	7654, 7657, 7660, 7662, 7666, 7669,		
	7672, 7674, 7675, 8275, 8282, 8285		
intarray commands:			
<code>\intarray_gset:Nnn</code>			
	7648, 7662, 7675, 7681		
<code>\intarray_gzero:N</code>	7680		
<code>\intarray_item:Nn</code>	7616, 7619,		
	7628, 7631, 7640, 7649, 7657, 7669		
<code>\intarray_new:Nn</code>	7610		
<code>\interpretmod</code>	145, 3569, 3571, 3573		
<code>interpretmodule (env.)</code>	145, 3507		
<code>\interpretmodule</code>	3543, 3545		
<code>\intestfalse</code>	8943		
<code>\intesttrue</code>	8941		
ior commands:			
<code>\ior_close:N</code>	638, 644, 846, 1028		
<code>\ior_map_inline:Nn</code>	1016		
<code>\ior_new:N</code>	1002		
<code>\ior_open:Nn</code>	632, 640, 1011		
<code>\ior_open:NnTF</code>	840		
<code>\ior_str_get:NN</code>	843		

<code>\ior_str_map_inline:Nn</code>	634, 641	<code>\libusepackage</code> . . .	82, 83, 140, <u>213</u> , 8498
<code>\g_tmpa_ior</code>	632, 634, 638, 640, 641, 644, 840, 843, 846	<code>\libusestheme</code>	61, 8497
ioe commands:		<code>\libusetikzlibrary</code> . . .	82, 121, 140, <u>214</u>
<code>\iow_close:N</code> . . .	627, 637, 1798, 10267	<code>\linewidth</code> . . .	9298, 9305, 9358, 9363, 9406, 9411, 9458, 9463, 10195, 10208
<code>\iow_new:N</code>	585, 1796, 10264	<code>\list</code>	7823
<code>\iow_now:Nn</code>	635, 642, 730, 1816, 1819, 8262, 9156, 10266	<code>\LoadClass</code>	16, 8330, 8332
<code>\iow_open:Nn</code> . . .	626, 633, 1797, 10265	<code>\loadsms</code>	127, <u>586</u> , 587
<code>\g_tmpa_iow</code>	633, 635, 637	<code>\long</code>	8789, 8798
<code>\isassociative</code>	43	<code>\lstinputlisting</code>	85, <u>141</u> , 2250
<code>\iscommutative</code>	43	<code>\lstinputmhlisting</code>	85, <u>141</u> , <u>2238</u>
<code>\item</code>	7827, 9215, 9489, 9508, 9536, 9701, 10118, 10158, 10235	M	
<code>\itshape</code>	8131	<code>\macro</code>	123, 124, 126, 135, 144
J		<code>\macroname</code>	31, 127
<code>\jobname</code>	79, 235, 604, 626, 631, 632, 633, 640, 645, 1797, 10265	<code>\magma</code>	46
<code>\join</code>	56	<code>\maincomp</code>	31–34, 47, 92, 100, 105, 4430, 4660, 4679, 4777, 4778, 5008, 5016, 5021, 5026, 5029, 5038, 5372, 5950, 6357, 6366, 6376, 6377, 6398, 6704, 6764, 7204, 8148, 8190, 8196, 8215, 8219, 8222, 8224, 8239
K		<code>\mainmatter</code>	1773, 1774, 1784, 1785
keys commands:		<code>\makeatletter</code>	2649, 9003, 9958, 10412
<code>\l_keys_choice_tl</code>	3753	<code>\makeatother</code>	2650, 9020, 9975, 10412
<code>\keys_define:nn</code>	29, 372, 8302, 8344, 8854, 8910, 9946, 10505	<code>\makebox</code>	9643, 9777
<code>\l_keys_key_str</code>	6279, 6282, 7024, 7027	<code>\maketitle</code>	139, 1518, 1519, 1610, 1619, 8430, 8431
<code>\l_keys_key_tl</code> . . .	13, 750, 6280, 7025	<code>\mapsto</code>	8191, 8197
<code>\keys_set:nn</code>	126, 376, 8863	<code>\marginnote</code>	8548, 10011
keyval commands:		<code>\marginpar</code>	152, 6232, 8968, 8991, 9125, 9130, 9218, 9223, 9521, 9526, 10034, 10039, 10076, 10081, 10122, 10127, 10163, 10168
<code>\keyval_parse:NNn</code>	4862	<code>\mathbb</code>	8227
L		<code>\mathbin</code>	32, 8154, 8158, 8167, 8203, 8214, 8218
<code>\label</code>	86, 1814, 1835, 1837, 8613	<code>\mathcal</code>	5875, 5877, 5910, 5912
<code>\labelsep</code>	8558	<code>\mathclose</code>	32, 7000, 8156, 8165, 8169, 8175, 8177, 8187, 8203, 8207
<code>\labelwidth</code>	8559	<code>\mathhub</code>	80, 132, 32, <u>837</u>
<code>\langle</code>	8175	<code>\mathop</code>	32, 8205, 8224
<code>\Large</code>	8362, 8745, 9572, 9573, 9576, 9577, 9737, 9738, 9741, 9742, 10018, 10020, 10317	<code>\mathopen</code>	32, 6995, 8156, 8165, 8169, 8175, 8177, 8186, 8203, 8207
<code>\lastslide</code>	8763, 8778	<code>\mathord</code>	32
<code>\LaTeX</code>	23	<code>\mathpunct</code>	32, 5702, 6779, 8186, 8190, 8196
<code>\latex</code>	23	<code>\mathrel</code>	31, 32, 8159, 8191, 8197, 8215, 8219, 8222
<code>\ldots</code>	8180, 8243, 8245, 8247	<code>\mathrm</code>	6764, 8148, 8161, 8186, 8190, 8196, 8239
<code>\leaders</code>	8747	<code>mathstructure (env.)</code>	98, <u>156</u> , <u>7031</u>
<code>\leavevmode</code>	8633	<code>\mathstructure</code>	7040, 7041
<code>\leftmargin</code>	8560	<code>\mathtt</code>	8164, 8165, 8230, 8233, 8237
<code>\let</code>	1746, 1747, 1748, 1749, 1759, 1760, 1761, 1762, 2091, 2167, 2688, 2696, 2704, 2726, 2731, 3000, 3917, 4658, 4679, 6016, 6039, 6056, 6151, 7210, 8248, 8249, 8264, 8265, 8431, 8754, 9678, 9682, 9685, 9689	<code>mcb (env.)</code>	1, 9529
<code>\libinput</code>	19, 82, 140, 2122, 2192		

\mcb	9050, 9557, 9579, 9722	\neginfprec	37, 92, 93, 154, 4691, 5350, 5361,
\mcc	70, 115, 9563, 9600		6406, 6409, 6418, 6421, 6434, 6971
\meaning	1959,	\newcommand	483, 517, 520,
	2580, 3135, 3138, 3141, 4690, 5170,		2024, 2113, 2118, 2122, 2133, 2144,
	5533, 5545, 5546, 6402, 6707, 6862		2179, 2245, 2251, 2261, 2271, 8107,
\medskip	2096, 8588, 8603, 8627		8111, 8115, 8479, 8548, 8629, 8738,
\meet	56		8742, 8825, 8836, 8843, 8846, 8868,
\message	28,		8870, 8956, 8979, 9041, 9476, 9610,
	1865, 2752, 8323, 8326, 8488, 10262		9652, 9748, 9786, 9812, 9817, 9822,
\mhframeimage	62, 111		9827, 9876, 9931, 9934, 9935, 9936,
\mhgraphics	85, 141, 2222, 8729, 8731		9937, 9938, 9939, 10352, 10482,
\mhinput	85, 140, 2099		10488, 10492, 10517, 10522, 10546
\mhtikzinput	121, 141, 2253	\newcounter	8333, 8334,
\min	77, 119		8335, 8336, 8463, 8475, 9040, 10290
\min	9524	\NewDocumentCommand ..	127, 486, 1468,
min	76, 113, 118		1829, 1840, 2036, 2041, 2054, 2099,
\mmlarg	129, 712		2588, 4426, 4429, 5275, 5421, 6010,
\mmlintent	129, 712		6027, 6044, 6058, 6070, 6084, 6107,
\mname	89, 98		6113, 6123, 6133, 6189, 6200, 7326,
mode commands:			7455, 7501, 7510, 7525, 7986, 8085,
\mode_if_math:TF			8103, 8269, 8497, 8716, 9237, 10335
..	3592, 3917, 3918, 4675, 4680, 9878	\NewDocumentEnironment	127
\mode_if_vertical:TF	1530, 1540, 3916	\NewDocumentEnvironment	
\mode_leave_vertical:	3249		523, 1553, 1608, 1652, 8072, 8757, 8772
\MSC	8254	\newenvironment	1770, 1781, 7959,
msg commands:			8069, 8682, 8684, 8689, 8691, 10395
\msg_error:nn		\newif 663, 687, 2111, 8365, 8931, 8939, 10402	
..	3060, 3241, 5677, 7468, 7517, 9067	\newlabel	8264, 8265
\msg_error:nnn		\newlength	8500, 8501, 8504
.....	58, 469, 795, 2146, 2192,	\newline	9298, 9305, 9358, 9363,
	2576, 2823, 2928, 3694, 3724, 4167,		9406, 9411, 9458, 9463, 10195, 10208
	4436, 5140, 5218, 6910, 7213, 9854	\newpage	8610, 9938, 10414
\msg_error:nnnn ..	70, 2127, 2184,	\newsavebox	8651
	3298, 3374, 4016, 4627, 5272, 5288,	\newseq	148, 4429
	5459, 5486, 6154, 6176, 6635, 7180	\newvar	148, 4426
\msg_fatal:nn	947, 10455, 10462, 10465	nexample (env.)	1, 1
\msg_fatal:nnnn	951, 2141, 2160	\nextslide	8759, 8774
\msg_none:nn	117	\ninputref	8736, 8738, 8742
\msg_redirect_module:nnn	132	\nobreak	8127
\msg_redirect_name:nnn	136	\noexpand	2746
\msg_set:nnn		\noindent	1583, 2067, 2080, 7304,
.....	114, 9845, 10422, 10426, 10430		7708, 7915, 7978, 8047, 8603, 8623,
\msg_warning:nn	859		8631, 8649, 9090, 9121, 9190, 9299,
\msg_warning:nnnn	651		9359, 9407, 9459, 10031, 10072, 10196
\mult	37, 93, 94	\nointerlineskip	8749
\multicolumn	10367	\normalmarginpar	10012
\multiple	76, 118	\notation	31, 32, 37, 90, 92, 97,
			143, 153, 3311, 6295, 8151, 8154,
			8157, 8158, 8159, 8174, 8176, 8202,
			8204, 8205, 8213, 8217, 8227, 8230
N		note (env.)	1, 1
nassertion (env.)	1, 1	notes	60, 76, 108, 113, 118
\Nat	34	\notesfalse	8377
ndefinition (env.)	1, 1		

notesslides commands:	
\c_notesslides_notes_bool	_notesslides_notes_env:nnnn ...
8304, 8305, 8318, 8321, 8329, 8345,	8687,
8346, 8358, 8366, 8494, 8675, 8681,	8707, 8708, 8709, 8710, 8711, 8712
8688, 8722, 8737, 8787, 8837, 8847	\l_notesslides_num
\c_notesslides_sectocframes_bool	8383, 8385, 8388,
..... 8347, 8450	8390, 8393, 8394, 8397, 8398, 8401,
\c_notesslides_topsect_str 8348,	8402, 8405, 8406, 8409, 8410, 8417
8371, 8374, 8435, 8438, 8439, 8442	_notesslides_setup_itemize: ...
notesslides internal commands: 8544, 8615
\c_notesslides_class_str	\l_notesslides_slideshow_-
..... 8300, 8303, 8309, 8330	counter_int
_notesslides_define_chapter: 8771, 8773, 8775, 8776, 8779, 8780
..... 8440, 8443, 8455	\l_notesslides_tmp
_notesslides_define_part:	\l_notesslides_uri
..... 8444, 8467	\g_notesslides_variables_prop ..
_notesslides_do_label:n 8380, 8416 8887, 8889, 8892, 8895
_notesslides_do_sectocframes: .	\notesslidesfont
..... 8379, 8451	8584, 8600, 8754
_notesslides_do_yes_param:Nn ..	\notesslidesfooter
..... 8506,	8588, 8603, 8752
8525, 8528, 8531, 8534, 8537, 8540	\notesslidestitleemph ..
\c_notesslides_docopt_str ... 8306	8623, 8625, 8744
\c_notesslides_document_str ...	\notesttrue
..... 8478, 8481	8367
_notesslides_eat: . 8692, 8696, 8699	nparagraph (env.)
_notesslides_excursion_args:n .	1, 1, 1, 1
..... 8859, 8871	nsproof (env.)
\l_notesslides_excursion_id_str	1, 1
..... 8855, 8861	\null
\l_notesslides_excursion_intro_-	8127
tl	\number
8856, 8860, 8875, 8877	76, 119
\l_notesslides_excursion_-	\numberline
mhrepos_str	8415
8857, 8862, 8876	\numberproblemsin
\l_notesslides_frame_allowdisplaybreaks_-	9040
bool	
8517, 8528	O
\l_notesslides_frame_allowframebreaks	\objective
bool	8979
8516, 8525	\OMDoc
_notesslides_frame_box_begin: .	25
..... 8581, 8592, 8616	\omdoc
_notesslides_frame_box_end: ...	16
..... 8586, 8602, 8618	\only
\l_notesslides_frame_fragile_-	8776, 8780
bool	\oplus
8518, 8531	8166, 8167
\l_notesslides_frame_label_str .	P
..... 8515, 8523, 8612, 8613	\PackageError
\l_notesslides_frame_shrink_-	8896
bool	\pagebreak
8519, 8534	9138, 10333
\l_notesslides_frame_squeeze_-	\pagenumbering ...
bool	1754, 1767, 1778, 1789
8520, 8537	\par ..
\l_notesslides_frame_t_bool ...	47, 1581, 1585, 2080, 7679, 7708,
..... 8521, 8540	7773, 7779, 7822, 7831, 7911, 7915,
_notesslides_inputref:	7960, 7965, 7974, 8040, 8127, 8588,
..... 8735, 8736, 8739	8603, 8623, 8631, 8636, 8644, 8649,
	8656, 8666, 9121, 9134, 9137, 9168,
	9298, 9305, 9358, 9363, 9406, 9411,
	9458, 9463, 9509, 9512, 9569, 9690,
	9734, 10031, 10060, 10068, 10072,
	10102, 10110, 10195, 10208, 10251,
	10254, 10316, 10321, 10331, 10333
	\paragraph
	26, 84
	\parent
	125
	\parsep
	7825, 9213, 9487,
	9506, 9534, 9699, 10116, 10156, 10233
	\part
	26, 84, 8471, 8472
	\partname
	8385, 8468, 8469

\parttitlename	8385	\g_problem_total_pts_fp ..	9058, 9115
\PassOptionsToClass	8308, 8309	\problemheader ..	9121, 9141, 10031, 10072
\PassOptionsToPackage		problems internal commands:	
..... 10, 8310, 8311, 8322,		__problems_activate_macros: ...	
8325, 8350, 8351, 8368, 8928, 9952	 9048, 9113, 9170	
\pause	8629	\l__problems_anscls_int	
\PDF	16, 25 9414, 9479, 9481, 10214, 10216	
\pdfbookmark	8417	\c__problems_boxed_bool	8924
peek commands:		__problems_do_yes_param:Nn ..	9580
\peek_charcode:NTF		__problems_exnote_start:n	
1393, 3643, 4684, 4697, 4757, 4759,	 9366, 9383, 9386	
4786, 4792, 4935, 4946, 5298, 5305		\c__problems_false_tl ...	9843, 9853
\peek_charcode_remove:NTF		\l__problems_fillin_solution_tl .	
... 4683, 4756, 4791, 5044, 5294,		.. 9833, 9857, 9869, 9898, 9909, 9916	
5296, 5303, 5313, 5316, 5517, 5560		__problems_fillinsol:nn	
\phantom	9922 9879, 9881, 9887, 9904	
\plus	35–38, 92–94	__problems_gnote_start:n	
\precondition	8956 9416, 9435, 9438	
\prematurestop	63, 111, 8478	\c__problems_gnotes_bool	
\premise	103, 157, 7437, 7525 8914, 9437, 9448, 9992, 10263	
prg commands:		\l__problems_has_min_bool .	9063,
\prg_new_conditional:Nnn		9107, 9108, 9204, 9222, 10126, 10167	
. 57, 62, 263, 272, 695, 2553, 2557,		\l__problems_has_pts_bool	
2561, 2675, 4106, 4109, 4284, 5465,	 9062, 9104, 9105,	
5751, 5761, 5767, 6210, 6221, 6225		9201, 9217, 10121, 10143, 10162, 10184	
\prg_new_protected_conditional:Nnn		__problems_hint_start:n	
..... 279	 9318, 9335, 9338	
\prg_return_false:	60,	\c__problems_hints_bool	
65, 264, 267, 273, 275, 295, 299,	 8916, 9337, 9348	
697, 2555, 2559, 2563, 2676, 4107,		\l__problems_in_problem_bool ...	
4118, 5483, 5487, 5757, 5758, 5759,		.. 9061, 9064, 9066, 9100, 9166, 9171	
5764, 5765, 5770, 5771, 6211, 6226		__problems_maybe_inline:n	
\prg_return_true: 9541, 9567, 9676, 9705, 9732	
..... 60, 65, 265, 267, 275, 299,		__problems_maybe_newline:	
697, 2555, 2559, 2563, 2676, 4107,	 9601, 9680, 9687	
4118, 5481, 5757, 5764, 5770, 6222		__problems_mccline:n	9531,
\printexcursion	64, 112	9550, 9553, 9572, 9576, 9612, 9667	
\printexcursions ..	8820, 8844, 8872, 8880	\c__problems_min_bool 8922, 9128,	
problem (env.)	1	9221, 9525, 10037, 10079, 10125, 10166	
problem commands:		\l__problems_min_tl	
\g_problem_id_counter		9102, 9106, 9116, 9129, 9130, 9157,	
..... 9245, 9258, 9260,		9173, 9205, 10038, 10039, 10080, 10081	
9321, 9323, 9369, 9371, 9419, 9421		\c__problems_notes_bool	
\l_problem_inputproblem_keys_tl 8912, 9385, 9396	
..... 9069, 9070, 9072, 9239		__problems_oldpar:	
\problem_mcc_box_default_tl 9678, 9682, 9685, 9689	
..... 9635, 9643, 9647, 9649		__problems_parse_fillin_-	
\problem_mcc_box_tl		arg:nnnn	
.. 9536, 9540, 9572, 9576, 9604, 9640		.. 9837, 9838, 9839, 9850, 9868, 9890	
\problem_scc_box_default_tl		\l__problems_path_seq	
..... 9773, 9777, 9781, 9783		.. 9081, 9082, 9176, 9177, 9181, 9182	
\problem_scc_box_tl		\l__problems_prob_imports_tl .	9037
.. 9701, 9713, 9737, 9741, 9747, 9774		\l__problems_prob_refnum_int .	9038
\g_problem_total_min_fp ..	9059, 9116		

<code>\reversemarginpar</code>	10011	<code>\seq_map_function:NN</code> 182, 186, 2129,	
<code>\rhd</code>	8551, 8554	2186, 3152, 4884, 7969, 8027, 8052	
<code>\Rightarrow</code>	8084, 8219	<code>\seq_map_indexed_function:NN</code> ...	
<code>\rightmargin</code>	7826, 9214, 9488,	7347, 7721, 7737
9507, 9535, 9700, 10117, 10157, 10234		<code>\seq_map_inline:Nn</code>	
<code>\rule</code>	9298, 9305, 9358, 9363,	866, 958, 1060, 1064,
9406, 9411, 9458, 9463, 10195, 10208		1225, 2197, 3144, 3165, 3197, 3286,	
rustex commands:		3366, 3370, 3427, 3435, 4130, 4253,	
<code>\rustex_direct_HTML:n</code>		5861, 6487, 6922, 6949, 7077, 7119,	
.....	8637, 8645, 8657, 8667	7255, 7412, 7431, 7687, 7880, 8281	
<code>\rustex_if:TF</code>	8635, 8636,	<code>\seq_new:N</code>	
8643, 8644, 8655, 8656, 8665, 8666		24, 761, 2301, 2644, 2664, 2679
<code>\rustexBREAK</code>	2653	<code>\seq_pop:NN</code>	179
		<code>\seq_pop_left:NN</code>	
		163, 292, 293, 1064, 1226,
		1228, 2210, 3610, 3612, 3619, 3623	
		<code>\seq_pop_left:NNTF</code>	
		1019, 6446, 6448, 6456
		<code>\seq_pop_right:NN</code>	
		. 197, 210, 212, 217, 219, 221, 979,	
		1190, 1192, 1204, 1277, 1317, 1320,	
		2198, 2878, 4213, 4250, 4875, 6790	
		<code>\seq_push:Nn</code> 185, 862, 2687, 3105, 3671	
		<code>\seq_put_left:Nn</code>	
		164, 183, 2445, 4148, 5178, 5192, 6651	
		<code>\seq_put_right:Nn</code> 200, 213, 223, 226,	
		235, 467, 873, 1209, 2207, 2211,	
		2218, 2385, 2572, 3093, 3114, 3229,	
		3246, 3348, 3597, 5214, 5774, 5791,	
		5825, 5832, 5857, 5863, 5867, 5881,	
		6429, 6436, 6457, 6459, 6776, 6924,	
		6926, 7072, 7107, 7252, 7287, 7377	
		<code>\seq_reverse:N</code>	4876
		<code>\seq_set_eq:NN</code>	
		188, 209, 216, 234, 280,
		281, 1063, 1189, 1224, 5872, 6931, 6959	
		<code>\seq_set_split:Nnn</code>	
		143, 211, 218, 865, 970, 971, 1018,	
		1061, 1191, 1223, 1276, 1316, 2195,	
		2196, 2877, 3607, 3617, 4212, 4249,	
		4874, 6445, 6449, 6485, 6789, 8273	
		<code>\seq_use:Nn</code>	206,
		213, 226, 472, 475, 478, 879, 952,	
		1021, 1215, 1282, 2125, 2182, 2200,	
		2879, 3615, 3621, 3625, 4218, 4251,	
		5401, 5701, 6691, 6779, 6901, 6932	
		<code>\l_tmpa_seq</code>	
		456, 467, 472, 475, 478, 860,
		862, 865, 866, 970, 974, 1061, 1062,	
		1064, 1191, 1192, 1193, 1204, 1209,	
		1215, 1223, 1225, 1316, 1317, 1319,	
		1320, 1373, 1378, 2877, 2878, 2879,	
		4249, 4250, 4251, 4872, 4874, 4875,	
		4876, 4884, 6485, 6487, 6774, 6776,	

S

<code>sassertion (env.)</code>	101, 157, <u>7428</u>
<code>scb (env.)</code>	<u>9675</u>
<code>\scb</code>	9051, 9558, 9723, 9744
<code>\scc</code>	9728, <u>9745</u>
<code>\scriptsize</code>	8548
<code>\scriptstyle</code>	8554
<code>sdefinition (env.)</code>	101, 157, <u>7423</u>
<code>\section</code>	26, 84, 139
<code>\sectiontitleemph</code>	8361, 8420
<code>sectocframes</code>	60, 108
seq commands:	
<code>\seq_clear:N</code>	142, 180, 456,
861, 864, 2092, 2194, 2689, 3083,	
3084, 3085, 3086, 3102, 3226, 3323,	
4138, 4872, 5146, 5147, 5155, 5718,	
5786, 5851, 5860, 6361, 6641, 6774,	
6921, 7067, 7102, 7249, 7284, 8271	
<code>\seq_count:N</code> ..	2135, 2157, 3608, 7396
<code>\seq_gclear:N</code>	2682, 2683, 5382
<code>\seq_get_right:NN</code> ...	9082, 9177, 9182
<code>\seq_gpush:Nn</code>	64, 801
<code>\seq_gput_right:Nn</code> ..	2647, 2667, 5411
<code>\seq_gset_eq:NN</code>	
.....	244, 252, 3122, 3123, 3124, 3125
<code>\seq_gset_split:Nnn</code>	5400
<code>\seq_if_empty:NNTF</code>	178,
196, 264, 273, 299, 930, 946, 1193,	
1278, 1319, 2126, 2183, 4214, 7346,	
7720, 7736, 7968, 8026, 8051, 8280	
<code>\seq_if_empty_p:N</code>	286, 287, 2202
<code>\seq_if_exist:NNTF</code>	5399
<code>\seq_if_in:NnNTF</code> 63, 800, 2206, 2217,	
2570, 2685, 2804, 2812, 3104, 3297,	
3347, 4147, 4986, 5105, 5233, 6650	
<code>\seq_item:Nn</code>	265, 266,
274, 2136, 2158, 5467, 6471, 6477, 7397	
<code>\seq_map_break:</code>	960, 1092, 4125
<code>\seq_map_break:n</code>	3211,
3219, 3397, 3425, 3433, 4128, 4237	

6779, 6789, 6790, 6921, 6924, 6926,
 6931, 6959, 8271, 8273, 8280, 8281
 \l_tmpb_seq 1063,
 1064, 1065, 1224, 1226, 1228, 1229
 \seqmap 98, 151, 296, 5763, 5905
 \setbox 144, 1900, 1992, 7713, 7926, 8678,
 8684, 8700, 9281, 9341, 9389, 9440
 \setcounter 10301, 10304
 \setkeys ... 2233, 2249, 2266, 8727, 10523
 \setlength 7824, 7825, 7826, 8500, 8501,
 8505, 8558, 8559, 8560, 9212, 9213,
 9214, 9486, 9487, 9488, 9505, 9506,
 9507, 9533, 9534, 9535, 9698, 9699,
 9700, 10115, 10116, 10117, 10155,
 10156, 10157, 10232, 10233, 10234
 \setlicensing 62, 110
 \setmetatheory 143, 2588, 2655
 \setnotation 32, 93, 153, 6606
 \setsectionlevel 26,
 84, 139, 1676, 8372, 8374, 8436, 8438
 \setSGvar 63, 112, 8888, 8898
 \setslidelogo 62, 110
 \setsource 62, 110
 sexample (env.) 101, 157, 7443
 \sf 8745
 \sffamily 8754
 sfragment (env.) 84, 139, 1547
 sfragment commands:
 _sfragment_do_level:nn
 1563, 1567, 1571, 1572,
 1573, 1574, 1575, 1579, 1591, 8413
 _sfragment_end: 1558, 1587, 1604, 1626
 \shorttitle 1614, 1615
 \skipfragment 84, 139, 1660
 \slideframewidth . 8504, 8505, 8594, 8723
 \slideheight 8501
 slides 60, 108
 \slidewidth
 .. 8500, 8597, 8633, 8723, 8725, 8729
 \smacro 94, 95
 \small 8131
 \smallskip 8127, 9125, 9130,
 9218, 9223, 9298, 9358, 9406, 9458,
 10034, 10039, 10076, 10081, 10122,
 10127, 10163, 10168, 10195, 10321
 smodule (env.) 88, 142, 2302
 \smodule 3314
 \Sn 19, 91, 6056
 \sn 19, 23, 91, 152, 1463, 6017
 \Sn\Sns 152, 6017
 \Sns 19, 91, 6057
 \sns 19, 91, 152, 6017
 \solution 68
 solution (env.) 1, 9245
 \solution 9049, 9308, 9559, 9724
 solutions 76, 113, 118
 \solutionsfalse 8937, 9313
 \solutionstrue 8935, 9310
 \source 124
 sparagraph (env.) 101, 157, 7446
 \spfarg 159, 7707, 8115
 spfblock (env.) 158, 8072
 \spfblock 7704, 8078
 \spfby 158, 7706, 8111
 \spfescape 7703, 7783, 7786
 \spfjust 158, 7705, 8107
 \spfsketch 158, 7960
 spfsketchenv (env.) 158, 7959
 \spfsketchenvautorefname 7978
 \spfstep 158, 7698, 8082
 \spfstepautorefname 7834, 8070
 spfstepenv (env.) 158, 8069
 \spfstepenv 7994, 8087
 \spfstepenvautorefname 8070
 sproblem (env.) 9048
 \sproblemautorefname 9147
 sproof (env.) 103, 158, 7677
 \sproofautorefname 7799
 \sproofend 159, 7801, 7818, 8119
 \sqcap 8222
 \square 8120, 9605, 9636
 \sr 19, 23, 90, 152, 6010
 \sref 86, 87, 101, 1840, 8839
 \sreflabel 86, 88, 1811
 \srefsetin 86, 2024
 \srefsym 91, 2041
 \srefsymuri 91, 2043, 2046
 \startsolutions 114, 9309
 \stepcounter 1662, 1666, 1670,
 1671, 1672, 1673, 1674, 8414, 8609
 \TeX 14, 16, 83
 \stex 16, 23, 83
 stex commands:
 _l_stex_aB_args_seq 295,
 5701, 5718, 5774, 5786, 5791, 5825,
 5832, 5851, 5857, 5861, 5872, 5881,
 6901, 6922, 6931, 6932, 6949, 6959
 _stex_activate_module:N
 143, 2411, 2429, 2565, 2565, 2826, 2833
 _stex_activate_module:n 143, 2397,
 2400, 2565, 2566, 2569, 2971, 5156,
 7085, 7127, 7129, 7139, 7154, 7260
 _stex_activate_notations: 2579, 6530
 _stex_activate_symbols: . 2578, 4083
 _stex_add_definiens:nn
 157, 3319, 7472,
 7477, 7477, 7690, 7692, 7883, 7885

`\stex_add_morphism:nnnn` 145, 2973,
 3025, 3025, 3030, 3148, 7082, 7124
`\stex_add_notation:nnnnn`
 153, 3177, 6501, 6508, 6515, 6528, 6608
`\stex_add_symbol:nnnnnnN` .. 147, 4067
`\stex_add_symbol:nnnnnnnN`
 148, 3201,
 3206, 3867, 3935, 4067, 7053, 7383
`\l_stex_all_module_seq` 5155
`\l_stex_all_modules_seq`
 142, 2092, 2301, 2385,
 2445, 2570, 2572, 2689, 4130, 4253
`\l_stex_allow_semantic_bool`
 151, 4618, 4639,
 4640, 4642, 4709, 4979, 5011, 5036,
 5078, 5260, 5284, 5371, 5380, 5495,
 5596, 5602, 5620, 5638, 5671, 5794,
 5884, 5917, 5942, 6139, 6162, 7195
`\stex_annotate:nn`
 128, 129, 709, 714, 717, 1531, 1541,
 1853, 1966, 2114, 2115, 3249, 3251,
 3592, 4022, 4043, 4046, 4053, 4058,
 4060, 4400, 4403, 4410, 4414, 4417,
 4554, 4559, 4562, 4569, 4573, 4576,
 4712, 4898, 4907, 4912, 5131, 5652,
 5685, 5800, 5983, 6190, 6193, 6687,
 6694, 6700, 6701, 6710, 6721, 6731,
 7305, 7332, 7479, 7519, 7530, 7708,
 7759, 7762, 7765, 7900, 7915, 8049,
 8104, 8108, 8112, 8116, 8278, 8283,
 8623, 8814, 9090, 9144, 9190, 9489,
 9497, 9521, 9526, 9614, 9618, 9626,
 9752, 9756, 9764, 9858, 9893, 10237
`\stex_annotate_env` 2327, 9706
`\stex_annotate_force_break:n` ...
 . 129, 712, 1584, 1646, 3250, 3591,
 4040, 4058, 4397, 4415, 4552, 4574,
 4911, 5047, 5125, 5657, 5684, 5688,
 5700, 5805, 6687, 7305, 7333, 7480,
 7520, 7708, 7711, 7754, 7925, 8279,
 9090, 9094, 9190, 9197, 9864, 9897
`\stex_annotate_invisible:n` . 129,
 709, 710, 1656, 2335, 2621, 3075,
 3577, 4021, 5423, 6687, 7757, 8276
`\stex_annotate_invisible:nn`
 129, 709,
 1506, 1701, 1707, 1713, 2940, 2976,
 3013, 3590, 3664, 3704, 3714, 4378,
 4536, 5601, 5619, 5637, 5916, 7079,
 7121, 7257, 7505, 8972, 8995, 10245
`\stex_apply_patch_begin:n`
 525, 541, 3453, 3519
`\stex_apply_patch_end:n`
 531, 556, 3458, 3524
`\stex_archive_base:N`
 132, 884, 884, 1079
`\stex_archive_base:n`
 132, 884, 888, 1118, 1127, 1245, 1414
`\stex_archive_id:N` 132, 881,
 881, 903, 1072, 1076, 1079, 1080,
 1085, 1086, 1174, 1362, 2027, 2196
`\stex_archive_path:N`
 132, 891, 891, 1100, 1222, 2195
`\stex_archive_path:n`
 132, 891, 895, 1110
`\stex_args_end:` 5385,
 5409, 5412, 6146, 6170, 6181, 6185
`\stex_assign_do:n`
 3264, 3577, 3579, 3586, 3638
`\l_stex_assoc_args_count`
 3967, 3971, 4013, 4014
`\l_stex_brackets_dones_bool`
 6829, 6970, 6976, 6977, 6994
`\stex_check_term:nn`
 152, 2691, 3588, 6230, 6231, 6239,
 6245, 6249, 6253, 6258, 6678, 8136
`\stex_check_terms:` ... 152, 3866,
 3930, 4323, 4488, 6241, 6242, 6262
`\c_stex_check_terms_bool`
 36, 6217, 6220, 8138
`\stex_close_module:`
 ... 142, 2355, 2447, 2447, 2628, 8250
`\stex_comma_sep_uri:nn`
 7347, 7357, 7721, 7737
`\stex_css_link:n` 129, 712
`\stex_css_literal:n`
 129, 81, 712, 818, 1722, 8568
`\l_stex_current*` 149, 150
`\l_stex_current_archive`
 132, 133, 902, 903, 912, 923,
 1060, 1099, 1100, 1172, 1174, 1221,
 1222, 1360, 1362, 2191, 2195, 2196
`\l_stex_current_args_tl`
 .. 4662, 5384, 5385, 5388, 6754, 6769
`\l_stex_current_arity_str`
 4661, 4729, 4744,
 4746, 5356, 5505, 5506, 5541, 6775
`\l_stex_current_def_uri`
 3238, 3240, 3243, 3247, 3249, 3255,
 7368, 7390, 7391, 7397, 7398, 7465,
 7467, 7470, 7472, 7514, 7516, 7519
`\l_stex_current_display_tl`
 149, 4622,
 5263, 5279, 5282, 5668, 5788, 6032,
 6036, 6049, 6053, 6064, 6078, 6092,
 6116, 6126, 6164, 6374, 6789, 7202
`\l_stex_current_doc_uri` 7350

<code>\l_stex_current_document_uri</code> . . .	<code>\l_stex_current_symbol_arity_int</code>
. 125, 135, 144, 253, 254, 149, 150, 272
259, 1159, 1221, 1295, 1315, 1334,	<code>\l_stex_current_symbol_invoke_cs</code> 149
1338, 1343, 1345, 1452, 2423, 8141	<code>\l_stex_current_symbol_name_str</code> .
<code>\l_stex_current_domain_str</code> . . . 3732 6142, 6143, 6144, 6145
<code>\l_stex_current_domain_uri</code>	<code>\l_stex_current_symbol_return_tl</code>
. . . 3032, 3045, 3049, 3051, 3056, 149, 150, 272
3064, 3072, 3088, 3092, 3121, 3150,	<code>\l_stex_current_symbol_type_tl</code> . .
3175, 3470, 3495, 3537, 3561, 3681 149, 150, 272
<code>\g_stex_current_file</code>	<code>\l_stex_current_symbol_uri</code>
. . 125, 244, 252, 257, 1102, 1112, 149, 150,
1374, 2420, 2894, 9081, 9176, 9181	272, 294, 3795, 3808, 3809, 3814,
<code>\l_stex_current_full_tl</code> 149,	3826, 3862, 3877, 3886, 3891, 3894,
4621, 4630, 4669, 4674, 4689, 4702,	3900, 3925, 3946, 4620, 4621, 4623,
4714, 4842, 4925, 5051, 5083, 5127,	5666, 6140, 6141, 6142, 6995, 7000
5262, 5272, 5278, 5281, 5288, 5323,	<code>\l_stex_current_term_tl</code> 4644, 4665,
5324, 5335, 5348, 5349, 5378, 5459,	4918, 4924, 5122, 5285, 5567, 5583,
5486, 5654, 5667, 5787, 5837, 5885,	5597, 5603, 5618, 5621, 5636, 5639,
5886, 5985, 5988, 6031, 6048, 6063,	5669, 5795, 5796, 5797, 5887, 6149
6077, 6091, 6141, 6154, 6163, 6173,	<code>\stex_current_this:</code> . 4658, 7194, 7210
6174, 6176, 6373, 6709, 6720, 7197	<code>\l_stex_current_this_tl</code>
<code>\l_stex_current_language_str</code> 7019, 7022, 7196, 7204
. 125, 127, 254, 258, 453,	<code>\l_stex_current_type_tl</code>
458, 459, 1195, 1201, 1206, 1235, 4664, 4711, 4784, 4839,
2329, 2863, 2865, 2881, 9086, 9186	4847, 4850, 4854, 5156, 5157, 5232
<code>\l_stex_current_module_str</code>	<code>\stex_deactivate_macro:Nn</code>
. 2627, 3697, 3706, 3716 123, 67, 67, 2611, 2952,
<code>\l_stex_current_module_uri</code>	2959, 3007, 3308, 3309, 3310, 3311,
142, 1293, 1422, 2093, 2301, 2328,	3312, 3313, 3314, 3477, 3483, 3502,
2349, 2384, 2416, 2433, 2444, 2450,	3543, 3549, 3569, 3584, 3659, 3728,
2451, 2452, 2453, 2454, 2459, 2461,	3789, 3820, 3909, 6111, 6120, 6130,
2465, 2470, 2472, 2548, 2549, 2554,	6136, 6319, 7040, 7095, 7145, 7475,
2574, 2575, 2580, 2581, 2583, 2605,	7509, 7524, 7532, 7786, 7957, 8015,
2690, 3026, 4068, 4075, 4084, 4085,	8078, 8102, 8106, 8110, 8114, 8118,
4097, 4102, 6519, 6521, 6531, 6532,	8293, 9308, 9365, 9413, 9465, 9518,
6540, 7134, 7378, 7390, 7484, 8248	9557, 9558, 9559, 9560, 9561, 9562,
<code>\l_stex_current_ns_uri</code> 2315	9579, 9674, 9722, 9723, 9724, 9725,
<code>\l_stex_current_redo_tl</code>	9726, 9727, 9744, 9809, 9930, 10259
4645, 4650, 4657, 4667, 4668, 4671,	<code>\stex_debug:nn</code> 124, 103, 103,
4843, 4881, 4917, 5073, 5267, 5537	133, 137, 166, 173, 237, 472, 547,
<code>\l_stex_current_return_tl</code>	616, 847, 852, 880, 922, 931, 949,
. . 4663, 4679, 5325, 5337, 5357, 5542	996, 1033, 1067, 1075, 1096, 1166,
<code>\l_stex_current_section_level_-</code>	1237, 1315, 1353, 1366, 1381, 1390,
int 139, 1526, 1562,	1498, 1851, 1877, 1879, 1888, 1892,
1564, 1568, 1582, 1599, 1600, 1629,	1933, 1939, 1953, 1956, 1959, 1991,
1631, 1634, 1637, 1638, 1645, 1661,	2094, 2125, 2182, 2200, 2204, 2215,
1663, 1667, 1678, 1679, 1680, 1681,	2368, 2375, 2377, 2405, 2408, 2410,
1682, 1683, 1685, 1689, 1692, 1699,	2413, 2425, 2433, 2435, 2437, 2449,
1703, 1709, 1713, 1718, 8417, 8424	2497, 2549, 2571, 2580, 2585, 2589,
<code>\l_stex_current_section_level_-</code>	2591, 2692, 2745, 2766, 2770, 2788,
str 139, 1526, 1533, 1543, 1602, 8425	2805, 2813, 2895, 2910, 2912, 2938,
<code>\l_stex_current_symbol_args_tl</code> . .	3011, 3046, 3064, 3106, 3134, 3171,
. 149, 150, 272	3186, 3243, 3262, 3363, 3419, 3587,
	3606, 3609, 3613, 3633, 3637, 3662,

3682, 3698, 3702, 3712, 3733, 3735,
 3808, 4068, 4084, 4097, 4211, 4252,
 4254, 4273, 4290, 4292, 4329, 4457,
 4674, 4690, 4783, 5027, 5154, 5170,
 5293, 5312, 5324, 5326, 5329, 5335,
 5348, 5370, 5378, 5493, 5533, 5544,
 5714, 5729, 5731, 5738, 5741, 6243,
 6247, 6251, 6255, 6402, 6433, 6444,
 6516, 6531, 6539, 6541, 6544, 6680,
 6707, 6808, 6821, 6862, 6980, 7152,
 7234, 7237, 7241, 7391, 7395, 7398,
 7400, 7470, 7491, 7527, 7688, 7881
 \c_stex_debug_clist 30,
 42, 104, 107, 121, 125, 127, 131, 135
 \c_stex_default_metatheory
 2091, 2688, 8249
 \l_stex_default_notation
 ... 4704, 5338, 5343, 5363, 5821,
 6757, 6758, 6764, 6773, 6778, 6781
 _stex_definiens_impl:nn
 3320, 7457, 7462
 \stex_do_default_notation:
 ... 154, 4703, 5336, 5820, 6752, 6752
 \stex_do_default_notation_op: ...
 154, 5360, 6752, 6753, 6762
 _stex_do_deprecation:n 649, 649, 3949
 _stex_do_for_list: ... 157, 3318,
 7283, 7283, 7367, 7683, 7876, 7963
 _stex_do_id: 654, 654,
 1556, 1623, 7312, 7330, 7696, 7896,
 7909, 7920, 7964, 7998, 9112, 10312
 \stex_do_up_to_module:n
 142, 2543, 2543,
 2546, 2550, 4078, 6523, 7138, 7185
 \g_stex_document_kind_parameters_-
 tl 1480, 1486, 10471
 \g_stex_document_kind_str
 14, 1479, 1481, 1485, 10459
 \stex_document_uri:n 134
 \stex_document_uri_archive:N ...
 134, 1137, 1137, 1338
 \stex_document_uri_element:N ...
 135, 1149, 1149
 \stex_document_uri_from_archive_-
 file:Nn 135, 1164,
 1164, 1876, 1965, 2056, 2101, 8829
 \stex_document_uri_language:N ...
 135, 254, 1146, 1146, 1235
 \stex_document_uri_name:N
 135, 1143, 1143, 1343, 1345
 \stex_document_uri_path:N
 135, 1140, 1140, 1334
 \stex_document_uri_with_language:Nn
 135, 1152, 1152, 2422
 _stex_eat_exclamation_point: ...
 150, 3955, 5327, 5338, 5559, 5559, 5560
 _stex_end: 414, 422
 \stex_end: 7220, 7230
 \stex_every_module:n
 142, 2539, 2540,
 2612, 2960, 3008, 3478, 3484, 3503,
 3544, 3550, 3570, 3790, 3821, 3910,
 6320, 7041, 7096, 7147, 7191, 8294
 \g_stex_every_module_tl
 2366, 2539, 2541, 8146
 \l_stex_every_symbol_tl 4599, 4601,
 4602, 4608, 4609, 4612, 4641, 4667
 \stex_execute_in_module:n
 142, 2390, 2547, 2547,
 2552, 2970, 3893, 7084, 7113, 7126
 \l_stex_feature_name_str .. 3067,
 3149, 3185, 3186, 3192, 3375, 3674
 \stex_file_in_smsmode:Nn
 144, 2426, 2678, 2681, 2926
 \stex_file_resolve:Nn 124, 147, 153,
 169, 176, 233, 243, 251, 867, 869, 1373
 \stex_file_set:Nn
 124, 141, 141, 146, 160, 171, 860, 1165
 \stex_file_split_off_ext:NN ...
 124, 208, 208
 \stex_file_split_off_lang:NN ...
 124, 215, 215, 2420, 9081, 9176, 9181
 \stex_file_use:N ... 124, 166, 173,
 205, 205, 237, 840, 860, 862, 870,
 874, 962, 996, 997, 999, 1012, 1065,
 1102, 1112, 1166, 1181, 1374, 1378,
 1799, 2203, 2214, 2425, 2427, 2894
 \l_stex_fors_seq
 157, 3226, 3229, 7284, 7287, 7346,
 7347, 7377, 7396, 7397, 7412, 7431,
 7687, 7720, 7721, 7736, 7737, 7880,
 7968, 7969, 8026, 8027, 8051, 8052
 \stex_get_env:Nn
 124, 86, 87, 98, 119, 229,
 231, 239, 241, 568, 574, 667, 838, 6214
 \stex_get_in_morphism:n 145, 3228,
 3237, 3362, 3362, 3575, 3631, 3654
 \stex_get_mathstructure:n
 156, 3048, 7069, 7103, 7211, 7211, 7240
 \stex_get_mathstructure_maybe:nTF
 7212, 7216, 7233
 \l_stex_get_structure_module_uri
 156, 3049,
 7111, 7114, 7217, 7220, 7234, 7241
 \l_stex_get_svariable_str 6167
 \stex_get_symbol:n
 148, 2042, 4163, 4165,

4586, 6013, 6030, 6047, 6161, 6297,	\c_stex_home_file .. 125, 238, 840, 860
6623, 7286, 7464, 7513, 8957, 8980	\stex_html_literal:n
\stex_get_symbol:nTF .. 148, 1462,	... 27, 35, 764, 772, 1689, 1692, 2068
4163, 4166, 4171, 4191, 4192, 7218	\stex_html_node:nnn
\l_stex_get_symbol_args_tl 128, 129, 709, 1484, 1583
... 148, 3400, 3939,	\stex_if_check_terms: 6210, 6221, 6225
3959, 3961, 3962, 4200, 4240, 4301,	\stex_if_check_terms:TF
4356, 4368, 4461, 4514, 4527, 4590,	... 152, 3811, 4323, 4332, 4488,
5388, 6568, 6570, 6707, 6769, 7386	4491, 6208, 6230, 6241, 6299, 6338
\l_stex_get_symbol_arity_int ...	\stex_if_check_terms_p: ... 152, 6208
... 148, 3399, 3882, 3938, 3960,	\stex_if_do_html:
3970, 3976, 3980, 3984, 3988, 3992,	\stex_if_do_html:TF 128, 695, 704,
3996, 4000, 4004, 4008, 4011, 4012,	1505, 1529, 1539, 1687, 2347, 2357,
4013, 4014, 4051, 4163, 4199, 4239,	2614, 2629, 2939, 2975, 3012, 3069,
4300, 4328, 4355, 4367, 4408, 4460,	3127, 3244, 3576, 3589, 3663, 3813,
4513, 4526, 4567, 4589, 6363, 6397,	3878, 3889, 3932, 4326, 4334, 4377,
6428, 6435, 6455, 6504, 6609, 7385	4487, 4493, 5682, 6301, 7078, 7120,
\l_stex_get_symbol_def_tl	7256, 7389, 7478, 7504, 7684, 7689,
... 148, 3276,	7695, 7709, 7794, 7806, 7814, 7877,
3401, 4201, 4241, 4302, 4462, 4591	7882, 7892, 7913, 7916, 7923, 7934,
\l_stex_get_symbol_invoke_cs ...	7954, 8022, 8046, 8756, 9079, 9111,
... 148, 3404,	9118, 9119, 9122, 9142, 9179, 9207,
4204, 4244, 4305, 4465, 4594, 7219	9209, 9263, 9272, 9285, 9290, 9326,
\l_stex_get_symbol_macro_str . 3398	9334, 9345, 9350, 9374, 9382, 9393,
\l_stex_get_symbol_name_str .. 3269	9398, 9425, 9434, 9445, 9450, 9484,
\l_stex_get_symbol_return_tl ...	9539, 9564, 9566, 9613, 9704, 9729,
... 148, 3403, 4203,	9731, 9751, 9877, 9883, 10236, 10306
4243, 4304, 4330, 4464, 4479, 4593	\stex_if_do_html_p:
\l_stex_get_symbol_type_tl	\stex_if_file_absolute:N ... 263, 272
... 148, 3402, 4202, 4242, 4303,	\stex_if_file_absolute:NTF
4463, 4592, 7070, 7105, 7220, 7250	... 125, 262, 868
\l_stex_get_symbol_uri . 148, 1463,	\stex_if_file_absolute_p:N . 125, 262
2043, 3230, 3238, 3261, 3268, 3364,	\stex_if_file_starts_with:NN 125, 279
3369, 3373, 3389, 3410, 3587, 3590,	\stex_if_file_starts_with:NNTF ..
3598, 3662, 3665, 3672, 3674, 3809,	... 125, 279, 1062
3886, 4172, 4188, 4198, 4238, 4299,	\stex_if_html_backend
4588, 6031, 6033, 6048, 6050, 6140,	\stex_if_html_backend:TF ... 128,
6163, 6165, 6303, 6307, 6325, 6326,	179, 2, 529, 685, 688, 712, 725, 739,
6502, 6613, 6616, 6624, 6625, 6626,	1482, 1578, 1642, 1655, 1697, 1963,
6627, 6632, 6636, 7288, 7465, 7514,	2061, 2112, 2325, 4647, 4678, 5264,
8958, 8969, 8973, 8981, 8992, 8996	5563, 5579, 5593, 5613, 5631, 5650,
\stex_get_var:n	5680, 5982, 6138, 6159, 6188, 6209,
4432, 6061, 6073, 6087, 6335, 7503	7303, 7316, 8317, 8357, 8487, 8567,
\l_stex_get_variable_str	8622, 8634, 8642, 8654, 8664, 8813,
... 4433, 4435, 4459, 6063, 6064,	8824, 8867, 9603, 9746, 9849, 9886
6075, 6077, 6078, 6089, 6091, 6092,	\stex_if_html_backend_p: ... 128, 685
6168, 6169, 6336, 6341, 6344, 7505	\stex_if_in_module:
\stex_has_definiens:N	\stex_if_in_module:TF
\stex_has_definiens:n	143, 1292, 2360, 2547, 2553, 6607, 7483
\stex_has_definiens:NTF ... 147, 4106	\stex_if_in_module_p: ... 143, 2553
\stex_has_definiens:nTF	\stex_if_module_exists:N
... 147, 4106, 4107	\stex_if_module_exists:n
\stex_has_definiens_p:N ... 147, 4106	\stex_if_module_exists:NTF
\stex_has_definiens_p:n ... 147, 4106	... 143, 1350, 2376,

2409, 2436, 2557, 2821, 2830, 2927
 \stex_if_module_exists:nTF
 143, 2557, 7071, 7106, 7251
 \stex_if_module_exists_p:N 143, 2557
 \stex_if_module_exists_p:n 143, 2557
 \stex_if_smsmode: 2675
 \stex_if_smsmode:TF
 . 144, 655, 1496, 1812, 1831, 2348,
 2356, 2673, 2893, 2943, 2979, 3016,
 3256, 3445, 3460, 3468, 3493, 3511,
 3527, 3535, 3559, 3785, 3816, 3898,
 6310, 7232, 7296, 7315, 7331, 7337,
 7411, 7430, 7444, 7482, 9207, 9209
 \stex_if_smsmode_p: 144, 2673
 \stex_ignore_spaces_and_pars: ...
 123, 45, 45, 48,
 2604, 2606, 7337, 9135, 10061, 10103
 \l_stex_import_uri 2590,
 2591, 2593, 2932, 2933, 2941, 2945,
 2946, 2964, 2965, 2971, 2974, 2977,
 2982, 2983, 2990, 2993, 3003, 3004,
 3014, 3018, 3019, 3052, 3054, 3056
 \stex_in_archive:nn
 .. 132, 898, 898, 1875, 1940, 1964,
 2031, 2055, 2100, 2123, 2151, 2180,
 2224, 2240, 2255, 2267, 2852, 8828
 \g_stex_in_comp_bool
 5676, 5976, 5981, 5992
 \stex_in_invisible_html_bool ...
 4041, 4398, 5562, 5595
 \l_stex_in_meta_bool
 2387, 2396, 2399, 2401
 \l_stex_in_this_bool
 5079, 5565, 5575,
 5581, 5591, 5615, 5629, 5633, 5647,
 5683, 5697, 5943, 7192, 7203, 7205
 \l_stex_inpararray_bool 6982
 \stex_input_with_hooks:Nn .. 125,
245, 245, 248, 2075, 2095, 2103, 2106
 \stex_invoke_sequence:
 ... 150, 4518, 4531, 4682, 4682, 5756
 \stex_invoke_structure:
 150, 4771, 4771, 7055, 7219
 \stex_invoke_symbol:
149, 3943, 4360, 4372, 4673, 4673, 7387
 \stex_invoke_symbol:NnnnnnN ...
 149, 4587, 4617, 4635, 4638
 \stex_invoke_symbol:nnnnnnN ...
 149, 4101, 4179, 4196, 4197,
4617, 4617, 4636, 5181, 5195, 7169
 \stex_invoke_text_symbol:
 149, 3874, 4677, 4677
 \stex_invoke_variable:nnnnnn ...
 150, 5259

_stex_invoke_variable:nnnnnnN ..
 4365, 4524, 5259, 5754
 _stex_is_sequentialized:n
 5709, 5792, 5865, 5882
 \stex_iterate_break:
 148, 4120, 4121, 4124
 \stex_iterate_break:n
 148, 3339, 3703,
 3713, 3731, 4120, 4121, 4127, 4298
 \stex_iterate_morphisms:nn
 145, 3322, 3322, 3681, 3697
 \stex_iterate_notations:nn
 154, 3175, 5232, 6640, 6640
 \stex_iterate_symbols:n
 148, 4120, 4120, 4291
 \stex_iterate_symbols:nn
 ... 148, 3092, 3734, 4137, 4137, 5157
 \l_stex_key_* 147
 \l_stex_key_answerclass_str
 9248, 9252, 9266, 9300, 9301
 \l_stex_key_archive_str 379,
 382, 1875, 1928, 1933, 1939, 1940, 1964
 \l_stex_key_args_str
 3745, 3750, 3759, 3799,
 3830, 3963, 3972, 3977, 3981, 3985,
 3989, 3993, 3997, 4001, 4005, 4009,
 4024, 4380, 4474, 4475, 4538, 7268
 \l_stex_key_argtypes_clist
 3764, 3768, 4057, 4059, 4413, 4416,
 4572, 4575, 6256, 6257, 6258, 7272
 \l_stex_key_assoc_str 3748, 3753,
 4031, 4032, 4384, 4385, 4542, 4543
 \l_stex_key_autogradable_bool ...
 9022, 9028, 9033, 9087, 9187
 \l_stex_key_continues_tl . 7592, 7602
 \l_stex_key_def_tl 3761, 3771, 3798,
 3829, 3844, 3848, 3871, 3940, 4045,
 4046, 4357, 4369, 4402, 4403, 4515,
 4528, 4561, 4562, 6248, 6249, 7270
 \stex_key_def_tl 152
 \l_stex_key_deprecate_str
 405, 407, 650, 651
 \l_stex_key_due_tl
 10282, 10286, 10326, 10327
 \l_stex_key_fallback_tl
 1803, 1806, 1857, 1889, 1957
 \l_stex_key_feedback_tl
 9470, 9473, 9496, 9499,
 9512, 9513, 9588, 9593, 9625, 9627,
 9661, 9662, 9763, 9765, 9795, 9796,
 10227, 10244, 10247, 10254, 10255
 \l_stex_key_file_str
 380, 383, 1843, 1876,
 1883, 1929, 1933, 1939, 1941, 1965

<code>\l_stex_key_for_clist</code>	6377, 6506, 6598, 6599, 6602, 6611,
157, 3227, 7267, 7275, 7285, 7536, 7541	6615, 6617, 6627, 6629, 6718, 6722
<code>\l_stex_key_for_str</code>	7302
<code>\l_stex_key_from_tl</code>	7594, 7604
<code>\l_stex_key_Ftext_tl</code>	
.. 9591, 9597, 9622, 9659, 9760, 9793	6036, 6053, 6075, 6089, 6116, 6126
<code>\l_stex_key_functions_tl</code> .	7593, 7603
<code>\l_stex_key_given_tl</code>	
..... 10281, 10285, 10323, 10324	1801, 1863, 1868, 5996, 6000,
<code>\l_stex_key_hide_bool</code> 7591, 7599,	6036, 6053, 6075, 6089, 6116, 6126
7713, 7723, 7739, 7806, 7926, 7954	<code>\l_stex_key_prec_str</code>
<code>\l_stex_key_id_str</code> .	3885,
387, 389, 656,	6268, 6274, 6405, 6408, 6411, 6417,
657, 7349, 7350, 7405, 7407, 7893,	6420, 6423, 6426, 6432, 6444, 6445
7906, 7995, 9257, 9259, 9265, 9320,	<code>\l_stex_key_proofend_tl</code>
9322, 9328, 9368, 9370, 9376, 9418, 7590, 7596, 8126, 8127
9420, 9427, 9478, 9480, 9490, 9508,	<code>\l_stex_key_pts_str</code>
10051, 10093, 10140, 10181, 10201,	9469, 9472, 9491, 9492, 9509, 9510,
10213, 10215, 10226, 10235, 10238	10228, 10239, 10240, 10251, 10252
<code>\l_stex_key_intent_args_clist</code> ...	<code>\l_stex_key_pts_tl</code>
..... 6271, 6277, 6469, 6478, 6480	9024,
<code>\l_stex_key_intent_str</code>	9030, 9088, 9097, 9101, 9188, 9199,
..... 3884, 6270, 6276, 6394, 6472	9202, 9218, 10122, 10145, 10163, 10186
<code>\l_stex_key_just_tl</code>	<code>\l_stex_key_reorder_str</code> 3747, 3751,
..... 7539, 7544, 7755, 7761, 7762	4034, 4035, 4390, 4391, 4548, 4549
<code>\l_stex_key_macroname_str</code>	<code>\l_stex_key_return_tl</code>
.. 7266, 7276, 7362, 7364, 7380, 7384	3762, 3767, 3942, 4048, 4051, 4330,
<code>\l_stex_key_method_tl</code>	4359, 4371, 4405, 4408, 4479, 4517,
..... 7538, 7543, 7755, 7764, 7765	4530, 4564, 4567, 6252, 6253, 7271
<code>\l_stex_key_mhrepos_str</code>	<code>\stex_key_return_tl</code>
.. 9027, 9035, 9231, 9233, 9241, 10339	152
<code>\l_stex_key_min_tl</code>	<code>\l_stex_key_role_str</code>
9025, 3746, 3754, 3861, 4037,
9031, 9102, 9205, 9223, 10127, 10168	4038, 4387, 4388, 4545, 4546, 7381
<code>\l_stex_key_name_str</code>	<code>\l_stex_key_root_tl</code>
3758, 5998, 6002, 6006, 6008
3766, 3796, 3827, 3842, 3846, 3856,	<code>\l_stex_key_short_tl</code>
3857, 3859, 3863, 3869, 3901, 3922,	.. 1548, 1550, 1593, 1596, 1613, 1615
3923, 3926, 3937, 3949, 4023, 4318,	<code>\l_stex_key_showname_bool</code> .
4319, 4329, 4336, 4339, 4352, 4354,	7547,
4366, 4379, 4471, 4472, 4495, 4498,	7552, 7556, 7562, 7567, 7584, 7846
4510, 4512, 4525, 4537, 6336, 6593,	<code>\l_stex_key_sig_str</code> ..
6594, 6595, 6599, 6600, 6601, 7265,	2305, 2320,
7274, 7301, 7363, 7364, 7370, 7378,	2330, 2364, 2405, 2423, 2425, 2427
7384, 7390, 7550, 7554, 7561, 7564,	<code>\l_stex_key_style_clist</code>
7585, 7743, 7744, 7844, 7851, 7856,	399, 401, 504, 508, 542, 546, 7352,
7981, 7983, 7989, 7990, 8031, 8032,	7353, 7447, 9544, 9545, 9549, 9552,
8056, 8057, 9026, 9032, 9080, 9082,	9677, 9684, 9708, 9709, 9714, 9717
9085, 9175, 9177, 9180, 9182, 9185	<code>\l_stex_key_T_bool</code>
<code>\l_stex_key_number_tl</code> 9589, 9594, 9595, 9615, 9619,
..... 10280, 10284, 10300, 10301	9642, 9656, 9753, 9757, 9776, 9790
<code>\l_stex_key_numname_str</code>	<code>\l_stex_key_term_tl</code>
7551, 7555, 7560, 7569, 7731, 7742, 7537, 7542, 7755, 7758, 7759
7845, 7850, 7855, 8019, 8030, 8055	<code>\l_stex_key_testspace_dim</code>
<code>\l_stex_key_op_tl</code> 9246, 9249, 9251,
... 3883, 4481, 4483, 6269, 6275,	9280, 9834, 9836, 9892, 9921, 9924
6356, 6357, 6358, 6365, 6366, 6371,	<code>\l_stex_key_title_str</code>
	7277
	<code>\l_stex_key_title_tl</code>
 393, 395, 1926, 1933,
	1979, 1980, 2341, 7300, 7306, 7307,
	9090, 9091, 9092, 9096, 9190, 9191,

9192, 9194, 9195, 9299, 9300, 9359,
 9360, 9407, 9408, 9459, 9460, 10051,
 10093, 10140, 10181, 10196, 10197,
 10201, 10307, 10308, 10318, 10319
 \l_stex_key_Ttext_tl
 .. 9590, 9596, 9620, 9657, 9758, 9791
 \l_stex_key_type_tl
 3760, 3770, 3797, 3828, 3843, 3847,
 4042, 4043, 4358, 4370, 4399, 4400,
 4558, 4559, 6244, 6245, 7269, 7278
 \stex_key_type_tl 152
 \l_stex_key_variant_str
 3832, 4341, 4500,
 6267, 6273, 6280, 6282, 6324, 6346,
 6392, 6503, 6593, 6599, 6689, 6722
 \l_stex_key_wikidata_str
 3763, 3769, 4028, 4029
 \l_stex_keys_cls_id 6, 13, 36, 43, 44,
 46, 81, 743, 750, 773, 780, 781, 783, 818
 \l_stex_keys_counter_id
 7, 10, 28, 37, 61, 62, 63, 64,
 744, 747, 765, 774, 798, 799, 800, 801
 \l_stex_keys_counter_parent
 8, 11, 29, 48, 50, 51, 52, 53,
 54, 55, 56, 58, 745, 748, 766, 785,
 787, 788, 789, 790, 791, 792, 793, 795
 \stex_keys_define:nnnn 126,
 5, 359, 359, 378, 386, 392, 398,
 404, 410, 742, 1547, 1800, 1810,
 2304, 3744, 3757, 3841, 5995, 6005,
 6097, 6102, 6266, 6287, 6289, 7018,
 7264, 7535, 7549, 7589, 7607, 7980,
 8514, 9023, 9230, 9247, 9316, 9468,
 9587, 9832, 10279, 10383, 10434, 10447
 \stex_keys_set:nn
 126, 17, 374, 374, 754, 1554, 1609,
 1841, 1943, 2340, 3778, 3805, 3853,
 3854, 4316, 4469, 6012, 6029, 6046,
 6060, 6072, 6086, 6108, 6114, 6124,
 6296, 6334, 7043, 7293, 7328, 7682,
 7843, 7875, 7962, 7988, 8608, 9071,
 9075, 9165, 9240, 9256, 9279, 9319,
 9367, 9417, 9477, 9530, 9611, 9653,
 9695, 9749, 9787, 9888, 9905, 10212,
 10294, 10298, 10338, 10396, 10458
 \stex_kpsewhich:Nn 123, 77, 77, 89, 99
 \c_stex_language_abbrevs_prop ...
 127, 426
 \c_stex_languages_clist . 31, 454, 457
 \c_stex_languages_prop
 127, 222, 426, 465, 1203, 1208
 \g_stex_last_feature_str 2627
 \l_stex_macroname_str
 147, 2617, 2618, 3775, 3780, 3782,
 3806, 3855, 3868, 3936, 4025, 4026,
 4317, 4353, 4364, 4381, 4382, 4470,
 4511, 4523, 4539, 4540, 7056, 7380
 \c_stex_main_archive 133, 1060, 2027
 \c_stex_main_document_uri
 135, 1221, 1878
 \c_stex_main_file
 124, 228, 244, 1224, 1231
 _stex_map_args:N
 153, 4052, 4409, 4568, 5390,
 6465, 6491, 6567, 6567, 6714, 6771
 _stex_map_notation_args:N
 153, 6567, 6580, 6734
 \c_stex_mathhub<id>_archive .. 132
 \c_stex_mathhub_file 962
 \c_stex_mathhub_files
 132, 837, 930, 946, 952, 958, 1060
 \c_stex_mathhub_main_archive ...
 1070, 1071
 _stex_maybe_brackets:nn
 154, 4691,
 5350, 5361, 6811, 6827, 6975, 6975
 \c_stex_metadata_bool 37, 8967, 8990
 \stex_metagroup_do_in:n
 125, 126, 327,
 327, 331, 348, 2544, 3245, 3596, 3670
 \stex_metagroup_new:
 126, 323, 324, 2373, 2406, 2432, 3079
 \l_stex_metatheory_uri 2091, 2302,
 2309, 2311, 2331, 2332, 2367, 2368,
 2391, 2593, 2595, 2688, 8248, 8249
 _stex_module_code_macro:N
 2295, 2379, 2439, 2451, 2470, 2504,
 2548, 2558, 2575, 2580, 2581, 2605
 _stex_module_code_macro:n
 2298, 2562, 7153
 \stex_module_setup:n
 142, 2345, 2359, 2359, 2623
 _stex_module_setup_top_nosig:n .
 2365, 2372, 8145
 \stex_module_setup_top_nosig:n ..
 142, 2372
 \stex_module_uri:n 135
 \stex_module_uri_archive:N
 135, 1251, 1251, 2845
 \stex_module_uri_as_qm:n
 136, 1285, 1285, 4254, 4255
 \stex_module_uri_name:N 136, 1260,
 1260, 2328, 2847, 2946, 2983, 3019
 \stex_module_uri_name:n
 136, 1260, 1264, 4262, 4265
 \stex_module_uri_path:N
 135, 1254, 1254, 2846

<code>\stex_module_uri_path:n</code>	<code>_stex_notation_check:</code> . 154, 3811,
..... 135, <u>1254</u> , 1257, 4274	4332, 4491, 6299, 6338, <u>6677</u> , 6677
<code>\stex_module_uri_split_name:NNN</code> .	<code>\l_stex_notation_cs</code>
..... 136, <u>1267</u> , 1267, 7111 154, 4690, 4693, 4704,
<code>\stex_module_uri_split_name:NNn</code> .	4731, 4735, 4743, 4750, 4765, 4926,
..... 136, <u>1267</u> , 1271	5327, 5331, 5352, 5818, 6667, 6673
<code>\l_stex_morphism_assigns_seq</code> 3086,	<code>_stex_notation_do_html:n</code>
3124, 3141, 3197, 3246, 3297, 3597 154, 3814,
<code>\l_stex_morphism_morphisms_seq</code> ..	3891, 4336, 4495, 6303, <u>6686</u> , 6686
..... 3085, 3114, 3125	<code>\l_stex_notation_downprec</code>
<code>\l_stex_morphism_renames_seq</code> 3084, 5670, 6967, 6974, 6980, 6981
3123, 3138, 3152, 3165, 3370, 3671	<code>\l_stex_notation_macrocode_cs</code> ...
<code>\l_stex_morphism_symbols_prop</code> 153, 3834, 4343, 4502, 6328,
..... 3271, 3737	6348, 6390, 6402, 6505, 6593, 6596,
<code>\l_stex_morphism_symbols_seq</code> ...	6610, 6614, 6625, 6680, 6681, 6712
..... 3083, 3093, 3122,	<code>_stex_notation_make_args:</code>
3135, 3144, 3286, 3366, 3427, 3435 154, 3835,
<code>_stex_morphisms_macro:N</code> .. 2283,	4344, 6329, 6349, 6682, <u>6733</u> , 6733
2380, 2440, 2452, 2461, 2498, 3026	<code>\stex_notation_parse:n</code>
<code>_stex_morphisms_macro:n</code> 153, 3810, 3887, 4331,
.. 2286, 3103, 3349, 4149, 6655, 7156	4482, 4485, 6298, 6337, <u>6355</u> , 6355
<code>\stex_new_document_element_uri:n</code>	<code>_stex_notation_set_default:n</code> ...
..... 135, <u>1158</u> , 1158, 1813	... 153, 6306, 6341, <u>6606</u> , 6606, 6631
<code>\stex_new_module_uri:n</code>	<code>_stex_notations_macro:N</code>
..... 136, <u>1291</u> , 1291,	... 2289, 2382, 2442, 2454, 2472,
2374, 2407, 2434, 7050, 7114, 7139	2505, 2531, 6521, 6531, 6532, 6540
<code>\stex_new_statement:nnn</code>	<code>_stex_notations_macro:n</code> . 2292, 6652
157, <u>7291</u> , 7291, 7423, 7428, 7443, 7446	<code>\stex_par:</code>
<code>\stex_new_stylable_cmd:nnnn</code> 127,	.. 9678, 9679, 9682, 9685, 9686, 9689
482, 482, 2931, 2954, 3002, 3489,	<code>\stex_persist:n</code>
3555, 3574, 3653, 3679, 3777, 3804,	127,
3852, 4315, 4468, 6295, 6333, 7960	184, <u>586</u> , 614, 615, 726, 727, 729,
<code>\stex_new_stylable_env:nnnnnnn</code> ..	732, 735, 736, 1035, 1078, 1085, 2458
.. 127, <u>516</u> , 516, 2339, 3442, 3451,	<code>\c_stex_persist_force_bool</code> .. 35, 581
3507, 3517, 7031, 7066, 7100, 7292,	<code>\c_stex_persist_mode_bool</code>
7788, 7810, 7874, 9065, 9164, 9271, 33, 571, 582, 600
9333, 9381, 9433, 9529, 9694, 10291	<code>_stex_persist_read_now:</code>
<code>\stex_new_symbol_uri:n</code> 599, 1084, 8256
136, <u>1421</u> , 1421, 3178, 3863, 3926, 7199	<code>\c_stex_persist_write_mode_bool</code> .
<code>\stex_new_symbol_uri:nn</code> 34, 577, 601, 607, 724, 1083, 2448
..... 136, <u>1424</u> , 1424, 3094,	<code>\stex_pseudogroup:nn</code> 125, 126, 250,
4102, 4238, 4299, 5164, 7169, 7390	<u>303</u> , 303, 700, 899, 2573, 5379, 7005
<code>_stex_next_symbol:n</code>	<code>\stex_pseudogroup_restore:N</code>
149, <u>4596</u> , 4598, 4616, 4990, 5120, 6134 126, 258,
<code>\c_stex_no_archive</code>	259, <u>306</u> , 306, 912, 2583, 7010, 7011
133, <u>1051</u>	<code>\stex_pseudogroup_with:nn</code>
<code>\c_stex_no_archive_str</code> 126, <u>313</u> , 313, 4121, 4139,
..... 133, <u>1051</u> , 1177, 1377, 2848	4196, 5979, 6642, 6899, 6915, 6940
<code>\c_stex_no_archive_uri</code> ... 1053, 1059	<code>\c_stex_pwd_file</code>
<code>\c_stex_no_frontmatter_bool</code> 39, 1743 124, 228, 870, 1062, 1063, 1799
<code>_stex_notation_add:</code>	<code>\stex_reactivate_macro:N</code>
... 153, 3812, 3888, 6300, <u>6500</u> , 6500 123, <u>67</u> , 73, 2612, 2953,
<code>\l_stex_notation_args_tl</code>	2960, 2999, 3008, 3315, 3316, 3317,
..... 6362, 6395,	3479, 3485, 3504, 3545, 3551, 3571,
6464, 6470, 6476, 6581, 6583, 6680	3790, 3821, 3910, 6320, 7041, 7097,

7148, 7416, 7417, 7418, 7419, 7420,
 7421, 7435, 7436, 7437, 7438, 7439,
 7440, 7441, 7697, 7698, 7699, 7700,
 7701, 7702, 7703, 7704, 7705, 7706,
 7707, 8294, 9049, 9050, 9051, 9052,
 9053, 9054, 9055, 9443, 9563, 9728
 \stex_ref_new_doc_target:n
 657, 1811, 1811, 1829
 _stex_ref_new_id:n 7406
 \stex_ref_new_sym_target:n
 1830, 1830, 6173, 7413, 7432
 \stex_ref_new_symbol:n
 2038, 2501, 3876, 3945
 \l_stex_relative_import_bool ...
 1312, 1314, 1352, 2991
 _stex_renamedec1_do:nn
 3651, 3655, 3661
 \stex_require_module:N . 145, 2595,
2820, 2820, 2933, 2965, 3004, 3054
 \stex_require_module_noerr:N ...
 145, 2820, 2829, 2840
 \stex_require_module_noerr:n ...
 145, 2820, 2838, 2993
 _stex_return_args:nn
 3954, 4052, 4409, 4568
 \l_stex_return_notation_tl
 4656, 4826, 4834, 4998,
 4999, 5087, 5113, 5114, 5524, 5530
 \stex_set_language:n 2318
 _stex_set_meta_archive:.. 928, 8140
 \stex_sms_allow:N 144, 2634,
 2636, 2649, 2650, 2651, 2652, 2653
 \stex_sms_allow_env:n
144, 2644, 2646, 2654, 3481, 3487,
 3547, 3553, 7039, 7094, 7144, 7320
 \stex_sms_allow_escape:N
144, 2639, 2641, 2655, 2656, 2961,
 3506, 3573, 3583, 3658, 3727, 3791,
 3822, 3911, 6321, 7246, 7340, 7476
 \stex_sms_allow_import:Nn
 144, 2657, 2660, 2998
 \stex_sms_allow_import_* 144
 \stex_sms_allow_import_env:nn ...
 144, 2664, 2666, 2671
 \g_stex_sms_import_code
 144, 2672, 2684, 2701, 2992
 \stex_smsmode_do: 144, 2353, 2597,
 2608, 2726, 2728, 2731, 2731, 2956,
 3473, 3499, 3540, 3566, 3581, 3656,
 3786, 3817, 3906, 6316, 7033, 7088,
 7132, 7244, 7313, 7337, 7338, 7459
 \stex_source_path:n
 133, 1098, 1098, 1107, 1883,
 1941, 2032, 2065, 2075, 2094, 2095,
 2103, 2106, 2231, 2247, 2264, 2854
 \stex_source_path:nn
 133, 1098, 1105, 2226, 2242, 2257
 _stex_sref_do_aux:n 8137
 \stex_str_if_ends_with:nn .. 123, 62
 \stex_str_if_ends_with:nnTF
 123, 62,
 588, 591, 3688, 3711, 4255, 4262, 4274
 \stex_str_if_ends_with_p:nn
 123, 62, 4227, 4296
 \stex_str_if_starts_with:nn 123, 57
 \stex_str_if_starts_with:nnTF ...
 . 123, 57, 413, 1029, 1911, 4804, 4808
 \stex_str_if_starts_with_p:nn ...
 123, 57
 \l_stex_struct_this_tl
 4790, 4992, 4993, 4996, 5074, 5077,
 5086, 5107, 5108, 5111, 5125, 5944
 \stex_structural_feature_-
 module:nn ... 143, 2613, 2613, 7058
 \stex_structural_feature_module_-
 end: 143, 2613, 2626, 7035, 7090, 7137
 \stex_structural_feature_-
 morphism:nnnnn 145,
3032, 3039, 3443, 3491, 3508, 3557
 \stex_structural_feature_-
 morphism_check_total:
 3285, 3510, 3526, 3564
 \stex_structural_feature_-
 morphism_end: . 145, 3032, 3120,
 3448, 3463, 3498, 3514, 3530, 3565
 \stex_structural_feature_-
 morphism_with_macros:nnnnn ..
145, 3032, 3035, 3455, 3490, 3521, 3556
 \stex_style_apply:
 127, 3, 482, 487, 524,
 530, 740, 2351, 2356, 2948, 2985,
 3021, 3446, 3452, 3457, 3461, 3471,
 3496, 3512, 3518, 3523, 3528, 3538,
 3562, 3801, 3838, 3903, 4347, 4504,
 6313, 6352, 7310, 7316, 7695, 7797,
 7814, 7892, 7913, 7916, 7939, 7973,
 9111, 9118, 9207, 9209, 9277, 9289,
 9339, 9349, 9387, 9397, 9430, 9449,
 9564, 9568, 9729, 9733, 10311, 10314
 \stex_suppress_html:n
 128, 699, 699, 2714, 6385
 _stex_symbol_macro:N
 2277, 2381, 2441, 2453, 2465,
 2499, 2500, 4075, 4084, 4085, 7134
 _stex_symbol_macro:n
 2280, 4111, 4131, 4152, 4256,
 4269, 4275, 7161, 7167, 7492, 7494

<code>\stex_symbol_uri:n</code>	136	<code>\stex_uri_resolve:Nn</code>	2315
<code>\stex_symbol_uri_archive:N</code>		<code>\stex_uri_use:N</code>	7350
.....	136, 1427, 1427	<code>\stex_use_archive_uri:n</code>	
<code>\stex_symbol_uri_module:N</code>	134, 1117, 1117
.....	137, 1433, 1433	<code>\stex_use_document_uri:N</code>	
<code>\stex_symbol_uri_module:n</code>		134, 1120, 1120, 1237, 1315, 1452,	
137, 1433, 1436, 4111, 7484, 7492, 7494		1814, 1815, 1817, 1822, 1877, 1882,	
<code>\stex_symbol_uri_name:N</code>		1966, 2070, 2094, 2425, 2692, 8830	
137, 1443, 1443, 3674, 4623, 6033,		<code>\stex_use_document_uri:n</code>	
6050, 6142, 6165, 6325, 8969, 8992		134, 1120, 1123
<code>\stex_symbol_uri_name:n</code>		<code>\stex_use_module_uri:N</code>	135,
137, 1443, 1446, 3169, 4113, 7493, 7494		1238, 1238, 1353, 1366, 1381, 1390,	
<code>\stex_symbol_uri_path:N</code>		2332, 2349, 2368, 2375, 2408, 2433,	
.....	136, 1430, 1430	2435, 2450, 2497, 2549, 2591, 2823,	
<code>\stex_symdecl_do:</code>		2928, 2941, 2945, 2977, 2982, 3014,	
.....	147, 257, 333, 3865,	3018, 3064, 3072, 3470, 3495, 3537,	
3929, 3948, 3948, 4322, 4477, 7382		3561, 4068, 4097, 6519, 7234, 7241	
<code>_stex_symdecl_html:</code>		<code>\stex_use_module_uri:n</code>	135,
... 147, 3879, 3932, 4020, 4020, 7389		1238, 1241, 2571, 2576, 2585, 3027,	
<code>_stex_term_arg:nnn</code>		4898, 7080, 7122, 7152, 7156, 7258	
... 151, 5496, 5672, 5675, 5681, 5697		<code>\stex_use_notation:nn</code>	154
<code>_stex_term_arg:nnnnn</code>		<code>\stex_use_notation:nnTF</code>	
.....	151, 5664, 5664, 5716,	154, 4702, 4925, 5323, 5817, 6665, 6665	
5775, 5793, 5883, 6748, 6871, 6879		<code>\stex_use_op_notation:nnTF</code>	
<code>_stex_term_arg_aB:nnnnn</code>	154, 4689, 5349, 6665, 6671
... 151, 5699, 5713, 6884, 6892, 6909		<code>\c_stex_use_sref_bool</code>	27
<code>_stex_term_do_aB_clist:</code> ...	295,	<code>\stex_use_symbol_uri:N</code>	
5707, 5723, 5777, 5806, 5835, 5893,		136, 1407, 1407, 2043, 3243, 3249,	
6899, 6900, 6916, 6920, 6941, 6946		3587, 3590, 3662, 3665, 3795, 3808,	
<code>_stex_term_oma:nn</code>		3814, 3826, 3877, 3891, 3894, 3900,	
... 150, 5389, 5613, 5614, 5629, 6823		3946, 4621, 6031, 6048, 6141, 6163,	
<code>_stex_term_oma_or_omb:nn</code>		6303, 6326, 6613, 6616, 6624, 6625,	
.....	6823, 6828, 6876, 6889	6626, 6627, 6636, 7391, 7398, 7470,	
<code>_stex_term_omb:nn</code> ...	150, 5417,	7519, 7969, 8027, 8052, 8973, 8996	
5418, 5631, 5632, 5647, 6876, 6889		<code>\stex_use_symbol_uri:n</code>	
<code>_stex_term_oms:nn</code> 136, 1407, 1410, 2532, 3298,	
... 150, 4649, 4651, 5563, 5564,		4628, 5235, 5236, 5238, 5239, 5246,	
5575, 5578, 6014, 6037, 6054, 6722		5247, 5249, 5250, 6522, 6539, 6542,	
<code>_stex_term_oms_or_omv:nn</code>		6556, 6557, 6558, 6561, 6562, 6563,	
.....	4649, 4651, 4679,	7198, 7359, 7413, 7432, 7479, 7491	
4692, 4844, 4882, 4921, 4977, 5266,		<code>_stex_var_notation_macro:</code>	
5268, 5351, 5362, 5372, 5578, 6812		... 153, 4333, 4492, 6339, 6592, 6592	
<code>_stex_term_omv:nn</code>		<code>_stex_variable:nnnnnnnN</code> .	4313, 4458
.....	150, 5266, 5268, 5286,	<code>\l_stex_variables_prop</code>	
5579, 5580, 5591, 6066, 6080, 6094		148, 4311, 4352, 4441, 4510
<code>_stex_titlefragment:</code>	8429	<code>\stex_with_file_hooks:Nnn</code>	
<code>\stex_undefine:N</code>	125, 246, 249, 249, 2693
. 123, 53, 53, 56, 310, 320, 2093, 2690		stex internal commands:	
<code>\stex_uri_from_archive_file:Nn</code> .	135	<code>\l__stex_annotate_do_output_bool</code>	
<code>\stex_uri_from_pair:Nn</code> 685, 690, 693, 696, 701, 705, 8139	
.....	136, 1392, 1392, 2311	<code>__stex_annotate_env_str</code> ...	667, 668
<code>\stex_uri_from_pair:Nnn</code>		<code>__stex_check_env_str</code> 6214, 6215, 6216	
136, 1312, 1313, 1330, 1401, 1405,		<code>__stex_comps_do_defref:nn</code>	
2590, 2932, 2964, 2990, 3003, 3052		6109, 6115, 6125, 6158

_stex_comps_do_ref:nNn ..	6014,	_stex_expr_arg_inner:nn	5424, 5427, 5431, 5437
6035, 6052, 6062, 6074, 6088, 6137		\l_stex_expr_assigned_seq	4986, 5105, 5147, 5214
_stex_comps_slash:w	6145, 6169, 6181, 6185	_stex_expr_assoc_make_seq:nnn ..	5789, 5816, 5902
_stex_comps_split_slash:n	6017, 6033, 6050	_stex_expr_assoc_seq:nnnnnnn ..	5733, 5784
_stex_comps_split_slash_i:nw ..	6018, 6021, 6023	_stex_expr_check:n	5465
_stex_comps_uppercase:n	6042, 6053, 6089, 6126	_stex_expr_check:nTF ...	5434, 5440
_stex_debug:nn	105, 108, 113	_stex_expr_check_b:nn ..	5390, 5415
\l_stex_debug_cl	121, 123, 126	_stex_expr_check_comp:	5566, 5575, 5582, 5591, 5616,
_stex_debug_env_str ..	119, 120, 123	5629, 5634, 5647, 5675, 5689, 5697	
_stex_doc_check_topsect:	1698, 1704, 1710, 1716	\l_stex_expr_clist ..	4710, 4717,
_stex_doc_do_section:n	1555, 1561, 1565, 1569, 1622	4724, 4728, 4960, 4981, 4987, 4989	
\g_stex_doc_ftml_link_text_tl ..	1457, 1460, 1472	\l_stex_expr_comp_cs	5013, 5018, 5037, 5039
_stex_doc_maketitle: ...	1518, 1523	\l_stex_expr_count_int	5712, 5719, 5728, 5775, 5793, 5883
_stex_doc_orig_backmatter	1759, 1764, 1782	\l_stex_expr_cs	4962, 4968, 4975, 5818, 5821, 5841
_stex_doc_orig_endbackmatter	1760	_stex_expr_current_type:	4916, 5052, 5128
_stex_doc_orig_endfrontmatter ..	1747, 1756	\l_stex_expr_current_type_tl ...	4841, 4878, 4919, 4924
_stex_doc_orig_frontmatter ...	1746, 1751, 1771	_stex_expr_custom:n	5303, 5313, 5377
_stex_doc_set_title:n ..	1495, 1512	\l_stex_expr_customs_prop	5381, 5393, 5394,
_stex_doc_skip_section:	1643, 1649, 1653	5395, 5410, 5445, 5451, 5466, 5469	
_stex_doc_skip_section_i:	1628, 1631, 1634, 1644, 1649	\l_stex_expr_customs_seq	5382, 5399, 5400, 5401, 5411, 5467
_stex_doc_title_html:	1500, 1504, 1515	_stex_expr_do_aB_clist:	5699, 5707, 5777, 5835
\g_stex_doc_title_tl	1477, 1492, 1499, 1506, 1511	_stex_expr_do_ab_next:nn	5389, 5391, 5417, 5418
\l_stex_doc_titlefragment_bool ..	1606, 1611, 1621, 1626	_stex_expr_do_all:w ...	4759, 4768
_stex_doc_x_matter: ...	1744, 1793	_stex_expr_do_assign:nn	4862, 4866
_stex_expr_aB_arg:nnnnn	5721, 5727	_stex_expr_do_assign_list:n ...	4837, 4859
_stex_expr_aB_simple_arg:nnnnn	5742, 5773	_stex_expr_do_decl:nnnnnnnnn ..	5162, 5190
_stex_expr_add_prop_arg:nnw ...	5385, 5409, 5412	_stex_expr_do_decl_nomacro:nnnnnnnnn	5160, 5176
\l_stex_expr_after_tl	5166, 5170, 5171, 5172, 5243, 5254	_stex_expr_do_first_arg:n	4746, 4753
_stex_expr_annotate:nnn	5568, 5584,	_stex_expr_do_first_next:	4748, 4755
5600, 5617, 5635, 5651, 5661, 5915		_stex_expr_do_headterm:nn	5570, 5586, 5594, 5611, 5892
_stex_expr_arg:n	5386, 5421	_stex_expr_do_one:w ...	4757, 4763
\l_stex_expr_arg_counter_int ...	5375, 5383, 5398, 5433, 5434	_stex_expr_do_seqmap:nnnnnn ..	5739, 5848
_stex_expr_arg_do:nnn	5435, 5441, 5457, 5492, 5499		

__stex_expr_end:	__stex_expr_op_custom:n
.. 4805, 4809, 4825, 4830, 5735, 5748 5296, 5313, 5369
\l__stex_expr_field_name_str	__stex_expr_op_notation:w
..... 5092, 5096, 5097, 5131 5298, 5299, 5317, 5347
\l__stex_expr_fields_clist 4838,	__stex_expr_present: .. 4934, 5065
4860, 4867, 4885, 4888, 5205, 5206	__stex_expr_present:nn
\l__stex_expr_first_args_tl 4938, 4944, 4949, 4956, 4959
..... 4745, 4765, 4769	__stex_expr_present_entry:nn
__stex_expr_full_notation:n 4964, 4970, 4985
..... 5319, 5322	__stex_expr_present_i:w
__stex_expr_get_field_name:n 4930, 4936, 4942
..... 5091, 5103	__stex_expr_present_ii:nw 4947, 4955
__stex_expr_get_index_notation:n	\l__stex_expr_prop
..... 4696, 4701, 4764	5094, 5102, 5137, 5145, 5167, 5177,
__stex_expr_gobble:nnnnnnnn	5179, 5191, 5193, 5213, 5216, 5222
..... 5735, 5748	__stex_expr_prop_do_decls:
\l__stex_expr_iarg_tl 5828, 5830, 5841 5149, 5152
__stex_expr_invoke:nnnnN 4625, 4646	__stex_expr_prop_do_notations:
__stex_expr_invoke_field:n 5169, 5231
..... 5067, 5101	__stex_expr_range:w
__stex_expr_invoke_maybe_-	\l__stex_expr_redo_tl
field:nn 5121, 5148, 5171, 5225, 5242, 5253
5056, 5060	\l__stex_expr_reset_tl
__stex_expr_invoke_this:n 4798, 5043 4596, 4600, 4604, 4605, 4611
__stex_expr_is_argsep:n	__stex_expr_ret_cs:
5767 5504, 5509, 5540, 5545, 5550
__stex_expr_is_seqmap:n	__stex_expr_return:
5761	5519, 5522
__stex_expr_is_seqmap:nTF	__stex_expr_return_arg:n 5506, 5512
5737	\l__stex_expr_return_args_tl
__stex_expr_is_varseq:n 5502, 5513, 5518, 5527, 5547, 5552
5751	__stex_expr_return_next: 5507, 5516
__stex_expr_is_varseq:nTF 5730, 5853	\l__stex_expr_return_this_tl
__stex_expr_make_mod:n .. 4884, 4896 5501, 5518,
__stex_expr_make_oml:n .. 4888, 4903	5523, 5527, 5533, 5536, 5546, 5554
__stex_expr_make_oml:nn .. 4904, 4906	\l__stex_expr_seq
__stex_expr_make_prop: 5146, 5178, 5192, 5233
..... 4928, 5061, 5144	__stex_expr_seq_arg:n .. 4711, 4722
__stex_expr_make_prop_assign:	__stex_expr_seq_first: .. 4685, 4741
..... 4929, 5064, 5204	__stex_expr_seq_op:w .. 4684, 4688
__stex_expr_make_prop_assign:nn	\l__stex_expr_set_comp_tl
..... 5207, 5212 4772, 4827, 4831, 4991, 5116
__stex_expr_make_prop_assign_-	__stex_expr_set_custom_comp:n ..
replace:nnnn 4832, 5007, 5025
5215, 5221	__stex_expr_set_custom_i:nn
__stex_expr_make_type:n 5028, 5033
..... 4847, 4852, 4854, 4870	__stex_expr_set_customcomp:
__stex_expr_math: 4805, 4830
4675, 5292	__stex_expr_set_this:n .. 5062, 5071
__stex_expr_maybe_notation:w	__stex_expr_set_thiscomp: 4772, 4776
..... 4793, 4795, 4923	__stex_expr_set_thisnotation: ..
__stex_expr_maybe_return:n 4809, 4825
..... 5330, 5341, 5500	__stex_expr_setup:nnnnn
__stex_expr_merge:nw .. 4787, 4803 4648, 4655, 5265
\l__stex_expr_more_nextsymbol_tl	
..... 5104, 5106, 5134	
__stex_expr_notation:w	
..... 4769, 5305, 5306, 5315, 5358	
\l__stex_expr_old_seq	
..... 5860, 5863, 5867, 5872	

```

\__stex_expr_struct_top:n . 4773,
    4782, 4806, 4810, 4813, 4816, 4818
\__stex_expr_struct_type:n 4789, 4836
\__stex_expr_text: . . . . . 4675, 5311
\__stex_expr_varseq_in_map:n . . . . . 5855, 5901
\__stex_groups_do: . . . . 340, 346, 354
\__stex_groups_do_in:n . . . . 329, 333
\l__stex_groups_lvl_int 323, 325, 328
\l__stex_import_archive_str . . . .
    . . . . . 2845, 2848, 2852, 2920
\__stex_import_check_file:nnn . . . .
    . . . . . 2864,
    2865, 2866, 2880, 2881, 2882, 2909
\l__stex_import_doc_uri . . 2919, 2926
\l__stex_import_file_str . . . . .
    . . . . . 2832, 2862,
    2892, 2894, 2897, 2902, 2913, 2926
\__stex_import_import_module:nn .
    . . . . . 2955, 2963, 3000
\__stex_import_import_module_i:n
    . . . . . 2966, 2969
\__stex_import_import_module_-
    presms:nn . . . . . 2989, 3000
\l__stex_import_language_str . . . .
    . . . . . 2863, 2867, 2883, 2923
\__stex_import_load_check:n . . . .
    . . . . . 2850, 2856, 2861
\__stex_import_load_check_i:n . . . .
    . . . . . 2869, 2876
\__stex_import_load_check_ii: . . . .
    . . . . . 2873, 2891
\__stex_import_load_file: . . . . .
    . . . . . 2898, 2903, 2918
\__stex_import_load_module:NTF . . . .
    . . . . . 2822, 2831, 2843
\l__stex_import_name_str . . . . .
    . . 2847, 2864, 2865, 2866, 2878, 2922
\l__stex_import_path_str . . . . .
    . . 2846, 2849, 2854, 2877, 2879, 2921
\l__stex_import_pre_str . . . . .
    . . . . . 2849, 2853, 2910, 2911, 2913
\__stex_import_requiremodule: . . . .
    . . . . . 3005, 3010
\l__stex_import_uri . . . . .
    . . . . . 2839, 2840, 2844, 2927, 2928
\__stex_import_usemodule: 2934, 2937
\l__stex_inputs_gin_repo_str . . . .
    . . . . . 2259, 2262, 2267
\l__stex_inputs_id_seq . . . . .
    . . . . . 2196, 2197, 2202, 2210
\l__stex_inputs_libinput_files_-
    seq . . . . . 2125, 2126, 2129,
    2135, 2136, 2157, 2158, 2182, 2183,
    2186, 2194, 2206, 2207, 2217, 2218
\l__stex_inputs_path_seq . . . . .
    . . 2195, 2198, 2200, 2203, 2211, 2214
\l__stex_inputs_path_str . . . . .
    2203, 2204, 2205, 2206, 2207, 2210,
    2211, 2214, 2215, 2216, 2217, 2218
\__stex_inputs_ref:n 2057, 2078, 2088
\__stex_inputs_ref_i:n . . . . .
    . . . . . 2064, 2080, 2083
\c__stex_inputs_ref_post_str . . . .
    . . . . . 2063, 2071
\c__stex_inputs_ref_pre_str . . . .
    . . . . . 2062, 2069
\l__stex_inputs_tmp_str . . 2136, 2138
\__stex_inputs_up_archive:nn . . . .
    . . 2124, 2134, 2156, 2181, 2190, 2190
\l__stex_inputs_uri . . 2056, 2070,
    2075, 2094, 2095, 2101, 2103, 2106
\__stex_inputs_usetikzlibrary:n .
    . . . . . 2149, 2151, 2155
\__stex_inputs_usetikzlibrary_-
    i:nn . . . . . 2158, 2164
\__stex_keys_split_at_bracket:w .
    . . . . . 414, 422
\l__stex_keys_tl . . . . .
    . . . . . 364, 365, 366, 368, 371, 372
\l__stex_lang_turkish_bool . . . . .
    . . . . . 455, 463, 473
\__stex_mathhub: . . . . . 1006, 1030
\l__stex_mathhub_base_str . . . . .
    . . 1014, 1024, 1029, 1030, 1032, 1036
\l__stex_mathhub_bool . . . . .
    . . . . 972, 973, 975, 978, 987, 989, 998
\__stex_mathhub_check_manifest: .
    . . . . . 977, 985
\__stex_mathhub_check_manifest:n
    . . . . . 986, 988, 990, 995
\l__stex_mathhub_cs . . . . .
    . . . . 900, 903, 905, 909, 913, 914, 919
\__stex_mathhub_do_manifest:nn . . . .
    . . . . . 932, 950, 957, 1041
\l__stex_mathhub_file_str 1012, 1032
\__stex_mathhub_find_manifest:n 962
\__stex_mathhub_find_manifest:nn
    . . . . . 959, 968, 1065
\l__stex_mathhub_id_str . . . . .
    . . . . . 1013, 1023, 1032, 1036
\l__stex_mathhub_key 1019, 1020, 1022
\c__stex_mathhub_manifest_ior . . . .
    . . . . . 1002, 1011, 1016, 1028
\l__stex_mathhub_manifest_str . . . .
    . . . . . 960, 963, 969, 999, 1011, 1066

```

__stex_mathhub_parse_manifest:n	2385, 2407, 2408, 2409, 2411, 2416,
..... 964, 1010, 1069	2429, 2434, 2435, 2436, 2439, 2440,
__stex_mathhub_replace_https:w .	2441, 2442, 2444, 2445, 2496, 2497,
..... 1006, 1030	2498, 2499, 2500, 2504, 2505, 2531
__stex_mathhub_require:n .. 921, 944	__stex_morphisms_add_definiens:nn
__stex_mathhub_restore:nn 3255, 3260, 3319
..... 1036, 1040, 1079	__stex_morphisms_add_symbol:nnnnnnnnN
\l__stex_mathhub_seq 971, 974, 979, 3172, 3188, 3190, 3195
996, 997, 999, 1012, 1018, 1019, 1021	__stex_morphisms_add_symbol_-
__stex_mathhub_set_current:n ...	i:nnnnnnnnN 3202, 3205
..... 908, 920	\l__stex_morphisms_ass_tl
\l__stex_mathhub_str 3605, 3621, 3625, 3636, 3637, 3638
..... 838, 839, 843, 844,	__stex_morphisms_begin_copy:Nnnn
849, 855, 858, 865, 867, 868, 869, 874 3443, 3455, 3466
\l__stex_mathhub_tl	__stex_morphisms_begin_interpret:Nnnn
\l__stex_mathhub_val 1021, 1023, 1024 3508, 3521, 3533
__stex_modules_export:n . 2601, 2603	__stex_morphisms_break: . 3263, 3267
__stex_modules_html_annots: ...	__stex_morphisms_check_name:nnnn
..... 2326, 2347 3383, 3386
__stex_modules_load_meta:	\l__stex_morphisms_continue_bool
..... 2369, 2387, 2389 3324, 3327, 3340, 3359
__stex_modules_load_meta_i:n ...	__stex_morphisms_cs:nnnn 3325, 3350
..... 2391, 2395	__stex_morphisms_definiens_-
__stex_modules_load_sig: 2414, 2419	impl:nn 3235, 3320
__stex_modules_macro_short:nnn .	__stex_morphisms_do:n
..... 2274, 2278, 2281, 3088, 3101, 3116
2284, 2287, 2290, 2293, 2296, 2299	__stex_morphisms_do_decls:
__stex_modules_nested:n 3087, 3091
..... 2360, 2431, 2431	__stex_morphisms_do_elaboration:
__stex_modules_persist_module: 3130, 3133
..... 2448, 2457, 8135	__stex_morphisms_do_for_list: ..
__stex_modules_persist_not_- 3225, 3318
i:nn 2473, 2478	__stex_morphisms_do_morph:nnnn .
__stex_modules_restore_module:nnnn 3107, 3112
..... 2459, 2495	__stex_morphisms_do_morph_-
__stex_modules_restore_not_:n ..	assign:nnn 3686, 3689, 3730
..... 2509, 2514	__stex_morphisms_do_parsed_-
__stex_modules_restore_not_:i:n	assign: 3627, 3630
..... 2515, 2518, 2537	__stex_morphisms_do_parsed_-
__stex_modules_restore_not_-	newname: 3634, 3642
ii:nnnn 2522, 2526	__stex_morphisms_do_parsed_-
\l__stex_modules_sig 2421, 2425, 2426	newname:w 3644, 3646, 3650
\l__stex_modules_sigfile	\l__stex_morphisms_dones_seq ...
..... 2420, 2425, 2427 3102, 3104, 3105
__stex_modules_stringify_-	__stex_morphisms_elab:nn 3145, 3163
uri:nnn 2483, 2496	__stex_morphisms_elab_check_-
__stex_modules_stringify_-	assign:nnnn 3198, 3209
uri:nnnn 2488, 2533	__stex_morphisms_elab_check_-
\l__stex_modules_tl . 2527, 2529, 2534	rename:nnn 3166, 3217
__stex_modules_top:n ... 2360, 2363	__stex_morphisms_elab_i:nnn ...
__stex_modules_top_sig:n 3169, 3184
..... 2365, 2404, 2404	__stex_morphisms_end: ... 3634, 3650
\l__stex_modules_uri . 2374, 2375,	\l__stex_morphisms_feature_str ..
2376, 2379, 2380, 2381, 2382, 2384, 3066, 3151

<code>__stex_morphisms_get_check:nn</code> ..	<code>__stex_morphisms_total_check:nn</code>
..... 3367, 3382, 3428, 3436 3287, 3291
<code>\l__stex_morphisms_get_str</code>	<code>__stex_morphisms_total_check:nnnnnnnN</code>
..... 3365, 3388, 3292, 3295
3409, 3419, 3424, 3426, 3432, 3434	<code>\l__stex_morphisms_with_macros_-</code>
<code>__stex_morphisms_gobble:nnnnnnN</code>	bool ... 3033, 3036, 3040, 3126, 3187
..... 3412, 3416	<code>__stex_notations_activate_-</code>
<code>__stex_morphisms_iterate:nn</code> ...	not:nn 6525, 6537
..... 3330, 3334, 3343, 3345	<code>__stex_notations_add:nnnnn</code>
<code>__stex_morphisms_macro:nnnnnnnN</code> 6856, 6861, 6866
..... 3392, 3408	<code>__stex_notations_add_last:nnnnn</code>
<code>\l__stex_morphisms_mods_seq</code> 6849, 6860
..... 3323, 3347, 3348	<code>__stex_notations_add_missing_-</code>
<code>__stex_morphisms_morphism:nnnnn</code>	args:nn 6491, 6494
..... 3037, 3041, 3044	<code>__stex_notations_add_next:nnnnnn</code>
<code>\l__stex_morphisms_morphism_dom_-</code> 6851, 6855
str ... 3680, 3693, 3705, 3715, 3732	<code>\l__stex_notations_after_tl</code>
<code>\l__stex_morphisms_name_str</code> 6836, 6863
..... 3603, 3612, 3613, 3614,	<code>__stex_notations_args_end:</code>
3618, 3619, 3620, 3631, 3633, 3637 6570, 6573, 6576,
<code>\l__stex_morphisms_newname_str</code> ..	6583, 6586, 6589, 6844, 6847, 6857
.. 3604, 3623, 3624, 3632, 3633, 3634	<code>\l__stex_notations_args_tl</code>
<code>\l__stex_morphisms_next_tl</code> 6755, 6757, 6770
..... 3610, 3615, 3617	<code>__stex_notations_augment_arg:nn</code>
<code>__stex_morphisms_parse_assign:n</code> 6771, 6784
..... 3492, 3558, 3602	<code>\l__stex_notations_bind_bool</code> ...
<code>__stex_morphisms_reactivate:</code> 6290, 6292
..... 3078, 3307	<code>__stex_notations_check_aB_-</code>
<code>__stex_morphisms_rename:n</code> 3152, 3155	arg:Nn 6898, 6907, 6914, 6939
<code>__stex_morphisms_rename:nn</code>	<code>\l__stex_notations_clist_count_-</code>
..... 3156, 3159	int 6937, 6945, 6951
<code>__stex_morphisms_rename_all:</code> . 3280	<code>\l__stex_notations_code_tl</code>
<code>__stex_morphisms_renamed_-</code> 6794, 6809, 6824, 6838,
check:nn 3371, 3418	6839, 6862, 6870, 6878, 6883, 6891
<code>__stex_morphisms_renamed_-</code>	<code>__stex_notations_complex:nnnnnn</code>
check:nnnnnn 3420, 3423 6803, 6820
<code>\l__stex_morphisms_seq</code>	<code>__stex_notations_const_precs:</code> ..
..... 3607, 3608, 3610, 3612, 6364, 6404
3615, 3617, 3619, 3621, 3623, 3625	<code>\l__stex_notations_cs</code>
<code>__stex_morphisms_set:nnnnnnN</code> 6810, 6815, 6825, 6834
..... 3390, 3396, 3411	<code>__stex_notations_default_args:</code> .
<code>__stex_morphisms_set_definiens_-</code> 6756, 6767
macros: 3263, 3267	<code>__stex_notations_do_argname:nn</code> .
<code>__stex_morphisms_set_definiens_-</code> 6465, 6468
macros_i:nnnnnnn 3270, 3275	<code>__stex_notations_do_argnames:</code> ..
<code>__stex_morphisms_setup:</code> . 3077, 3082 6440, 6463
<code>__stex_morphisms_split_qm:w</code> . 3303	<code>__stex_notations_do_missing_-</code>
<code>\l__stex_morphisms_tmp</code> 3164,	args:n 6370, 6483
3168, 3171, 3172, 3178, 3185, 3220	<code>__stex_notations_fun_precs:</code> ...
<code>\l__stex_morphisms_tmp_b</code> 6369, 6416
..... 3196, 3200, 3212	<code>__stex_notations_it_not_-</code>
<code>\l__stex_morphisms_todo_tl</code>	check:nnnn 6656, 6660
..... 3329, 3333, 3346, 3359	<code>__stex_notations_it_not_i:n</code> ...
 6645, 6649, 6662

<code>\l_stex_notations_left_bracket_-</code>	<code>\l_stex_notations_str</code>
<code>str</code> 6968, 6996, 7006, 7010	... 6446, 6447, 6448, 6450, 6456, 6457
<code>_stex_notations_macro:nn</code>	<code>_stex_notations_styledefs:</code> ...
6508, 6556, 6557, 6558, 6593, 6594, 6312, 6323
6595, 6613, 6624, 6625, 6666, 6667	<code>_stex_others_newlabel:n</code> 8257, 8265
<code>_stex_notations_make_arg:nnnn</code> .	<code>_stex_others_old_newlabel:</code> ...
..... 6734, 6737 8259, 8264
<code>_stex_notations_make_arg_-</code>	<code>_stex_path:</code> 148, 158
<code>html:nn</code> 6714, 6730	<code>\l_stex_path_a_seq</code> ... 280, 286, 292
<code>_stex_notations_make_name:</code> ...	<code>\l_stex_path_a_tl</code> 292, 294
..... 6763, 6768, 6788	<code>\l_stex_path_b_seq</code> 281, 287, 293, 299
<code>_stex_notations_map_args_i:w</code> ..	<code>\l_stex_path_b_tl</code> 293, 294
..... 6569, 6573, 6576	<code>\l_stex_path_can_seq</code>
<code>_stex_notations_map_args_ii:w</code> 180, 183, 188, 196, 197, 200
..... 6582, 6586, 6589	<code>\l_stex_path_can_str</code>
<code>_stex_notations_map_cs:</code> 179, 181, 185, 197
..... 6699, 6701,	<code>_stex_path_canonicalize:N</code>
6917, 6919, 6927, 6942, 6944, 6955 161, 172, 177
<code>\l_stex_notations_missing_str</code> ..	<code>_stex_path_dodots:n</code> . 182, 186, 192
..... 6484, 6485, 6486, 6488, 6496	<code>\l_stex_path_return_tl</code>
<code>\l_stex_notations_missing_tl</code> 282, 288, 295, 298, 300
..... 6399, 6490, 6497	<code>\l_stex_path_seq</code> 211, 212,
<code>\l_stex_notations_mods_seq</code>	213, 218, 219, 221, 223, 226, 251, 252
..... 6641, 6650, 6651	<code>\l_stex_path_str</code> 149, 150, 151, 154,
<code>\l_stex_notations_name_str</code>	156, 157, 158, 160, 163, 170, 171,
..... 6764, 6790	210, 211, 212, 217, 218, 219, 221,
<code>_stex_notations_not_cs:nnnnn</code> ..	222, 223, 229, 231, 233, 239, 241, 243
..... 6642, 6643, 6653	<code>\l_stex_path_win_drive</code>
<code>_stex_notations_op_macro:nn</code> 150, 155, 162, 164
6511, 6561, 6562, 6563, 6599, 6600,	<code>_stex_path_win_take:w</code> 148, 158
6601, 6616, 6626, 6627, 6672, 6673	<code>_stex_persist_env_str</code>
<code>\l_stex_notations_opprec_tl</code> 6393, 568, 569, 570, 574, 575, 576
6406, 6409, 6411, 6418, 6421, 6423,	<code>_stex_persist_load_file:n</code>
6427, 6434, 6447, 6453, 6459, 6690 589, 592, 594, 604, 611, 645
<code>_stex_notations_parse_args:nnnnw</code>	<code>_stex_persist_read_and_write:</code> .
..... 6844, 6847, 6857 602, 630
<code>_stex_notations_parse_precs:</code> ..	<code>\c_stex_persist_sms_iow</code>
..... 6438, 6443 585, 626, 627, 642, 730
<code>\l_stex_notations_pre_tl</code>	<code>_stex_persist_write_only:</code>
..... 6823, 6838, 6875, 6888 607, 625, 639, 646
<code>\l_stex_notations_precs_seq</code> ...	<code>_stex_proof_add_counter:</code> 7653, 7912
..... 6361, 6429,	<code>_stex_proof_begin_proof:nn</code> ...
6436, 6457, 6459, 6471, 6477, 6691 7679, 7789, 7811
<code>_stex_notations_process:nnnnnn</code>	<code>\l_stex_proof_counter_intarray</code> .
..... 6793, 6799 7610, 7616,
<code>\l_stex_notations_right_-</code>	7619, 7628, 7631, 7640, 7648, 7649,
<code>bracket_str</code> . 6969, 7001, 7007, 7011	7657, 7662, 7669, 7675, 7680, 7681
<code>\l_stex_notations_seq</code>	<code>\l_stex_proof_counter_str</code>
..... 6445, 6446, 6448, 6449, 6456	... 7568, 7569, 7579, 7732, 7747,
<code>_stex_notations_set_macro:nnnnn</code>	7847, 7865, 7866, 8020, 8035, 8060
..... 6524, 6533, 6546, 6555	<code>_stex_proof_do_after_subproof:N</code>
<code>_stex_notations_simple:nnnnn</code> 7836, 7870
..... 6801, 6807	<code>_stex_proof_do_after_subproof_-</code>
	<code>i:nn</code> 7838, 7842

```

\__stex_proof_do_spf_name: 7559, 7992
\__stex_proof_do_spf_name_i: ...
..... 7573, 7840
\__stex_proof_do_spf_varname: ...
..... 7583, 7911, 8082, 8083
\__stex_proof_end_comment: .....
..... 7777, 7784,
7805, 7945, 7953, 7987, 8080, 8086
\__stex_proof_end_list: .....
.. 7795, 7830, 7936, 7940, 8005, 8093
\__stex_proof_html: .....
..... 7727, 7750, 7753, 8038, 8063
\__stex_proof_html_env_proof: ...
..... 7685, 7717
\__stex_proof_html_env_subproof:
..... 7730, 7878
\l__stex_proof_in_spfblock_bool .
..... 7677, 7772,
7778, 7790, 7812, 7862, 7890, 7902,
7929, 7936, 7940, 8001, 8073, 8089
\__stex_proof_inc_counter: .....
..... 7636, 7931, 8082, 8083
\l__stex_proof_inc_counter_bool 7609
\__stex_proof_insert_number: ...
..... 7612, 7911, 8082, 8083
\l__stex_proof_key_name_str ....
..... 7575, 7576, 7844
\l__stex_proof_key_numname_str ..
..... 7574, 7579, 7845
\l__stex_proof_key_showname_bool
..... 7846
\__stex_proof_make_step_macro:Nnnnn
..... 7985, 8082, 8083, 8084
\__stex_proof_number_as_string:N
..... 7568, 7623,
7732, 7865, 7894, 7907, 7996, 8020
\__stex_proof_proof_box_tl 7590, 8119
\__stex_proof_remove_counter: ...
..... 7665, 7930
\__stex_proof_set_after_subproof:n
..... 7849, 7891, 7918
\__stex_proof_set_after_subproof_-
i:n ..... 7851, 7852, 7859
\__stex_proof_set_after_subproof_-
with_number:n ..... 7854, 7904
\__stex_proof_set_after_subproof_-
with_number_i:n . 7856, 7857, 7864
\__stex_proof_set_aftergroup: ...
..... 7862, 7867, 7869
\__stex_proof_set_aftergroup_-
double: ..... 7862
\__stex_proof_start_comment: ...
..... 7771, 7784,
7792, 7927, 7949, 8009, 8075, 8096
\__stex_proof_start_list:n .....
..... 7791, 7821, 7911, 8005, 8093
\__stex_proof_step_html:nn .....
..... 8005, 8018, 8093
\__stex_proof_step_html_i:nn ...
..... 8003, 8011, 8045, 8091, 8098
\__stex_refs_check_i:nnnn 1901, 1909
\__stex_refs_check_in:nnnn 1994, 2002
\l__stex_refs_default_archive ...
..... 1928, 2021, 2027, 2029, 2031
\l__stex_refs_default_file .....
..... 1923, 1927, 1930, 2020, 2032
\l__stex_refs_default_relpath ...
..... 1929, 2022, 2025, 2032
\l__stex_refs_default_title ....
..... 1926, 2019, 2034
\l__stex_refs_file .....
..... 1883, 1884, 1902, 1927,
1933, 1941, 1950, 1953, 1991, 1995
\__stex_refs_find: ..... 1885, 1899
\__stex_refs_find_in: ... 1951, 1990
\l__stex_refs_in .....
.. 1874, 1919, 1939, 1943, 1965, 1966
\__stex_refs_in: .... 1934, 1944, 1948
\__stex_refs_in_text:nn .. 1961, 1976
\__stex_refs_in_text:w ... 1977, 1984
\c__stex_refs_iow .....
..... 1796, 1797, 1798, 1819
\c__stex_refs_iow_str ... 1799, 1930
\l__stex_refs_key ..... 1873, 1910
\__stex_refs_local:n 1844, 1850, 1880
\__stex_refs_local_ref:n .....
..... 1854, 1861, 1921, 1924, 1931
\__stex_refs_maybe_in: ... 1893, 1918
\l__stex_refs_new_tl .....
..... 1813, 1814, 1815, 1817, 1822
\__stex_refs_remote:nn ... 1846, 1872
\__stex_refs_stop: ..... 1977, 1984
\l__stex_refs_str ..... 7407
\l__stex_refs_tmp .... 1842, 1887,
1891, 1905, 1912, 1949, 1955, 1959,
1960, 1961, 1967, 1971, 1998, 2004
\l__stex_refs_unnamed_counter_-
int ..... 2008
\__stex_refs_uppercase:n . 1985, 1988
\l__stex_refs_uri 1876, 1877, 1878,
1882, 1891, 1892, 1911, 1920, 1921,
1924, 1931, 1966, 1971, 1991, 2003
\__stex_seqs_add: ..... 4489, 4508
\l__stex_seqs_args_tl ... 4568, 4570
\l__stex_seqs_cs ..... 4566, 4570
\__stex_seqs_html: ..... 4487, 4535
\__stex_seqs_macro: ..... 4490, 4522

```

\l__stex_seqs_range_clist	__stex_statements_setup:n 7361,
..... 4480, 4516, 4529, 4555	7361, 7425, 7429, 7444, 7449, 7452
\g__stex_smsmode_allowed_escape_-	__stex_statements_setup_def:...
tl 2639, 2642, 2769 7410, 7410, 7426, 7450
\g__stex_smsmode_allowed_import_-	__stex_statements_setup_named:n
env_seq 2664, 2667, 2792, 2795 7371, 7375
\g__stex_smsmode_allowed_import_-	__stex_statements_setup_noname:
tl 2657, 2661, 2787 7370, 7394
\g__stex_smsmode_allowed_tl	__stex_structures_begin:nn
..... 2634, 2637, 2765 7032, 7042, 7076, 7117
\g__stex_smsmode_allowedenvs_seq	__stex_structures_check_-
..... 2644, 2647, 2774, 2777	def:nnnnnnnn 7135, 7178
\g__stex_smsmode_bool 2673, 2676, 2716	__stex_structures_do_externals:
__stex_smsmode_check_begin:Nn 7036, 7062, 7091, 7141
..... 2774, 2792, 2803	__stex_structures_do_usestructure:
__stex_smsmode_check_cs:N 2738, 2744 7235, 7242, 7248
__stex_smsmode_check_cs:NNn ...	\l__stex_structures_exstruct_-
..... 2737, 2751	name_str ... 7114, 7117, 7139, 7188
__stex_smsmode_check_end:Nn ...	__stex_structures_extend_-
..... 2777, 2795, 2811	i:nnnnnnnn 7160, 7174
\l__stex_smsmode_cycles_seq	__stex_structures_extend_-
..... 2679, 2685, 2687	ii:nnnn 7159, 7166
__stex_smsmode_do:w	__stex_structures_extend_-
... 2733, 2735, 2737, 2740, 2748,	structure:nnnn 7114, 7151
2767, 2779, 2797, 2808, 2814, 2816	\l__stex_structures_extname_-
__stex_smsmode_do_aux:N	count 7186, 7188, 7191
..... 2737, 2747, 2758	__stex_structures_get:w . 7220, 7230
__stex_smsmode_do_aux_curr:N ...	\l__stex_structures_imports_seq .
..... 2696, 2704, 2760 7067, 7072, 7077,
__stex_smsmode_do_aux_imports:N	7102, 7107, 7119, 7249, 7252, 7255
..... 2696, 2786	\l__stex_structures_last_name_-
__stex_smsmode_do_aux_normal:N .	str 7111, 7114
..... 2704, 2764	\l__stex_structures_name_str ...
\g__stex_smsmode_import_setup_tl 7020, 7025, 7027, 7044,
..... 2658, 2662, 2668, 2697	7045, 7050, 7054, 7059, 7199, 7202
\l__stex_smsmode_importmodules_-	__stex_structures_new_extstruct_-
seq 2682	name: 7101, 7184
__stex_smsmode_in_smsmode:n ...	\l__stex_structures_parent_uri ..
..... 2694, 2713 7111, 7114
\l__stex_smsmode_sigmodules_seq 2683	\l__stex_structures_replace_-
__stex_smsmode_smsmode_do:	this_tl 7063
..... 2726, 2733	\l__stex_structures_struct_uri ..
__stex_smsmode_start_smsmode:n 7049, 7054
..... 2698, 2705, 2724	__stex_styles_apply_patch:n 488, 503
__stex_statements_add_definiens:n	\g__stex_styles_counters_seq ...
..... 7485, 7490 24, 63, 64, 761, 800, 801
__stex_statements_definiens_-	__stex_styles_css_patch:nnnn ...
inner:nnnnnnN 7494, 7498 16, 97, 521, 753, 834
__stex_statements_force_id: ...	\l__stex_styles_first_str
..... 7376, 7404, 7424, 7448 60, 62, 81, 797, 799, 818
__stex_statements_html_keyvals:nn	__stex_styles_patch:nnn ... 484, 495
..... 7298, 7332, 7343	__stex_styles_patch:nnnn
__stex_statements_make_macro:nnn 4, 88, 518, 741, 825
..... 7322, 7325	

__stex_styles_patch_html:	__stex_uris_absolute:N
34, 74, 77, 771, 811, 814	1324, 1355, 1359
__stex_styles_patch_html_c: . . .	\l__stex_uris_archive 1173, 1177, 1180
26, 66, 69, 763, 803, 806	__stex_uris_as_qm:nnn . . . 1286, 1288
__stex_styles_patch_i:nnn	__stex_uris_doc:nnnnn
21, 42, 758, 779	1121, 1124, 1126
__stex_syms_activate_i:nnnnnnnn	__stex_uris_doc_from_archive_-
4079, 4092, 4096	file:NN . . . 1167, 1170, 1229, 1231
__stex_syms_add_decl: . . . 3931, 3934	__stex_uris_end: . . . 1397, 1400, 1404
\l__stex_syms_args_tl . . . 4052, 4054	\l__stex_uris_file . . 1165, 1166, 1167
\l__stex_syms_cs	__stex_uris_first_three:nnnn . . .
. . 4050, 4054, 4174, 4176, 4178, 4206	1434, 1437, 1440
__stex_syms_def_style: . . 3816, 3824	\l__stex_uris_lang_str . . . 1183,
__stex_syms_do_args: . . . 3951, 3958	1192, 1194, 1195, 1200, 1201, 1203,
__stex_syms_get_from_one_-	1204, 1205, 1206, 1208, 1209, 1210
string:n 4215, 4289	__stex_uris_maybe_relative:N . . .
__stex_syms_get_symbol_from_cs:	1324, 1332
4180, 4195	__stex_uris_module:nnn
__stex_syms_get_symbol_from_-	1239, 1242, 1244
modules:nn 4217, 4248	\l__stex_uris_name_str
__stex_syms_get_symbol_from_-	. . . 1182, 1190, 1191, 1194, 1215,
string:n . . . 4181, 4183, 4186, 4210	1317, 1342, 1348, 1364, 1379, 1388
__stex_syms_has_definiens:nnnnnnnN	__stex_uris_new_mod:nnnnnn
4110, 4117	1295, 1299
__stex_syms_it_decl_check:nnnn .	__stex_uris_new_nested_mod:nnnnn
4150, 4158	1293, 1307
__stex_syms_it_decl_i:n	__stex_uris_pair_in_archive:Nn .
4142, 4146, 4160	1327, 1384
__stex_syms_maybe_activate:nnnnnnnn	__stex_uris_pair_no_archive:N . .
4086, 4090	1368, 1372
\l__stex_syms_mods_seq	__stex_uris_parent:NNnnn
4138, 4147, 4148	1268, 1272, 1275
\l__stex_syms_name 4213, 4219	__stex_uris_parse_i:Nw . . 1394, 1400
\l__stex_syms_name_str	__stex_uris_parse_ii:Nw . 1396, 1404
4250, 4262, 4264	\l__stex_uris_path_seq
__stex_syms_parse_arity: 3950, 3969	1181, 1189, 1190
\l__stex_syms_path_str	\l__stex_uris_path_str
4251, 4261, 4268, 4273, 4274	. . 1318, 1320, 1323, 1363, 1374, 1387
\l__stex_syms_seq	__stex_uris_split_file:N 1171, 1188
4212, 4213, 4214, 4218	__stex_uris_split_file_i: 1196, 1199
__stex_syms_set_textsymdecl_-	__stex_uris_split_file_ii: . . 1218
macro:nnn 3894, 3913	__stex_uris_symbol:nnnn
__stex_syms_style: 3785, 3793	1408, 1411, 1413
__stex_syms_sym_cs:nnnnnnnnN . . .	__stex_uris_with_elem:nnnnnn . . .
. . 4121, 4122, 4132, 4139, 4140, 4153	1159, 1161
__stex_syms_sym_from_str_i:nnnn	__stex_uris_with_language:nnnnnn
4223, 4257, 4270, 4276	1153, 1155
__stex_syms_sym_i_finish:nnnnnnnN	__stex_vars_add: 4324, 4350
4229, 4236	\l__stex_vars_args_tl . . . 4409, 4411
__stex_syms_sym_i_gobble:nnnnnn	\l__stex_vars_bind_bool . . 4312, 4393
4231, 4234	__stex_vars_check_var:nnnnnnnnN
__stex_syms_top:n	4442, 4446
3784, 3807, 3921, 3921	\l__stex_vars_cs 4407, 4411
__stex_syms_uri_match:n 4284	__stex_vars_get_var:n . . . 4434, 4440

<code>__stex_vars_html:</code>	4326, 4376	<code>\stexstylesymdef</code>	106
<code>__stex_vars_macro:</code>	4325, 4363	<code>\stexstyletextsymdecl</code>	106
<code>__stex_vars_set_vars:nnnnnnN</code>	4448, 4451, 4456	<code>\stexstyleusemodule</code>	106
<code>stex_annotate_env (env.)</code>	128, 709	<code>\stexstylevardef</code>	106
<code>stex_env_node (env.)</code>	128, 709	<code>\stexstylevarnotation</code>	106
<code>\stexcommentfont</code>	159, 7774, 8130	<code>\stexstylevarseq</code>	106
<code>\stexdoctitle</code>	138, 1477, 1580, 1592, 1618, 2343, 8621, 8805, 9092, 9192, 9195, 10308	<code>\stopsolutions</code>	114, 9309
<code>\STEXexport</code>	43, 88, 89, 143, 2599, 2656	str commands:	
<code>\STEXftmlink</code>	138, 1450	<code>\c_ampersand_str</code>	1129, 1131, 1132, 1134, 1247, 1249, 1416, 1418, 1419
<code>\stexhtmlfalse</code>	692	<code>\c_backslash_str</code>	156, 1866
<code>\stexhtmltrue</code>	689	<code>\c_colon_str</code>	1018
<code>\STeXInternalNewSRefLabel</code> 1815, 1817, 2015		<code>\c_dollar_str</code>	6478
<code>\STEXInternalNotation</code> 154, 6391, 6757, 6792		<code>\c_hash_str</code>	6485, 6496, 6517
<code>\STeXInternalSRefLabel</code>	1820, 1901, 1994, 2013	<code>\c_left_brace_str</code>	10050, 10092, 10139, 10180, 10200, 10225
<code>\STEXinvisible</code>	83, 129, 709, 6497, 7690, 7692, 7883, 7885, 8158, 8159	<code>\c_percent_str</code>	89, 90, 239
<code>\STEXlinkftml</code>	138, 1450	<code>\c_right_brace_str</code>	10067, 10109, 10149, 10190, 10206, 10229
<code>\STEXRestoreNotsEnd</code>	2474, 2512, 2519	<code>\str_case:Nn</code>	1022
<code>\STEXsetlinkftml</code>	138, 1450	<code>\str_case:nn</code>	5416, 6868
<code>\stexstyle<environmentname></code>	127	<code>\str_case:nnTF</code> 48, 785, 1677, 3973, 5443, 5471, 6738, 8381, 8959, 8982	
<code>\stexstyle<macroname></code>	127	<code>\str_clear:N</code>	6, 7, 8, 91, 155, 379, 380, 387, 405, 743, 744, 745, 849, 969, 1318, 2262, 2305, 2862, 3603, 3604, 3680, 3745, 3746, 3747, 3748, 3758, 3759, 3763, 3780, 3842, 3884, 3885, 4433, 6267, 6268, 6269, 6270, 6486, 7020, 7265, 7266, 7268, 7550, 7551, 7624, 7981, 8515, 8861, 8862, 9026, 9027, 9231, 9248, 9469, 10436, 10437, 10438
<code>\stexstyleassertion</code>	106	<code>\str_const:Nn</code>	1051, 8300
<code>\stexstyleassign</code>	106	<code>\str_count:n</code>	59, 64
<code>\stexstyleassignMorphism</code>	106	<code>\str_gput_right:Nn</code>	9993
<code>\stexstylecopymod</code>	106	<code>\str_gset:Nn</code>	7847
<code>\stexstylecopymodule</code>	106	<code>\str_gset_eq:NN</code>	844, 7844, 7845
<code>\stexstyledefinition</code>	106	<code>\str_if_empty:NTF</code>	43, 61, 120, 162, 181, 458, 569, 575, 650, 656, 780, 798, 837, 839, 858, 960, 963, 1066, 1323, 1345, 1843, 2330, 2364, 2617, 2832, 2892, 3618, 3632, 3693, 3856, 3922, 4025, 4028, 4031, 4034, 4037, 4261, 4268, 4318, 4381, 4384, 4387, 4390, 4435, 4471, 4474, 4539, 4542, 4545, 4548, 5096, 6215, 6279, 6405, 6417, 6426, 6472, 7024, 7044, 7349, 7362, 7363, 7370, 7405, 7560, 7561, 7574, 7575, 7731, 7742, 7743, 7850, 7851, 7855, 7856, 7893, 7906, 7989, 7995, 8019, 8030, 8031, 8055, 8056, 8371, 8435, 8612, 9080, 9175, 9180, 9257,
<code>\stexstyleexample</code>	106		
<code>\stexstyleextstructure</code>	106		
<code>\stexstylegnote</code>	10194		
<code>\stexstyleimportmodule</code>	106		
<code>\stexstyleinterpretmod</code>	106		
<code>\stexstyleinterpretmodule</code>	106		
<code>\stexstylemathstructure</code>	106		
<code>\stexstylemcb</code>	9571, 9575		
<code>\stexstyleMMTinclude</code>	106		
<code>\stexstylemodule</code>	106		
<code>\stexstylenotation</code>	106		
<code>\stexstyleparagraph</code>	106, 127		
<code>\stexstyleproblem</code>	10030, 10071		
<code>\stexstyleproof</code>	106		
<code>\stexstylerealization</code>	106		
<code>\stexstylerealize</code>	106		
<code>\stexstylerenamedekl</code>	106		
<code>\stexstylerequiremodule</code>	106		
<code>\stexstylescb</code>	9736, 9740		
<code>\stexstylespfsketch</code>	106		
<code>\stexstylesubproblem</code>	10113, 10153		
<code>\stexstylesubproof</code>	106		
<code>\stexstylesymdecl</code>	106		

9300, 9320, 9368, 9418, 9459, 9478, 9491, 9509, 10196, 10213, 10239, 10251, 10254, 10460, 10464, 10475	7895, 7908, 7997, 8425, 9259, 9322, 9370, 9420, 9480, 10215, 10459, 10461
<code>\str_if_empty:nTF</code>	<code>\str_set_eq:NN</code> 855,
... 89, 142, 193, 496, 826, 3667, 4091	1023, 1024, 1177, 1194, 1195, 1201,
<code>\str_if_eq:NNTF</code> 294, 1930, 2848	1206, 1891, 1927, 1928, 1929, 2849,
<code>\str_if_eq:nnTF</code> 58,	2863, 3067, 3809, 3832, 4341, 4500,
63, 90, 157, 194, 195, 266, 462,	6282, 6324, 6336, 6346, 7027, 7202,
570, 576, 668, 1034, 1301, 1342,	7301, 7364, 7380, 7407, 7834, 8070
1910, 2003, 2894, 3176, 3328, 3388,	<code>\str_uppercase:n</code> 1543, 8507
3409, 3424, 3432, 3685, 3701, 4264,	<code>\l_tmpa_str</code>
4285, 4447, 4450, 4729, 6191, 6201,	461, 465, 466, 467, 469, 1004, 1006,
6216, 6408, 6420, 6432, 6542, 7484,	1222, 1223, 3062, 3064, 3067, 3071
7598, 8439, 8442, 8507, 8693, 9581	<code>\string</code> 1879, 9979, 10020
<code>\str_if_eq_p:nn</code> 4225, 4226, 4294, 4295	<code>\subparagraph</code> 26, 84
<code>\str_if_in:NnTF</code> 6143, 6167, 6496	<code>subproblem</code> (env.) 9162
<code>\str_item:Nn</code> 157, 3963	<code>subproof</code> (env.) 158, 7834
<code>\str_item:nn</code> 266	<code>\subproof</code> 7697, 7957
<code>\str_lowercase:n</code> 9581	<code>\subproofautorefname</code> 7834
<code>\str_map_break:</code> 3974	<code>\subsection</code> 26, 84, 139
<code>\str_map_break:n</code> . 3975, 3979, 3983,	<code>\subsubsection</code> 26, 84
3987, 3991, 3995, 3999, 4003, 4007	<code>\svar</code> 97,
<code>\str_map_function:nN</code> 10003	150, 3955, 5275, 5875, 5877, 5910, 5912
<code>\str_map_inline:Nn</code> 3972	<code>\symbol</code> 91
<code>\str_new:N</code>	<code>\symdecl</code> 22, 30, 31, 36, 41, 89–91, 101,
453, 1479, 2020, 2021, 2022, 3775, 9985	106, 143, 147, 149, 257, 3308, 3775,
<code>\str_put_left:Nn</code> 62, 799	8171, 8173, 8201, 8212, 8226, 8229
<code>\str_put_right:Nn</code> 7631	<code>\symdef</code> 31–
<code>\str_range:Nnn</code> 2138	33, 37, 90, 97, 143, 147, 149, 257,
<code>\str_range:nnn</code> 59, 64	3310, 3804, 8148, 8150, 8153, 8156,
<code>\str_replace_all:Nnn</code> 156	8161, 8164, 8166, 8168, 8170, 8180,
<code>\str_set:Nn</code> 13, 14, 44, 46,	8181, 8182, 8183, 8184, 8185, 8189,
50, 51, 52, 53, 54, 55, 56, 60, 83, 94,	8195, 8207, 8208, 8222, 8224, 8233,
149, 150, 151, 154, 170, 254, 459,	8234, 8237, 8239, 8242, 8244, 8246
461, 750, 781, 783, 787, 788, 789,	<code>\Symname</code> 91, 102, 152, 6017
790, 791, 792, 793, 797, 879, 999,	<code>\symname</code> 18,
1004, 1012, 1013, 1014, 1020, 1021,	19, 23, 56, 91, 92, 102, 104, 152, 6017
1030, 1173, 1210, 1215, 1222, 1235,	<code>\symref</code> 18, 19,
1481, 1527, 1799, 1873, 1882, 1883,	23, 42, 90, 91, 95, 102, 104, 152, 6010
1912, 1941, 2025, 2027, 2029, 2032,	<code>\symrefemph</code> 104, 105, 151, 5971, 8950
2034, 2062, 2063, 2136, 2203, 2214,	<code>\symuse</code> . . 42, 91, 149, 4555, 4585, 4879,
2225, 2230, 2241, 2246, 2256, 2259,	4886, 4897, 4980, 5046, 5123, 5124,
2263, 2349, 2350, 2853, 2867, 2879,	5874, 5888, 5906, 5909, 5919, 5923
2883, 2913, 2945, 2946, 2982, 2983,	sys commands:
3018, 3019, 3062, 3066, 3268, 3269,	<code>\sys_get_shell:nnN</code> 80
3365, 3398, 3426, 3434, 3614, 3620,	<code>\sys_if_platform_windows:TF</code>
3624, 3732, 3753, 3782, 3806, 3855, 86, 147, 228, 238, 262
3857, 3859, 3861, 3923, 3977, 3981,	
3985, 3989, 3993, 3997, 4001, 4005,	
4009, 4251, 4317, 4319, 4459, 4470,	
4472, 4475, 4661, 5092, 5097, 6142,	
6144, 6168, 6280, 6325, 6484, 7025,	
7045, 7056, 7186, 7188, 7191, 7381,	

T

<code>\target</code>	124
<code>\test</code>	76, 118
<code>test</code>	113
<code>\testbigspace</code>	9937
<code>\testemptypage</code>	75, 118
<code>\testemptypage</code>	9931, 10271

testheading (env.)	77, 119	\hwexam@checktime	10400
\testheading	10373	\hwexam@duration	10376, 10379, 10385, 10390
\testmedspace	9936	\hwexam@kw@due	10327
\testnewpage	75, 118	\hwexam@kw@forgrading	10368
\testnewpage	9938, 9939	\hwexam@kw@given	10324
\testnewpageInProblem	9939	\hwexam@kw@grade	10369
\testsmallspace	75, 75, 75, 118, 118, 118	\hwexam@kw@probs	10369
\testsmallspace	9935	\hwexam@kw@pts	10370
\testspace	75, 118	\hwexam@kw@reached	10371
\testspace	9280, 9934	\hwexam@kw@sum	10369
\TeX	23	\hwexam@kw@testemptypage	9932, 10275
\tex	23	\hwexam@min	10377, 10384, 10389, 10400
TeX and L ^A T _E X 2 _ε commands:		\hwexam@minutes@kw	10377
\@	2168, 2171, 2175	\hwexam@reqpts	10386, 10391, 10403, 10407
\@addtoreset	9042	\hwexam@tools	10387, 10392
\@arabic	8816	\hwexam@totalmin	10399, 10400
\@author	8795	\hwexam@totalpts	10398, 10407
\@auxout	1816, 8262, 9156	\if@bonuspoints	10402
\@bonuspointsfalse	10404	\if@rustex	672
\@bonuspointstrue	10409	\itemize@inner	8547, 8553, 8562
\@currentHref	1823, 7895, 7908, 7997	\itemize@label	8551, 8554, 8557
\@currentlabel	7894, 7895, 7907, 7908, 7996, 7997, 8611	\itemize@level	8545, 8550, 8553, 8562
\@currentlabelname	1824	\itemize@outer	8546, 8550
\@currenvir	8481, 8482	\lst@mhrepos	2241, 2246, 2250
\@dblarg	8788, 8797	\ltx@ifpackageloaded	2222, 2238, 2253, 8120, 9005, 9605, 9636, 9960
\@gobble	48	\mdf@patchamsthm	8584
\@ifnextchar	47	\metakeys@show@keys	8548
\@ifstar	8736	\notesslides@slidelabel	8587, 8603
\@listdepth	2079	\ns@author	8788, 8789
\@mainmatterfalse	1753, 1766	\ns@title	8797, 8798
\@mainmattertrue	1777, 1788	\pgf@temp	2165, 2166, 2167
\@notprerr	1688	\pgfkeys@spdef	2165
\@onlypreamble	1688	\pgfutil@empty	2167
\@rustexfalse	664	\pgfutil@InputIfFileExists	2174
\@title	1520, 1521, 8804	\prematurestop@endsfragment	8480, 8483, 8489
\activate@excursion	8843, 8848	\problem@kw@*	9002
\bbl@loaded	9006, 9961	\problem@kw@correct	9620, 9758
\beamer@shortauthor	8791, 8793, 8808	\problem@kw@feedback	9513, 10255
\beamer@shorttitle	8800, 8802, 8811	\problem@kw@grading	9459, 10196
\c@chapter	8462	\problem@kw@hint	9359
\c@framenummer	8816	\problem@kw@minutes	9130, 9223, 9526, 10039, 10081, 10127, 10168
\c@part	8474	\problem@kw@note	9407
\compemph@uri	105, 151, 5927, 6151, 6705	\problem@kw@points	9510, 10034, 10076, 10122, 10163, 10252
\correction@table	10352	\problem@kw@pts	9125, 9218, 9521
\defemph@uri	105, 151, 5960, 6174	\problem@kw@solution	9299
\define@key	2223, 2239, 2254	\problem@kw@wrong	9622, 9760
\Gin@eheight	10525, 10528, 10533, 10538	\problem@restore	9157, 9161, 10346
\Gin@ewidth	8727, 8728, 10524, 10534, 10538	\stex@backend	128, 662
\Gin@exclamation	10524, 10525, 10533		
\Gin@mhrepos	2225, 2230, 2234, 2256, 2263, 2268		
\hwexam@bonuspts	10406		

<code>\symrefemph@uri</code>	104, 105, 151, <u>5971</u> , 6014, 6037, 6054, 8969, 8992	<code>\thistitle</code>	105, 2341, 2342, 2343, 7300, 9096, 9143, 9144, 9148, 9149
<code>\varemp@uri</code>	105, 151, <u>5951</u> , 6062, 6076, 6090	<code>\thistype</code>	3797, 3828
tex commands:		<code>\thisvarname</code>	4339, 4498, 6344
<code>\tex_undefined:D</code>	54	<code>\throwaway</code>	144
<code>\texname</code>	23	<code>\tikzinput</code>	121, 141, 2268, 10517, 10522, 10548
<code>\text</code>	19	tikzinput commands:	
<code>\textbf</code>	18, 5965, 8362, 8968, 8991, 9144, 9147, 9912, 10317, 10322	<code>\c_tikzinput_image_bool</code>	38, 10506, 10513
<code>\textcolor</code>	9908	<code>\times</code>	8214, 8218
<code>\textsymdecl</code>	23, 90, 147, 149, 3309, <u>3841</u>	<code>\tiny</code>	152, 6232, 8588, 8603, 8968, 8991
<code>\texttt</code>	9892, 9908	<code>\title</code>	76, 119
<code>\textwarning</code>	63, 111	<code>\title</code>	1612, 8797
<code>\textwidth</code>	8588, 8725, 8747, 10365	<code>titlefragment (env.)</code>	139, <u>1606</u>
<code>\the</code>	334, 335, 337, 339, 341, 353, 355, 2168, 2169, 2170	tl commands:	
<code>\theassignment</code>	10305, 10317	<code>\tl_clear:N</code>	282, 393, 505, 543, 1280, 1548, 1801, 1802, 1803, 1842, 1949, 2309, 2684, 2947, 2984, 3020, 3045, 3164, 3196, 3346, 3364, 3605, 3760, 3761, 3762, 3800, 3831, 3843, 3844, 3883, 3902, 3959, 4172, 4340, 4483, 4499, 4656, 4665, 5074, 5104, 5148, 5285, 5357, 5502, 5666, 5667, 5668, 5669, 5996, 5997, 5998, 6006, 6149, 6345, 6362, 6373, 6374, 6464, 6490, 6629, 6709, 6720, 6755, 6947, 7019, 7217, 7269, 7270, 7271, 7368, 7536, 7537, 7538, 7539, 7592, 7593, 7594, 8860, 9024, 9470, 9588, 9590, 9591, 9833, 10280, 10281, 10282, 10357, 10358, 10359, 10384, 10385, 10386, 10387
<code>\thechapter</code> 1567, 1633, 1666, 1706, 1717, 8388, 8393, 8397, 8401, 8405, 8409		<code>\tl_const:Nn</code>	1053, 6971, 6972, 9842, 9843
<code>\theframenumber</code>	8611	<code>\tl_count:N</code>	5448, 5454
<code>\thepage</code>	10273	<code>\tl_count:n</code>	5752, 5762, 5768
<code>\theparagraph</code>	8405, 8409	<code>\tl_gclear:N</code>	2379, 2439
<code>\thepart</code>	1563, 1630, 1662, 1700, 8383	<code>\tl_gput_left:Nn</code>	365, 367, 368
<code>\theplainsproblem</code>	9046, 9047	<code>\tl_gput_right:Nn</code>	335, 2541, 2548, 2605, 2637, 2642, 2661, 2662, 2668, 2992, 8844
<code>\thesection</code>	8393, 8397, 8401, 8405, 8409, 9047	<code>\tl_gset:Nn</code>	257, 337, 360, 361, 933, 1032, 1457, 1460, 1492, 1499, 2016, 2504, 4600, 4604, 6377, 7860, 7866, 8258, 9202, 9205, 10349, 10350
<code>\thesproblem</code>	9043, 9047, 9147, 9157, 10031, 10052, 10072, 10094, 10119, 10141, 10159, 10182, 10305	<code>\tl_gset_eq:NN</code>	81, 1080, 1081, 1086, 1087, 2627, 3121
<code>\thesubparagraph</code>	8409	<code>\tl_head:N</code>	4178, 4777, 5008, 5026
<code>\thesubsection</code>	8397, 8401, 8405, 8409	<code>\tl_head:n</code>	5753, 5763, 5769, 6908
<code>\thesubsubsection</code>	8401, 8405, 8409	<code>\tl_if_empty:NTF</code>	298, 557, 1463, 1511, 1520, 1593, 1613, 1857, 1887, 1889, 1919, 1923, 1955, 1957, 1979, 2331, 2342, 2367, 3051, 3168,
<code>\this</code>	50–52, 98, 100, 156, 273, 4658, 4996, 5080, 5081, 5086, 5111, <u>7192</u>		
<code>\thisarchive</code>	2981		
<code>\thisargs</code>	3799, 3830		
<code>\thiscopyname</code>	3469, 3494, 3536, 3560		
<code>\thisdeclname</code>	3796, 3827, 3901, 6325		
<code>\thisdecluri</code>	3795, 3826, 3900, 6326		
<code>\thisdefiniens</code>	3798, 3829		
<code>\thisfor</code>	7302		
<code>\thismodulename</code>	2350, 2946, 2983, 3019		
<code>\thismoduleuri</code>	2349, 2945, 2982, 3018, 3470, 3495, 3537, 3561		
<code>\thisname</code>	7301		
<code>\thisnotation</code> 3833, 4342, 4501, 6327, 6347			
<code>\thisnotationvariant</code>	3832, 4341, 4500, 6324, 6346		
<code>\thisstyle</code>	505, 508, 509, 510, 543, 546, 547, 548, 549, 557, 560, 561, 2947, 2984, 3020, 3800, 3831, 3902, 4340, 4499, 6345		

3200, 3240, 3369, 3373, 3636, 3871,	6781, 6870, 6878, 6883, 6891, 6954,
3940, 4042, 4045, 4048, 4188, 4399,	7153, 9857, 9869, 10361, 10362, 10363
4402, 4405, 4413, 4481, 4558, 4561,	\tl_set:Nn . 90, 91, 93, 94, 295, 308,
4564, 4572, 5325, 5337, 5384, 5468,	492, 497, 499, 536, 537, 827, 828,
5524, 5567, 5583, 5618, 5636, 5795,	830, 831, 1043, 1179, 1279, 1282,
5985, 6244, 6248, 6252, 6257, 6356,	1333, 1336, 1361, 1376, 1385, 1602,
6365, 6371, 6568, 6581, 6598, 6615,	1751, 1756, 1764, 1813, 1874, 1905,
6718, 6754, 7196, 7306, 7467, 7516,	1960, 1998, 2004, 2374, 2407, 2421,
7756, 7758, 7761, 7764, 8126, 8728,	2434, 2496, 2527, 2574, 2839, 2845,
8872, 8958, 8981, 9088, 9091, 9097,	2846, 2847, 2919, 2981, 3056, 3185,
9143, 9148, 9188, 9191, 9194, 9199,	3212, 3220, 3261, 3276, 3389, 3400,
9299, 9359, 9407, 9496, 9512, 9620,	3401, 3402, 3403, 3404, 3410, 3469,
9622, 9625, 9657, 9659, 9661, 9669,	3470, 3494, 3495, 3536, 3537, 3560,
9758, 9760, 9763, 9791, 9793, 9795,	3561, 3615, 3621, 3625, 3795, 3826,
9803, 9909, 10244, 10300, 10307,	3862, 3900, 3925, 4052, 4198, 4200,
10318, 10323, 10326, 10376, 10403	4201, 4202, 4203, 4204, 4238, 4240,
\tl_if_empty:nTF ... 265, 274, 362,	4241, 4242, 4243, 4244, 4299, 4301,
901, 1100, 1102, 1106, 1110, 1112,	4302, 4303, 4304, 4305, 4364, 4409,
1128, 1133, 1162, 1246, 1304, 1322,	4461, 4462, 4463, 4464, 4465, 4523,
1415, 1472, 1978, 2026, 2148, 2308,	4568, 4599, 4601, 4608, 4620, 4621,
2736, 3027, 3047, 3059, 3113, 3236,	4622, 4641, 4657, 4662, 4663, 4664,
3296, 3673, 4077, 4118, 4159, 4176,	4669, 4745, 4772, 4790, 4826, 4827,
4812, 4815, 4861, 4871, 4961, 4963,	4831, 4834, 4841, 4842, 4878, 4918,
5047, 5063, 5072, 5125, 5159, 5223,	4992, 4998, 5051, 5077, 5080, 5083,
5244, 5277, 5412, 5432, 5715, 6022,	5106, 5107, 5113, 5122, 5127, 5166,
6182, 6560, 6575, 6588, 6661, 6695,	5167, 5171, 5224, 5226, 5234, 5236,
6800, 6848, 7179, 7322, 7463, 7512,	5239, 5245, 5247, 5250, 5262, 5263,
7526, 8270, 8790, 8799, 9889, 9913	5278, 5279, 5281, 5282, 5448, 5454,
\tl_if_empty_p:N 288	5467, 5473, 5476, 5479, 5501, 5523,
\tl_if_eq:NNTF .. 974, 1878, 4777,	5787, 5788, 5796, 5828, 5830, 5837,
5008, 5026, 7219, 9852, 9853, 9862	5873, 5885, 5887, 5908, 5940, 6031,
\tl_if_eq:NnTF 8900,	6032, 6048, 6049, 6063, 6064, 6077,
9103, 9106, 9124, 9129, 10033,	6078, 6091, 6092, 6141, 6163, 6164,
10038, 10055, 10075, 10080, 10097	6326, 6357, 6366, 6390, 6406, 6409,
\tl_if_eq:nnTF .. 2519, 3210, 3218,	6418, 6421, 6427, 6434, 6453, 6556,
4723, 5763, 5769, 5824, 5862, 6923	6558, 6561, 6563, 6757, 6764, 6770,
\tl_if_exist:NNTF 307,	6809, 6823, 6824, 6836, 6875, 6888,
334, 347, 509, 548, 560, 671, 902,	6900, 6920, 6946, 6968, 6969, 7006,
1042, 1095, 1099, 1109, 1221, 1360,	7007, 7049, 7063, 7168, 7197, 7220,
1535, 1545, 1852, 2191, 2554, 2558,	7390, 7397, 7755, 8119, 8125, 8141,
2562, 2575, 6995, 7000, 8414, 10017	8383, 8384, 8388, 8389, 8393, 8394,
\tl_if_in:NnTF 2765, 2769, 2787	8397, 8398, 8401, 8402, 8405, 8406,
\tl_item:nn 5755	8409, 8410, 9025, 9097, 9141, 9172,
\tl_map_inline:nn 314, 318	9173, 9199, 9239, 9540, 9604, 9635,
\tl_new:N 1477, 1480,	9640, 9655, 9713, 9747, 9773, 9774,
2019, 2302, 2539, 2634, 2639, 2657,	9789, 9851, 10337, 10343, 10344,
2658, 2672, 3032, 4596, 4644, 4645	10398, 10399, 10400, 10406, 10471
\tl_put_left:Nn 4668, 5536, 6838, 6839, 9070, 10293	\tl_set_eq:NN 68, 253, 364, 366, 371,
\tl_put_right:Nn 1084, 3329, 3333, 3961,	923, 1059, 1070, 1072, 1234, 1351,
3962, 4650, 4667, 5073, 5171, 5225,	1926, 2341, 2384, 2416, 2444, 2593,
5242, 5243, 5253, 5254, 5267, 5513,	2844, 3049, 3238, 3796, 3797, 3798,
6470, 6476, 6488, 6497, 6773, 6778,	3799, 3827, 3828, 3829, 3830, 3886,
	3901, 4330, 4339, 4479, 4498, 4924,
	4996, 5086, 5111, 5388, 5777, 5835,

6140, 6344, 6411, 6423, 6447, 6593, 6595, 6599, 6625, 6627, 6769, 7300, 7465, 7514, 7590, 9096, 9101, 9102	
\tl_set_rescan:Nnn 466	
\tl_tail:n 2736, 4887, 5739	
\tl_to_str:n 68, 74, 104, 107, 128, 131, 143, 166, 173, 235, 315, 319, 320, 426, 439, 860, 934, 935, 1018, 1044, 1045, 1055, 1118, 1127, 1129, 1131, 1132, 1134, 1245, 1247, 1249, 1304, 1310, 1316, 1386, 1414, 1416, 1418, 1419, 1422, 1498, 1832, 1833, 1834, 1835, 1837, 2016, 2047, 2146, 2484, 2485, 2486, 2489, 2490, 2491, 2492, 2532, 2533, 2549, 2647, 2667, 2752, 2766, 2770, 2788, 2804, 2812, 3106, 3228, 3262, 3419, 3587, 3607, 3637, 4070, 4071, 4072, 4073, 4098, 4273, 4784, 4804, 4808, 4969, 5027, 5105, 5178, 5192, 5214, 5233, 5493, 5714, 5729, 6450, 6518, 6544, 6808, 6821, 6910, 6945, 7286, 8142, 8143, 8258, 8259, 9006, 9860, 9870, 9892, 9908, 9922, 9924, 9961, 10003	
\tl_trim_spaces:N 84	
\l_tmpa_tl 80, 81, 83, 1064, 1226, 1228, 1336, 1350, 1351, 2198, 2503, 2504, 4875, 5234, 5242, 5243, 5245, 5253, 5254, 5435, 5441, 5446, 5448, 5452, 5454, 5457, 5466, 5468, 5473, 5476, 5479, 5873, 5890, 5908, 5919, 6480, 6947, 6954, 6961, 7755, 7756, 9655, 9669, 9789, 9803, 9851, 9852, 9853, 9862, 10357, 10361, 10369	
\l_tmpb_tl 1333, 1341, 1345, 5435, 5441, 5443, 5457, 5467, 5471, 10358, 10362, 10370	
tmpc commands:	
\l_tmpc_tl 10359, 10363, 10371	
\to 8202, 8203, 8215	
\today 8818, 10461	
\TODO 4862, 8898	
token commands:	
\c_math_subscript_token 43, 89, 4430, 5940, 6740, 6741, 6744, 6745, 6749, 8183, 8205, 8207	
topsect 60	
\topsep 7824, 9212, 9486, 9505, 9533, 9698, 10115, 10155, 10232	
\trueFfalseT 9827	
\trueTfalseF 9822	
\type 76, 119	
	U
	\underbrace 6202, 6204, 8209
	\univ 56
	\unless 8481
	\uppercase 149, 1988, 6042
	\url 1451
	use commands:
	\use:N 375, 506, 510, 512, 544, 549, 551, 558, 561, 563, 889, 896, 1033, 1594, 1596, 1853, 1854, 1902, 1995, 2581, 3873, 7710, 7718, 7734, 7796, 7814, 7911, 7924, 7935, 7937, 7966, 7975, 8024, 8041, 8415, 8421
	\use:n 329, 2137, 2222, 2238, 2253, 4049, 4406, 4565, 4587, 4978, 5093, 5503, 5539
	\use:nn 93, 304, 474, 477, 889, 896, 1596, 1903, 1996, 2530, 3055, 3172, 3270, 3834, 3867, 3935, 4110, 4343, 4742, 4765, 4769, 5045, 5119, 5215, 5549, 5739, 5790, 5880, 6053, 6126, 6328, 6348, 6681, 6711, 7053, 7159, 7160, 7383, 7494, 7528, 7564, 7569, 7576, 7579, 7990, 8876, 9241, 10000, 10339
	\use_i:nn 3172, 5093, 10362
	\use_i:nnn 882, 1252
	\use_i:nnnn 1428
	\use_i:nnnnn 1138
	\use_ii:nn 3160, 3178, 5136
	\use_ii:nnn 885, 889, 1255, 1258
	\use_ii:nnnn 1431
	\use_ii:nnnnn 1141
	\use_iii:nnn 892, 896, 1261, 1265
	\use_iii:nnnnn 1144
	\use_iv:nnnn 1444, 1447
	\use_iv:nnnnn 1147
	\use_none:n 7851, 7856
	\use_none:nn 7170
	\use_v:nnnnn 1150
	\usebox 8671
	\usemodule . 16, 21, 22, 25, 35, 43, 95, 96, 98, 124, 128, 145, 416, 423, 2931
	\usepackage 7, 22, 2137, 8495
	\useSGvar 63, 112, 8891
	\usestructure 98, 156, 7231
	\usetHEME 8495
	\usetikzlibrary 82, 140, 2145, 10516
	V
	\varbind 41, 97, 102, 157, 7420, 7435, 7501
	\vardef 33, 41, 97, 98, 148, 4312, 4427, 7528, 7564, 7569, 7576, 7579, 7990
	\varempH 105, 151, 5951, 8952

<code>\Varname</code>	152, <u>6058</u>	<code>\vskip</code>	2080,
<code>\varname</code>	152, <u>6058</u>		2081, 8121, 8746, 8748, 9606, 9637
<code>\varnotation</code>	97, 153, <u>6333</u>	<code>\vspace</code>	9934
<code>\varref</code>	152, <u>6058</u>		
<code>\varseq</code>	38, 98, 149, 150, 4430, <u>4468</u>		W
<code>\vbox</code> 144, 1900, 1992, 2067, 2080, 7713,		<code>\wff</code>	19
7926, 8121, 8582, 8678, 8684, 8700,		<code>\withbrackets</code>	94, 155, <u>7004</u> , 7015
9281, 9341, 9389, 9440, 9606, 9637			
vbox commands:			X
<code>\vbox_set:Nn</code>	2715	<code>\xdef</code> 9979, 10031, 10072, 10119, 10159, 10273	
<code>\vdash</code>	8234	<code>\xspace</code>	
<code>\vfill</code>	8419, 9932, 9940, 10275		139, 1535, 1545, 3917, 3918, 4679, 4680
<code>\vM</code>	46		Y
<code>\vMs</code>	48	<code>\yesFnoT</code>	9817
<code>\vop</code>	39, 41	<code>\yesTnoF</code>	9812
<code>\vphantom</code>	9924	<code>\yield</code>	158, 7702, <u>8103</u>
<code>\vrule</code>	8121, 8747, 9606, 9637		