

The sTeX4 Package Collection*

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

2025-11-11

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	29
3.1	Simple Symbol Declarations	29
3.1.1	Semantic Macros and Notations	29
3.1.2	Types and Variables	32
3.1.3	Flexary Macros and Argument Modes	34
3.1.4	Precedences	36
3.1.5	Finishing Equality	36
3.1.6	Variable Sequences	37
3.2	Statements	37
3.2.1	Definitions	37
	Semantic Macros in Text Mode	38
	Definientia	39

*Version 4.0.0 (last revised 2025-11-11)

	Using Symbols Without Semantic Macros and Exporting Code in Modules	41
3.2.2	Assertions	42
3.2.3	Proofs	43
3.3	Mathematical Structures	43
3.3.1	Declaring and Using Structures	44
	Instantiating Structures	45
3.3.2	Extending Structures and Axioms	46
	Conservative Extensions	47
3.3.3	Nesting Structures and <code>\this</code>	49
3.4	Complex Inheritance and Theory Morphisms	51
3.4.1	Glueing Structures Together	53
3.4.2	Realizations	54
4	Extensions for Education	57
4.1	Slides and Course Notes	57
4.2	Problems and Exercises	57
4.2.1	Background	57
4.2.2	The Package, Options, and Configuration	58
4.2.3	(Open) Problems	59
4.2.4	Solutions and Answer Classes	60
4.2.5	Grading Support	61
4.2.6	Structured Problems	62
4.2.7	Single/Multiple Choice Blocks	63
4.2.8	Filling-In Concrete Solutions	66
4.2.9	Including Problems	68
4.2.10	Testing and Spacing	68
4.3	Homework Assignments and Exams	69
4.3.1	Introduction	69
4.3.2	Package Options	69
4.3.3	Assignments	69
4.3.4	Including Assignments	69
4.3.5	Typesetting Exams	70
II	User Manual	71
5	Basics	72
5.1	Package and Class Options	72
5.2	Math Archives and the <code>MathHub</code> Directory	73
5.2.1	The Structure of Math Archives	74
5.2.2	MANIFEST.MF-Files	74
5.3	The <code>lib</code> -Directory	75
5.4	Basic Macros	76
6	Document Features	77
6.1	Document Fragments	77
6.2	Using and Referencing Document Fragments	78
6.3	Cross-Document References	78

7	Modules and Symbols	81
7.1	Modules	81
7.1.1	Signature Modules, Languages, and Multilinguality	82
7.2	Symbol Declarations	82
7.2.1	Returns	83
7.3	Referencing Symbols	83
7.4	Notations and Semantic Macros	85
7.4.1	Precedences and Bracketing	86
7.4.2	Notations for Argument Sequences	87
7.4.3	Semantic Macros	87
7.5	Simple Inheritance	88
7.6	Variables and Sequences	90
7.7	Structures	91
7.7.1	Semantic Macros for Structures	91
8	Statements	94
8.1	More on Definitions	95
8.2	More on Assertions	96
9	Customizing Typesetting	97
9.1	Highlighting Symbol References	97
9.2	Styling Environments and Macros	98
9.3	Custom CSS for Environments	99
10	Additional Packages	100
10.1	NotesSlides Manual	100
10.1.1	Introduction	100
10.1.2	Package Options	100
10.1.3	Notes and Slides	101
10.1.4	Customizing Header and Footer Lines	102
10.1.5	Frame Images	102
10.1.6	Ending Documents Prematurely	103
10.1.7	Global Document Variables	103
10.1.8	Excursions	104
10.2	Problem Manual	105
10.2.1	Introduction	105
10.2.2	Problems and Solutions	105
10.2.3	Markup for Added-Value Services	107
	Multiple Choice Blocks	107
	Filling-In Concrete Solutions	108
10.2.4	Including Problems	109
10.2.5	Testing and Spacing	110
10.3	HWExam Manual	110
10.3.1	Introduction	110
10.3.2	Package Options	110
10.3.3	Assignments	111
10.3.4	Including Assignments	111
10.3.5	Typesetting Exams	111
10.4	Tikzinput Manual	112

III	Documentation	114
11	Utilities	115
11.1	kpsewhich and Environment Variables	115
11.2	Logging	116
11.3	File Paths	116
11.4	Group-like Behaviours	117
11.5	Key Handling	118
11.6	Languages	119
11.7	Styling	119
11.8	Persisting Dependencies	119
12	HTML Output	120
13	Math Archives	124
14	URIs	126
14.1	Document URIs	126
14.2	Module URIs	127
14.3	Symbol URIs	128
15	Documents	130
15.1	Sectioning	131
15.2	Inter-Document References	132
15.3	Inputting From MathHub Resources	132
16	Modules	134
16.1	SMS Mode	135
16.2	Inheritance and Morphisms	137
16.3	Theory Morphisms	137
17	Symbols	139
17.1	Declarations	139
17.2	Retrieval	140
17.3	Variables	140
17.3.1	Variable Sequences	141
17.4	Expressions	141
17.4.1	Term Annotations	142
17.4.2	Symbol Arguments	143
17.4.3	Notation components	143
17.4.4	Symbol References	144
17.5	Checking	144
18	Notations	145
18.1	Automated Bracketing	147
19	Structures	148
20	Statements	149
21	Proofs	150

22 Metatheory	152
23 Others	153
24 Additional Packages	154
24.1 NotesSlides Documentation	154
24.2 Problem Documentation	154
24.3 HWExam Documentation	154
24.4 Tikzinput Documentation	154
IV Implementation	155
25 Setting up	156
26 Utilities	158
26.1 kpsewhich and Environment Variables	159
26.2 Logging	160
26.3 File Paths	161
26.4 Group-like Behaviours	165
26.5 Key Handling	166
26.6 Languages	168
26.7 Styling	169
26.8 Persisting Dependencies	171
26.9 Auxiliary Methods	172
27 HTML Output	174
28 Math Archives	179
29 URIs	186
29.1 Document URIs	186
29.2 Module URIs	189
29.3 Symbol URIs	193
30 Documents	195
30.1 Sectioning	197
30.2 Inter-Document References	202
30.3 Inputting From MathHub Resources	207
31 Modules	213
31.1 SMS Mode	221
31.2 Inheritance and Morphisms	225
31.3 Theory Morphisms	229

32 Symbols	244
32.1 Declarations	244
32.2 Retrieval	252
32.3 Variables	256
32.3.1 Variable Sequences	259
32.4 Expressions	261
32.4.1 Term Annotations	281
32.4.2 Symbol Arguments	283
32.4.3 Notation components	288
32.4.4 Symbol References	290
32.5 Checking	294
33 Notations	296
33.1 Automated Bracketing	311
34 Structures	313
35 Statements	318
36 Proofs	325
37 Metatheory	335
38 Others	337
39 Additional Packages	338
39.1 Implementation: The notesslides Package	338
39.1.1 Class and Package Options	338
39.1.2 Notes and Slides	342
39.1.3 Environment and Macro Patches	346
39.1.4 Styling Across Notes/Slides	347
39.1.5 Beamer Compatibility	347
39.1.6 TODO Excursions	349
39.2 Implementation: The problem Package	350
39.2.1 Package Options	350
39.2.2 Problems and Solutions	352
39.3 Implementation: The hwexam Package	371
39.3.1 Package Options	371
39.3.2 QR Codes	371
39.3.3 Assignments	376
39.3.4 Leftovers	381
39.4 Tikzinput Implementation	381
Index	383



STEX is – by now – relatively stable and ready to use for the general public. However, it is also actively being developed further and subject to ongoing research. Some of the features described in here might not fully work as expected, some



are still experimental, there might occasionally be cryptic error messages, and in general bugs are expected.

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

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

Chapter 1

Introduction & Setup

1.1 What is sTeX?

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 pdf_lat_ex over sTeX-annotated documents formats them into normal-looking PDF.

But sTeX also comes with a conversion pipeline into semantically annotated HTML (FTML), which can host semantic added-value services that make the documents *active* (i.e. interactive and user-adaptive) – essentially turning L^AT_EX into a document format for (mathematical) knowledge management (MKM).

The sTeX system consists of the following components:

- The sTeX package,
- the Ru_STeX system for converting L^AT_EX documents to (semantically enriched) FTML,
- the FLM_f system; a semantically informed back-end for managing and hosting the FTML with integrated added-value services,
- a collection of math archives of sTeX document fragments and symbols for reuse, available of mathhub.info, and
- the FLM_f IDE: A VS Code extension that, besides the usual IDE functionalities like syntax highlighting, integrates FLM_f and allows for accessing the public math archives on mathhub.info to make the entire toolchain easily accessible to authors.

If PDF is all you are interested in, the sTeX package is all you *need*. Either way, however, we recommend using the sTeX IDE – it very much helps with semantically annotating your L^AT_EX documents.

1.2 The $\text{\texttt{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$ package

The $\text{\texttt{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$ package extends $\text{\texttt{L}}\text{\textsf{A}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$ 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 $\text{\texttt{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$ 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 $\text{\texttt{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$, it is strongly encouraged to follow a workflow of small document fragments and **modules** to maximize reuse.

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

$\text{\texttt{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$ therefore adds an organizational layer on top of $\text{\texttt{L}}\text{\textsf{A}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$ ’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 $\text{\texttt{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$ 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 $\text{\texttt{L}}\text{\textsf{M}}\text{\textsf{J}}$ 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 $\text{\texttt{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$ Manual.

1.4 Setting Up the **F $\text{\texttt{L}}\text{\textsf{M}}\text{\textsf{J}}$ IDE**

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

Setting up $\text{\texttt{S}}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$ with the dedicated **IDE** is easy:

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

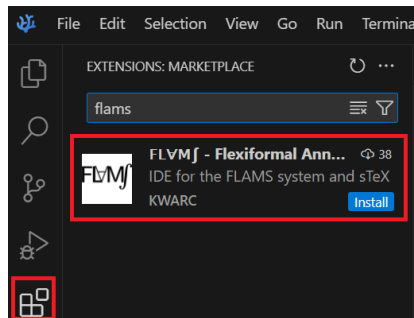


Figure 1: Installing the FLVMj IDE

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://stexmt.mathhub.info/sTeX/fullhtml?archive=sTeX/Documentation&filepath=tutorial.en.xhtml>*

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

In this part, we will give a broad but shallow introduction to [sTeX](#), and what you can get out of it. Additionally, this serves as an introduction to the [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^ATeX](#)-version of the above document fragment, without any [sTeX](#) markup, could look like this:

Example 1

Input:

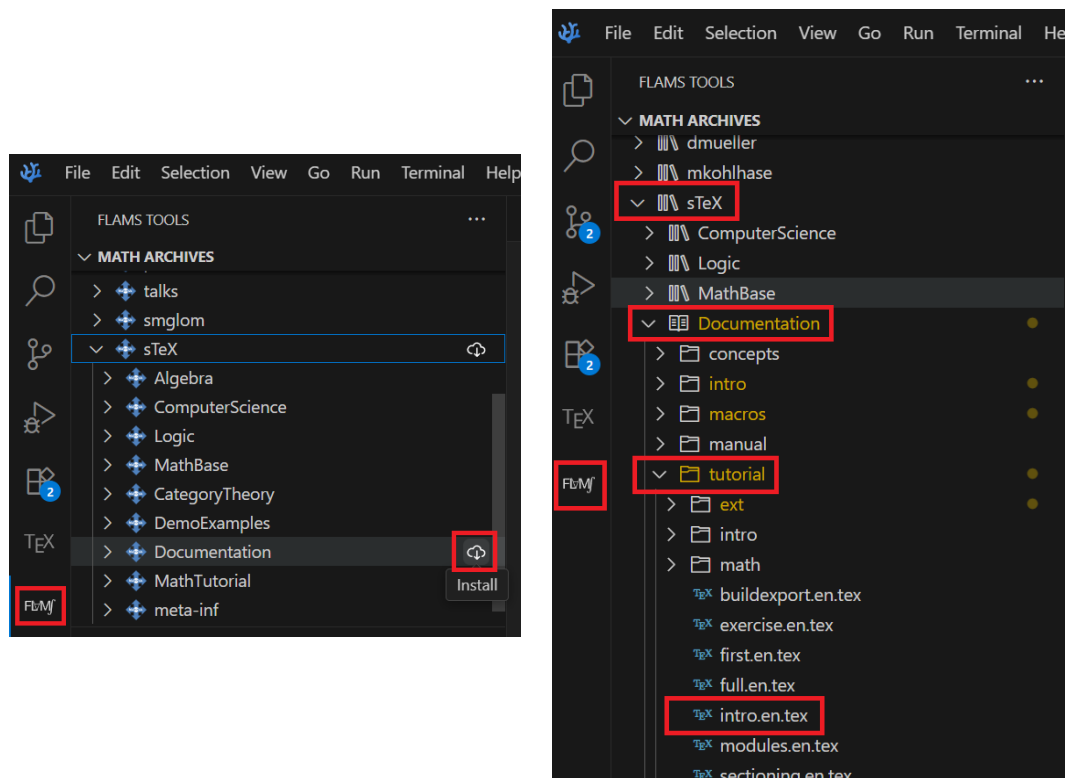


Figure 2: Installing Math Archives

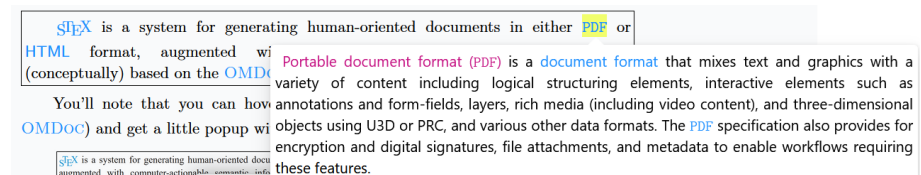


Figure 3: Definition on Hover

```

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).

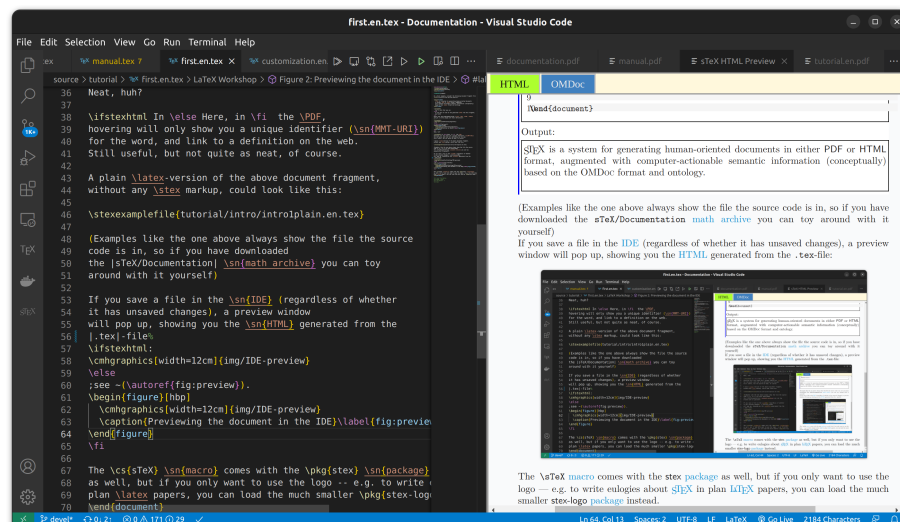


Figure 4: Previewing the document in the IDE

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 \LaTeX papers, you can load the much smaller `stex-logo` package instead.

2.1 Text symbols

The most central concept behind `TeX` 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, **TeX** is all about *reuse*, and naturally, there are **TeX symbols** for all of the above already. Let's start with the one for **TeX** 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).

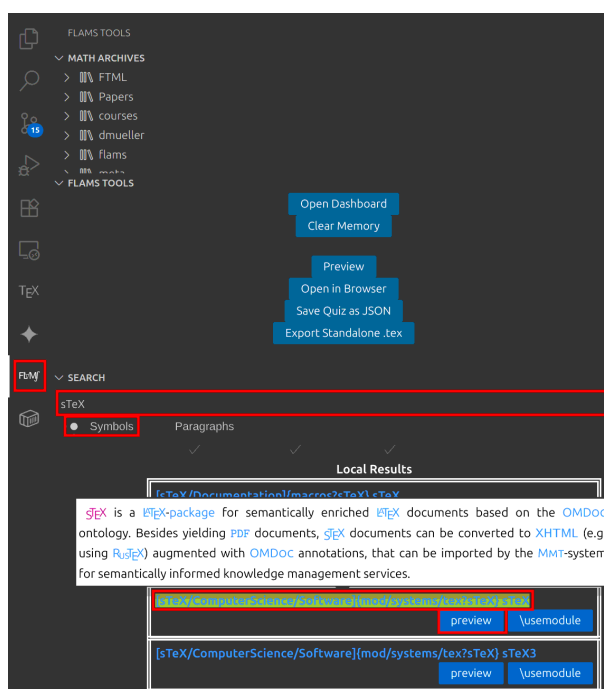


Figure 5: Search in the `STEX` IDE

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).

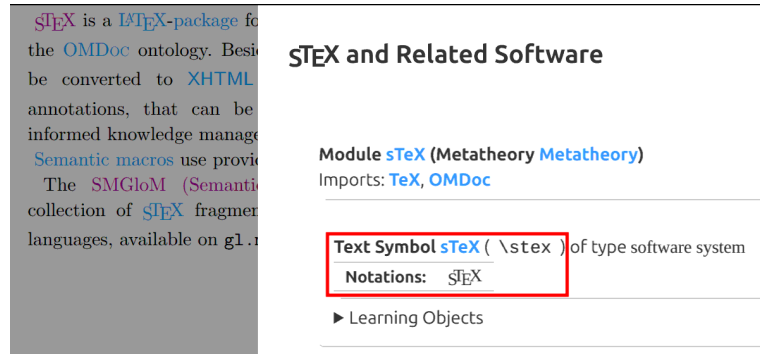


Figure 6: OMDoc Preview

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`”.

sTeX

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

1. the *name* of the *symbol*,

2. the name of its [semantic macro](#),
3. or any suffix of its [MMT-URI](#) containing at least the [module](#) name.

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 [L^AT_EX](#), that allows for semantically marking up document fragments; in particular concepts, [formulae](#) and [mathematical](#) statements (such as definitions, theorems and proofs). Running `pdflatex` over [STEX](#)-annotated documents formats them into normal-looking [PDF](#).

There’s only one problem: *the document does not compile*, with an error `Undefined control sequence`. The reason being that *some* [macro](#) in the [module](#) `Formula` uses the `\text` [macro](#). We can fix that by using the [amsfonts package](#) of course, but this points to a more general problem; namely that [modules](#) can make use of various [L^AT_EX packages](#) for typesetting [symbols](#).

Good practice suggests putting those packages into a *prelude* per [math archive](#), which we can then import from anywhere, using the `\libinput` [macro](#). For more on that, see section 5.3.

For now, suffice it to say that we can import all [packages](#) required for the [module](#) `Formula` from the [math archive](#) `sTeX/Logic/General` by adding the line

```
\libinput[sTeX/Logic/General]{preamble}
```

before the `\begin{document}`.

2.3 Modules and Simple Symbol Declarations

Consider again the first two paragraphs of section 1.1:

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

At its core is the [\$\text{sTeX}\$ package](#) for [\$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}\$](#) , that allows for semantically marking up document fragments; in particular concepts, [formulae](#) and [mathematical](#) statements (such as definitions, theorems and proofs). Running `pdflatex` over sTeX -annotated documents formats them into normal-looking [PDF](#).

Firstly, note that the first paragraph would be perfectly suitable to serve as a pop-up definition on hover for the [\$\text{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

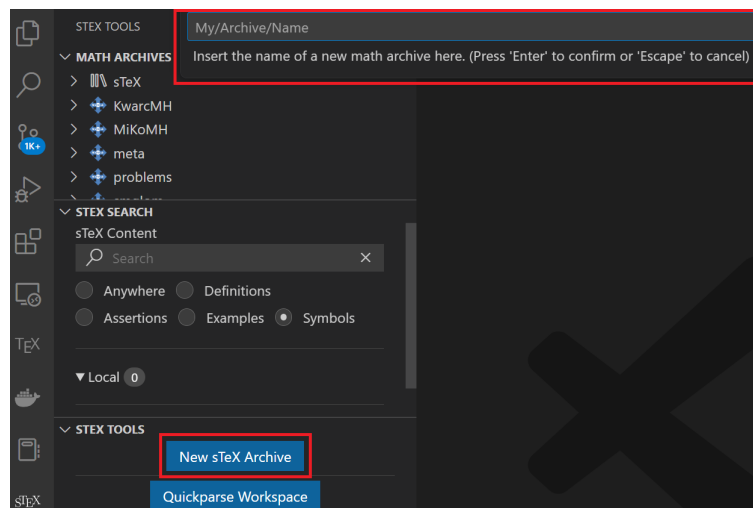


Figure 7: New [Math Archive](#) in the [IDE](#)

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:

T_EX 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.

L^AT_EX is a (nowadays) default collection of **T_EX macros** developed by Leslie Lamport. Among other things, **L^AT_EX** 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: **T_EX**, **macro**, **L^AT_EX**, **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}

```

F_EM_f 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 > latex.tex > {} https://mathhub.info?a=my/archive&p=latex&m=l
1 \documentclass{stex}
2 \begin{document}
3   https://mathhub.info?a=my/archive&p=latex&m=LaTeX
4 \begin{module}{LaTeX}
5   ...
6 \end{module}
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 > LaTeX.en.tex > ...
1 \documentclass{stex}
2 \begin{document}
3   https://mathhub.info?a=my/archive&m=LaTeX
4 \begin{module}{LaTeX}
5   ...
6 \end{module}
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.

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 `sTeX` 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 `sTeX` know that that is the case:

Within an `sparagraph` environment with `style=symdoc` (or an `sdefinition` environment), we can mark up *definienda*, meaning the terms *being defined*, explicitly. Analogously to `\symname` and `\symref`, we have the `macros` `\definame` and `\definiendum` for that purpose.

Note that the `\tex` `macro` induced by the `text` `symbol` above already marks up the “`TeX`” it produces, so wrapping it in another `\definiendum` would be redundant. However, every `text` `symbol` also generates a *second* `macro` with the suffix `name` that generates a non-marked-up version of the same presentation. In other words, we get the `macro` `\texname` for free, that produces “`TeX`” (of course, we could just as well use the `\TeX` `macro`, but that one you probably know already).

Furthermore, every `\definiendum` or `\definame` automatically adds the `symbol` being referenced to the internal `for=`-list of the `sparagraph` environment, obviating the need to list it explicitly.

As such, we can produce a better markup like this:

```
\begin{sparagraph}[style=symdoc]
\definiendum{tex}{\texname} is a document typesetting
software developed by Donald Knuth, with a focus on
mathematical formulae. It is based on a powerful
and extensible \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{}{\nauthor{Me}\title{The \texttt{my/archive} Archive}\maketitle\nifstexhtml \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.

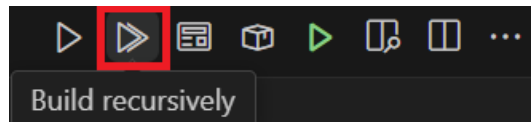


Figure 11: The Build `PDF/XHTML/OMDoc` Button

This will queue all dependencies of the file in a new build queue and open the `FVMf` dashboard for us (see Figure 12). Clicking on the `run` button in the dashboard will then for every queued file:

1. Run `pdflatex` twice,

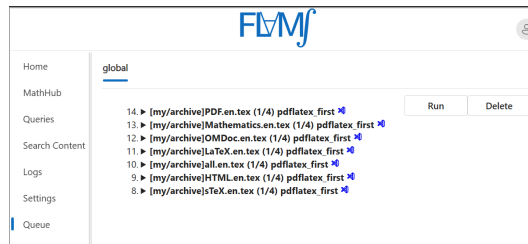


Figure 12: The FLAMf build queue

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

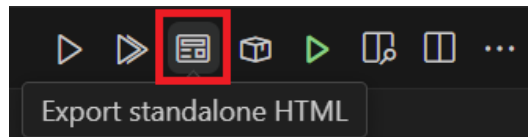


Figure 13: Exporting [HTML](#) in the [IDE](#)

In the dialog box that opens now, select an **empty** directory and [FLAMf](#) 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 \sTeX know that “`equal`” takes two arguments. We might also want to shorten the **semantic macro** to e.g. `\eq`, without changing the name. Hence:

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

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

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

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

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

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

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

STEX

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

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

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

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

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

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



You may be wondering about the role of the `\mathrel` macro in the example above: \TeX determines spacing/kerling 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 TeX-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** **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

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

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:

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

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



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

Now that we use imported symbols in types (and since we are in a module), we need to make sure that the imported modules are also (transitively) *exported*, since our new symbols now *depend* on the imported module.

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

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

Note that the HTML is aware of this now (after you save): Clicking on any occurrence of `addition` now yields Figure 15.

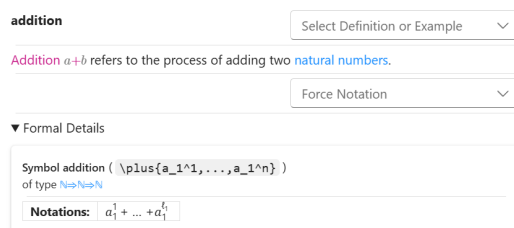


Figure 15: On-Click Popup in the HTML

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

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.

STEX

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**,

STEX

- a a – (left or right) associative – **sequence argument**, represented as a single TeX-argument $\{a, b, \dots\}$,
 - 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** $\{x, y, z, \dots\}$.
- 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 `\forallforal{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{spargraph}[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{spargraph}
```

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!

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 $a \cdot b \cdot c$.

²TODO: argmap

³TODO: argarraymap

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...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 `TeX` 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={\scriptstyle (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`.

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.

⁴TODO: seqmap

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 annotations!

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

Semantic Macros in Text Mode

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

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

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

Definientia

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

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

The `\definiens` macro is only allowed within the `sdefinition` environment, and requires that the `environment` lists the `symbol` that gets the definiens attached explicitly in its `for=` argument. It is possible to attach definiens 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 definiens 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 `gTeX` 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 definiens 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  \definame{associative}, if
19  \definiens{\foral{$\arg[2]{\eq{
20    \vop{\vop{va,vb}},vc},
21    \vop{va,(\vop{vb,vc})}
22  }}$ \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	$\{A: \text{SET}\}_I (\circ: A \rightarrow A \rightarrow A) \rightarrow \forall a: A, b: A, c: A. \left(\frac{(a \circ b) \circ c = a \circ (b \circ c)}{A} \right)$
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 [L^AT_EX](#) methods. This is where [\STEXexport](#) becomes very handy:

In a [module](#), we can put arbitrary [L^AT_EX](#) 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 [L^AT_EX](#) 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{ $\textcolor{teal}{plus}$ }` to produce “+ is [associative](#)”.



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

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

Exercise

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

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

TODO⁵

3.2.2 Assertions

Having defined [associativity](#) and [commutativity](#), we can now assert that both properties hold for [addition](#) and [multiplication](#).

For *assertions* (i.e. theorems, lemmata, axioms, claims,...), [sTeX](#) provides the [sassertion environment](#).

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

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

which yields

[Addition](#) is [associative](#)

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

So far, this is not too useful – after all, we could have just as well used e.g. the [amsthm package](#) and gone straight for the non-[sTeX](#) variant

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

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

⁵**TODO:** *intent?*

```

\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	assertion
Term term	associative (+)

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



[S_ITEX](#) 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 [S_TE_X](#), 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`. It’s 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:

```

7 \begin{mathstructure}{magma}
8 \syndef{univ}[name=universe,type=\collection]{U}
9 \syndef{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
\mathcal{M}	M
$\langle \mathcal{U} \rangle$	$\langle U_M, \circ_M \rangle$
\mathcal{U}	U_M
\mathcal{O}	\circ_M
$\mathcal{O}\{a, b, c\}$	$a \circ_M b \circ_M c$

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, sTeX attaches the **symbol's** (or **variable's**) **operator notation** as a subscript suffix to the notation component marked with `\maincomp` – e.g., since the “`\circ`” in the **notation** for `op` is marked with `\maincomp`, doing `\vM{op}{a,b}` ultimately outputs $a \circ_M b$. 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 $\mathcal{VM}! := \mathcal{VM}\{\}$ and $\mathcal{VMp}! := \mathcal{VMp}\{\}$ $\text{\textcolor{red}{sns}\textcolor{green}{magma}}$. Then...

yielding

Let $M := \langle U, \circ \rangle$ and $M' := \langle U', \circ' \rangle$ $\textcolor{teal}{magma}$ s. 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=\textcolor{green}{collection}]{A}
\vardef{vM}[name=M,return={\textcolor{green}{magma}[comp=##1][univ=\vA]}]{M}
\vardef{vMp}[name=Mp,return={\textcolor{green}{magma}[comp={\{##1\}'}][univ=\vA]}]{M'}

Let  $\mathcal{VM}! := \mathcal{VM}\{\}$  and  $\mathcal{VMp}! := \mathcal{VMp}\{\}$   $\text{\textcolor{red}{sns}\textcolor{green}{magma}}$  on  $\mathcal{VA}$ ....
```

Let $M := \langle A, \circ \rangle$ and $M' := \langle A, \circ' \rangle$ $\textcolor{teal}{magma}$ s.

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

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

Let $(M_i)_1^n := \langle U_i, \circ_i \rangle_1^n$ a sequence of $\textcolor{teal}{magma}$ s...

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

Also, note that if we want to produce M_i – i.e. the $\textcolor{teal}{magma}$ 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 $\text{\textcolor{teal}{s}\text{\textcolor{teal}{T}\text{\textcolor{teal}{E}\text{\textcolor{teal}{X}}}}$ to “stop eating arguments” and present whatever it has until now.

3.3.2 Extending $\textcolor{teal}{Structures}$ and Axioms

It is extremely common to “build up” $\textcolor{teal}{structures}$ 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 $\textcolor{teal}{structures}$ using `\importmodule{...?FooMod/foo-module}` – but extending $\textcolor{teal}{structures}$ 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 $\textcolor{teal}{structure}$ names. That allows us to easily define $\textcolor{teal}{semigroups}$:

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.

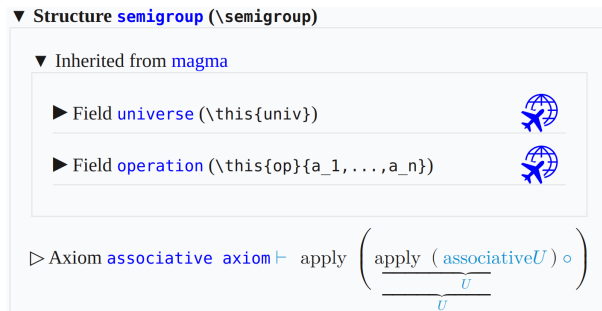


Figure 18: Axioms in **OMDoc**

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

Conservative Extensions

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

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.

STEX

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?

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

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

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

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

Consider the following conservative extension:

Example 13

Input:

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

...producing a^2 .

3.3.3 Nesting Structures and `\this`

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

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

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

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

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

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

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

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

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

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

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

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

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

Example 14

Input:

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

```

9 \begin{mathstructure}{\magmahom}{\magma homomorphism,this=F}
10 \symdef{dom}[name=domain,return={\magma[comp={##1}_1]}\{M_1\}
11 \symdef{cod}[name=codomain,return={\magma[comp={##1}_2]}\{M_2\}
12 \symdef{f}[op=\this,args=1,
13 type=\funspace{\dom{univ}}{\cod{univ}}
14 ]{\this \dobrackets{#1}}
15
16 \begin{sdefinition}[for={magmahom,dom,cod,f}]
17 \vardef{va}[name=a,type=\dom{univ}]{a}
18 \vardef{vb}[name=b,type=\dom{univ}]{b}
19 Let  $\dom!=\dom{\}$  and  $\cod!=\cod{\}$  \sns{magma}.
20 A \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
$\backslash\mathrm{vh}!\$$	H
$\backslash\mathrm{vh}\{\}\$$	$\langle M_1, M_2, H \rangle$
$\backslash\mathrm{vh}\{f\}!\$$	H
$\backslash\mathrm{vh}\{f\}\{a\}\$$	$H(a)$
$\backslash\mathrm{vh}\{\mathrm{dom}\}!\$$	M_1
$\backslash\mathrm{vh}\{\mathrm{cod}\}\{\}\$$	$\langle U_2, \circ_2 \rangle$
$\backslash\mathrm{vh}\{\mathrm{cod}\}\{\mathrm{univ}\}\$$	U_2
$\backslash\mathrm{vh}\{\mathrm{dom}\}\{\mathrm{op}\}!\$$	\circ_1
$\backslash\mathrm{vh}\{\mathrm{cod}\}\{\mathrm{op}\}\{a,b,c\}\$$	$a \circ_2 b \circ_2 c$

Note how – as one would expect – we can treat $\backslash\mathrm{vh}\{\mathrm{dom}\}$ and $\backslash\mathrm{vh}\{\mathrm{cod}\}$ 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 $\backslash\mathrm{this}$ 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 **TeX** is significantly more complicated *for now*, and will require some amount of research to make convenient – especially if we want to subsequently define *operators* on pairs of elements, as below. We therefore opt for the simpler version where it is defined as binary from the get go.

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

Definition 3.4.17. A poset $\langle U, R \rangle$ 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 `meetsl` and `joinsl`. In the first copy, we only want to rename `op` to `meet`. In the second, we want to rename `op` to `join`, and *also* redefine the universe to be the one from `meetsl`:

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

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

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

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

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

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

3.4.2 Realizations

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

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

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

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

Definition 3.4.21. Let $\langle U, \circ \rangle$ a *semilattice*. We let $a \textcolor{teal}{R} b$ 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* ($a \textcolor{teal}{R} a$):

2. *antisymmetry* ($a \textcolor{teal}{R} b$ and $b \textcolor{teal}{R} a$ 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 $a \textcolor{teal}{R} b$ and $b \textcolor{teal}{R} c$, then $a \textcolor{teal}{R} c$.):

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, $a \textcolor{teal}{R} c$.

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

By definition (and the *commutative axiom*), $a \circ b \textcolor{teal}{R} a$ and $a \circ b \textcolor{teal}{R} b$.

We need to show, that if $x \textcolor{teal}{R} a$ and $x \textcolor{teal}{R} b$, then $x \textcolor{teal}{R} a \circ b$.

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 $x \textcolor{teal}{R} a \circ b$

□

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 $\semilattice![univ,op]$ a \sn{semilattice}.
We let $\rel{\va,\vb}$
iff $\definiens{\eq{\op{\va,\vb},\va}}$.
\end{sdefinition}

```

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

Analogously, we can use the `sproof` and `subproof` environments to produce “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

`TODO`⁶

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

⁶`TODO: notesslides.sty`

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

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

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

- open problems, where expected answers are free-form, and classification into answer classes is usually a manual process; see subsection 4.2.3
- single/multiple choice questions where the answer classes and thus feedback/grading correspond to the selection pattern of items; see subsection 4.2.7.
- 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 document¹, these can be localized by in the files `ldf/problem-<lang>.ldf`⁷.

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:

¹EdNOTE: MK: document the attribute somewhere

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

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

```
1 \begin{document}
2 \begin{sproblem}[id=prob.elefants,pts=10,min=2,title=Fitting Elefants]
3   How many Elefants can you fit into a Volkswagen beetle?
4   \begin{hint}
5     Think positively, this is simple!
6   \end{hint}
7   \begin{exnote}
8     Justify your answer
9   \end{exnote}
10  \begin{gnote}[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.²

EdN:2

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}
```

²EDNOTE: 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 systems⁸. 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:

⁸like 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 L^AT_EX 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,autogradable]
3   What is the keyword to introduce a function definition in python?
4   \begin{mcb}
5     \mcc[T]{def} \mcc[F,feedback=that is for C and C++] {function}
6     \mcc[F,feedback=that is for Standard ML]{fun}
7     \mcc[F,Ftext=Noooooooooooo,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

^cNoooooooooooo

that is for Java

The problem has been marked as **autogradable** for semantic processing.

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,autogradable]
3   What is the keyword to introduce a function definition in python?
4   \begin{mcb}
5     \mcc[T]{def} \mcc[F,feedback=that is for C and C++] {function}
6     \mcc[F,feedback=that is for Standard ML]{fun}
7     \mcc[F,Ftext=Noooooooooooo,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.³

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:

³EdNOTE: 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

```
{⟨target⟩}{⟨TF⟩}{⟨feedback⟩}
```

where `⟨target⟩` is the target specification, `⟨TF⟩` is the verdict T or F on the correctness of an answer that is accepted by `⟨target⟩`, and `⟨feedback⟩` 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.

\testnewpage `\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

TODO⁹

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> <code>notes</code> <code>hints</code> <code>gnotes</code> <code>pts</code> <code>min</code> <hr/>	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>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 \LaTeX 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
<code>number</code>	— the ordinal of the <code>assignment</code> environment), <code>title</code> (for the assignment title; this is
<code>title</code>	referenced in the title of the assignment sheet), <code>type</code> (for the assignment type; e.g. “quiz”,
<code>type</code>	or “homework”), <code>given</code> (for the date the assignment was given), and <code>due</code> (for the date
<code>given</code>	the assignment is due).
<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.
---------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

⁹TODO: 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)

2025-11-11

You have one hour (sharp) for the test;

Write the solutions to the sheet.

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

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

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

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

good luck

10

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

Part II

User Manual

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

Chapter 5

Basics

5.1 Package and Class Options

- `debug=<prefixes>`: (see Developer Manual)
- `lang=<languages>`: If set, [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 in order for the `sTeX/Documentation` [archive](#) to be found by the `sTeX` system, it needs to be in `/user/foo/MathHub/sTeX/Documentation`.

Each such [archive](#) needs two subdirectories:

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

An additional `lib`-directory is optional, and is discussed in section 5.3.

5.2.2 MANIFEST.MF-Files

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

```
id: sTeX/Documentation
ns: http://mathhub.info/sTeX/Documentation
narration-base: http://mathhub.info/sTeX/Documentation
format: stex
title: The sTeX Documentation
teaser: The full Documentation for the sTeX system
url-base: https://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 $\text{\texttt{STeX}}$ 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 `graphicx` 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 `\sref` macro references `<label>` (declared in `<file1>` in `math archive <archive1>`). If the object (section, figure, etc.) with that label occurs (eventually) in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the `hyperref` package is not included.

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

In `HTML` mode, the reference additionally links to the `HTML` of the `file1`.¹¹

This works by storing labels during compilation in a file `<jobname>.sref`, analogous to e.g. the `.toc`. Note that this consequently requires both `file1.tex` and `file2.tex` to have been compiled previously, to generate the `.sref` file.

For example, doing

```
\sref[file=tutorial/full.en]{sec:basics}[file=tutorial.en,title=the \stex Tutorial]
```

in this very document fragment (`[sTeX/Documentation]macros/sref.en.tex`) will yield [Part I \(The Basics\) in the sTeX Tutorial](#) if compiled itself, or if compiled as part of the `sTeX` manual, and will yield the `\autoref` link [chapter 2](#) in the documentation (which includes the tutorial).

```
\srefsetin \srefsetin [<archive2>]{<file2>}{<title>}
```

Sets a default value for the optional arguments `<archive2>`, `<file2>` and `<title>` of `\sref`. If the second set of optional arguments in `\sref` are omitted, these default values are used. Particularly useful to set in a preamble.

```
\sreflabel \sreflabel {\<label>} sets a label analogous to \label{\<label>}, but for use in \sref.
```

Note that for every `sTeX` macro or environment that takes an optional `id=<id>` argument, the `<id>` (if non-empty) generates an `\sreflabel` automatically.

For example, `\begin{sfragment}[id=foo]{Foo}` is equivalent to `\begin{sfragment}{Foo}\sreflabel{foo}`.

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

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

Chapter 7

Modules and Symbols

7.1 Modules

A `module` is required to declare any new formal content such as `symbols` or `notations` (but not `variables`, which may be introduced anywhere).

`\M` → An `sTeX` `module` corresponds to an `MMT/OMDoc` *theory*. As such
`\M` → it gets assigned an `MMT-URI` (*universal resource identifier*) of the form
`\T` → `<namespace>?<module-name>`.

`smodule` (*env.*) A new module is declared using the basic syntax

`\begin{smodule}[options]{ModuleName}...\end{smodule}`.

A module is required to declare any new formal content such as `symbols` or `notations` (but not `variables`, which may be introduced anywhere).

The `smodule`-environment takes several keyword arguments, all of which are optional:

`title` (`<token list>`) to display in customizations.

`style` (`<string>*`) for use in customizations, see 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.

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

`\symdecl` takes the following optional arguments:

`name` see above,

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

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

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

`return` the `symbol`'s *return code* (see below), most commonly the `semantic macro` of a `mathematical structure`,
`assoc` how to resolve arguments of `mode` `a` or `B`; may be `pre`, `bin`, `binl`, `binr` or `conj`,
`reorder` how to reorder the arguments in `OMDoc` (*advanced*),
`role` `symbols` with certain roles are treated in particular ways in `MMT/OMDoc` (*advanced*),
`argtypes` `TODO`¹².

`\textsymdecl` `\textsymdecl{\langle mname \rangle}[\langle options \rangle]{\langle code \rangle}`

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 `\langle code \rangle` as output.

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

`\symdef` `\symdef{\langle mname \rangle}[\langle options \rangle]{\langle notation \rangle}`

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=\langle code \rangle`. 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. `TeX` will replace `#1` and `#2` in `\langle code \rangle` by `a` and `b`, yielding `\langle retcode \rangle`.
2. `TeX` will insert `\langle retcode \rangle{\foo{a}{b}!}` in the input token stream.

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

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{\langle symbol \rangle}{\langle text \rangle}`

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

¹²`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` `\infprec` 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**¹³

7.4.3 Semantic Macros

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

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

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

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

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

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

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

By default, they are numbered consecutively; e.g. `\smacro{\dots\arg{a}\dots\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{\dots\arg[2]{a}\dots\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 \hookrightarrow `MMT/OMDoc` as OMA-terms with head `<OMS name="⟨symbol⟩"/>`, or \hookrightarrow `<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{\langle module \rangle}` is allowed anywhere and makes the `module`'s contents available up to the current `TeX` group. This is the right `macro` to use outside of `modules`, or when none of its contents use any of the used `module`'s `symbols` directly (e.g. in `types` or `definientia`).

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

\importmodule `\importmodule{\langle module \rangle}` is only allowed in `modules` and makes the required `module`'s contents available within the current `module`. The imported `symbols` can be safely used anywhere, and the imported content exported to any `modules` subsequently importing the current one.

\hookrightarrow In `MMT`, every *document* and every `module` induces an `MMT` theory. `\usemodule`
 \hookrightarrow induces and `MMT` `include` in the document theory, `\importmodule` and
 \hookrightarrow `\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, `TeX` 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 `TeX` 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 `TeX` 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.

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

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

More precisely: `!\struct{<code>}` 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 `\TeX` does when using `\struct` in the *return* of a *symbol* (or *variable*).

Continuing:

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

Note the insertion of `!` behind the `A` – this is to make sure that assignments to *semantic macros* that takes arguments don’t accidentally eat more than they should.

Also note that multiple assignments can be done in the same pair of `[]`, or chained – i.e. both `!\struct[fielda=A,fieldb=B]...$` and `!\struct[fielda=A][fieldb=B]...$`

are valid and equivalent. `$\textcolor{green}{struct}[fielda=A,fielda=B] \dots$` 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> is to <code>\definiendum</code> as <code>\symname</code> is to <code>\symref</code> . Analogously, <code>\Definame</code> behaves like <code>\Symname</code> . <code>\defnotation</code> can be used in math mode to apply the <code>\definiendum</code> highlighting to <i>notations</i> .
<code>\definame</code>	
<code>\Definame</code>	
<code>\defnotation</code>	

<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).
 \hookrightarrow 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`.
 \hookrightarrow 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/OM-DOC 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.

\hookrightarrow M \rightarrow
 \hookrightarrow M \rightarrow
 \hookrightarrow T \rightarrow

$\text{sproof}(\text{env.})$ TODO¹⁴

¹⁴TODO: 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 *styable* using the *macros* `\stexstyle<name>[<style>]{...}`. These styable *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}}
```

<code>\stexstylemodule</code>	TODO ¹⁵
<code>\stexstylecopymodule</code>	
<code>\stexstyleinterpretmodule</code>	
<code>\stexstylerealization</code>	
<code>\stexstylemathstructure</code>	
<code>\stexstyleextstructure</code>	
<code>\stexstyledefinition</code>	
<code>\stexstyleassertion</code>	
<code>\stexstyleexample</code>	
<code>\stexstyleparagraph</code>	
<code>\stexstyleproof</code>	
<code>\stexstylesubproof</code>	

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.

<code>\stexstyleusemodule</code>	TODO ¹⁶
<code>\stexstyleimportmodule</code>	
<code>\stexstylerequiremodule</code>	
<code>\stexstyleassign</code>	
<code>\stexstylerenamedecl</code>	
<code>\stexstyleassignMorphism</code>	
<code>\stexstylecopymod</code>	
<code>\stexstyleinterpretmod</code>	
<code>\stexstylerealize</code>	
<code>\stexstylesymdecl</code>	
<code>\stexstyletextsymdecl</code>	
<code>\stexstylenotation</code>	
<code>\stexstylevarnotation</code>	
<code>\stexstylesymdef</code>	
<code>\stexstylevardef</code>	
<code>\stexstylevarseq</code>	
<code>\stexstylepsfsketch</code>	
<code>\stexstyleMMTinclude</code>	

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 10.1.3).
-------------------------------------------	---------------------------------------------------------------------------------------------------------------------------

<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.



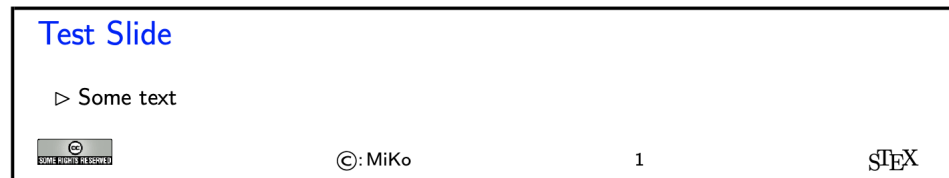
The **beamer** class recommends not to use the **allowframebreaks** option on frames (even though it is very convenient). This holds even more in the **notesslides** case: At least in conjunction with **\newpage**, frame numbering behaves funnily (we have tried to fix this, but who knows).

\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>\setSGvar{⟨vname⟩}{⟨text⟩}</code> to set the global variable <code>⟨vname⟩</code> to <code>⟨text⟩</code> and <code>\useSGvar{⟨vname⟩}</code>
<code>\useSGvar</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>	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.
<code>\printexcursion</code>	
<code>\excursionref</code>	

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

`solutions`
`notes`
`hints`
`gnotes`
`pts`
`min`
`boxed`
`test`

The `problem` package takes the options `solutions` (should solutions be output?), `notes` (should the problem notes be presented?), `hints` (do we give the hints?), `gnotes` (do we show grading notes?), `pts` (do we display the points awarded for solving the problem?), `min` (do we display the estimated minutes for problem solving). If theses are specified, then the corresponding auxiliary parts of the problems are output, otherwise, they remain invisible.

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

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

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

`\startsolutions` Sometimes we would like to locally override the `solutions` option we have given to
`\stopsolutions` 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.¹⁷

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.

¹⁷For 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)
2025-11-11

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

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

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

good luck

18
Local Variables: mode: latex TeX-master: ../stex-manual End:
LocalWords: hwexam solutions,notes,hints,gnotes,pts,min gnotes testemptypage 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

¹⁸MK: 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` \star **`\stex_str_if_starts_with:nn`** $\{\langle str1 \rangle\}$ $\{\langle str2 \rangle\}$
`\stex_str_if_starts_with:nnTF` \star

Tests if $\langle str1 \rangle$ starts with $\langle str2 \rangle$.

`\stex_str_if_ends_with_p:nn` \star **`\stex_str_if_ends_with:nn`** $\{\langle str1 \rangle\}$ $\{\langle str2 \rangle\}$
`\stex_str_if_ends_with:nnTF` \star

Tests if $\langle str1 \rangle$ ends with $\langle str2 \rangle$.

`\stex_deactivate_macro:Nn` **`\stex_deactivate_macro:Nn`** $\backslash macro$ $\{\langle scope \rangle\}$

`\stex_reactivate_macro:N` redefines $\backslash macro$ to throw an error, that the $\backslash macro$ is only allowed in $\langle scope \rangle$. This allows for more informative error messages than `undefined control sequence`.

$\backslash stex_reactivate_macro:N$ makes the macro valid again.

11.1 kpsewhich and Environment Variables

`\stex_kpsewhich:Nn` **`\stex_kpsewhich:Nn`** $\backslash macro$ $\{\langle args \rangle\}$

Calls “`kpsewhich` $\langle args \rangle$ ” and stores the result in $\backslash macro$,

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

<hr/> <hr/>	<code>\stex_get_env:Nn \stex_get_env:Nn \macro {<envvar>}</code>
	Stores the value of the environment variable <code><envvar></code> in <code>\macro</code> .

11.2 Logging

<hr/> <hr/>	<code>\stex_debug:nn \stex_debug:nn {<prefix>} {<msg>}</code>
	Logs the debug message <code>{<msg>}</code> under the prefix <code>{<prefix>}</code> . A message is shown if its prefix is in a list of prefixes given either via the package option <code>debug=<prefixes></code> or the environment variable <code>STEX_DEBUG=<prefixes></code> , 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.

<hr/> <hr/>	<code>\stex_file_set:Nn \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> .

<hr/> <hr/>	<code>\stex_file_resolve:Nn \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> .

<hr/> <hr/>	<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).

<hr/> <hr/>	<code>\stex_file_split_off_ext:NN \stex_file_split_off_ext:NN \target \source</code>
	splits off the file extension of <code>\source</code> and stores the resulting file path in <code>\target</code> .

<hr/> <hr/>	<code>\stex_file_split_off_lang:NN \stex_file_split_off_lang:NN \target \source</code>
	splits off <i>both</i> the file extension <i>and</i> language component of <code>\source</code> and stores the resulting file path in <code>\target</code> ; e.g. if <code>\source</code> is <code>foo.en.tex</code> , <code>\target</code> will be <code>foo</code> . Explicitly checks that the segment before the file extension is a valid language identifier; otherwise, it will not be stripped.

<hr/> <hr/>	<code>\c_stex_pwd_file</code>
<code>\c_stex_main_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:NTF` `*`

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

`\stex_if_file_starts_with:NNTF` `\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.

TeXhackers 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` $\langle\textit{macroname}\rangle$ $\langle\textit{argument spec}\rangle$ $\langle\textit{definition}\rangle$
`\stex_style_apply:` $\langle\textit{default}\rangle$

Defines a new stylable command `\macroname` with $\langle\textit{argument spec}\rangle$ (passed on to `\NewDocumentCommand`) with $\langle\textit{definition}\rangle$ as its macro body and default typesetting style $\langle\textit{default}\rangle$.

$\langle\textit{default}\rangle$ may be empty. $\langle\textit{definition}\rangle$ 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 $\langle\textit{default}\rangle$.

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

`\stex_new_stylable_env:nnnnnnn` `\stex_new_stylable_env:nnnnnnn` $\langle\textit{environment name}\rangle$ $\langle\textit{argument spec}\rangle$ $\langle\textit{begin code}\rangle$ $\langle\textit{end code}\rangle$ $\langle\textit{default begin}\rangle$ $\langle\textit{default end}\rangle$ $\langle\textit{prefix}\rangle$

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

$\langle\textit{begin code}\rangle$ and $\langle\textit{end code}\rangle$ 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[\langle\textit{name}\rangle]\langle\textit{begin}\rangle\langle\textit{end}\rangle` 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`

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

`\c_stex_mathhub_files` represents the comma-separated, absolute path *strings* of the user’s MathHub directories.
`\mathhub` If the `\mathhub` macro is set when the `stex` package is loaded, this one is used for finding archives. Otherwise it is determined from the `MATHHUB` environment variable, if set, or from the path in `<HOME>/stex/mathhub.path`, if existent, or set to `<HOME>/MathHub` if all else fails. In all cases, `\mathhub` will be defined.

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.

`\stex_archive_id:N` * expands to the id of the given archive (a macro defined as a triple as described above)

`\stex_archive_base:N` * expands to the base URI of the given archive; either given as a macro defined as a triple
`\stex_archive_base:n` * as described above, or as the archive’s id. The latter requires the archive to be loaded beforehand!

`\stex_archive_path:N` * expands to the file path *as a string* of the given archive; either given as a macro defined
`\stex_archive_path:n` * as a triple as described above, or as the archive’s id. The latter requires the archive to be loaded beforehand!

`\stex_in_archive:nn` `\stex_in_archive:nn {<archive id>} {<code>}`

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> <code>{\langle relative path \rangle}</code>
<code>\stex_source_path:nn</code>	<code>\stex_source_path:nn</code> expands to the absolute file path (string) of the given <code>\langle relative path \rangle</code> 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 *`

<code>\stex_module_uri_name:N</code>	★ expands to the name of the module URI
<code>\stex_module_uri_name:n</code>	★

<code>\stex_module_uri_split_name:NNN</code>
<code>\stex_module_uri_split_name:NNn</code>

sets #1 as the parent module of the module URI and #2 as the last name

<code>\stex_module_uri_as_qm:n</code>	★
---------------------------------------	---

<code>\stex_new_module_uri:n</code>	★ <code>\stex_new_module_uri:n {<name>}</code>
-------------------------------------	------------------------------------------------------

expands to a new module URI triple with the given *<name>* based on the current document URI or module URI, if existent

<code>\stex_uri_from_pair:Nnn</code>	<code>\stex_uri_from_pair:Nnn {<archive id>} {<module path>}</code>
--------------------------------------	---------------------------------------------------------------------------------

resolves the pair [*<archive id>*]{*<module path>*} to a module URI and stores the result in #1

<code>\stex_uri_from_pair:Nnn</code>	like <code>\stex_uri_from_pair:Nnn</code> , but takes a token list as argument that may or may not be of the form [archive]module
--------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------

14.3 Symbol URIs

<code>\stex_use_symbol_uri:N</code>	★ expands to the string representation of the module uri (represented as a macro containing a 4-tuple as above). The variant <code>\stex_symbol_uri:n</code> expects a 4-tuple directly.
<code>\stex_use_symbol_uri:n</code>	★

<code>\stex_new_symbol_uri:n</code>	★ <code>\stex_new_symbol_uri:n {<name>}</code>
-------------------------------------	------------------------------------------------------

expands to a new symbol URI tuple with the given *<name>* based on the current module URI, which is assumed to exist(!).

<code>\stex_new_symbol_uri:nn</code>	★ expands to a new symbol URI tuple with the given <i><name></i> based on the <i>given</i> module URI (either as macro or as tuple), which is assumed to exist(!).
--------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<code>\stex_symbol_uri_archive:N</code>	★
-----------------------------------------	---

expands to the archive ID of the symbol URI

<code>\stex_symbol_uri_path:N</code>	★ 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

`\STEXftmllink`
`\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 `\STEXftmllink`.

`\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

<hr/> <hr/> <code>\l_stex_current_module_uri</code>	stores the URI of the current module, if existent
<hr/> <hr/> <code>\l_stex_all_modules_seq</code>	stores the URIs of all modules <i>currently in scope</i> .
<code>smodule (env.)</code>	Parses the optional arguments, (if relevant) inserts HTML annotations, and then delegates to <code>\stex_module_setup:n</code>
<hr/> <hr/> <code>\stex_module_setup:n</code>	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
<hr/> <hr/> <code>\stex_module_setup_top_nosig:n</code>	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
<hr/> <hr/> <code>\stex_close_module:</code>	closes the current module.
<hr/> <hr/> <code>\stex_every_module:n</code>	adds code to be executed every time a new module is opened (e.g. activating certain macros).
<hr/> <hr/> <code>\stex_do_up_to_module:n</code> <code>\stex_do_up_to_module:e</code>	Execute code in the current module (i.e. as if the <code>\begin{<smodule>}</code> was the current tex group)
<hr/> <hr/> <code>\stex_execute_in_module:n</code> <code>\stex_execute_in_module:e</code>	Execute code in the current module (i.e. as if the <code>\begin{<smodule>}</code> was the current tex group) and also adds it to the initialization code of the module; i.e. exports all macros defined.

`\stex_if_in_module_p:` \star conditional whether we currently are in a module. **TODO: deprecate in favor of every document being a module**
`\stex_if_in_module:` \underline{TF} \star

`\stex_if_module_exists_p:` N \star
`\stex_if_module_exists:` \underline{NTF} \star
`\stex_if_module_exists_p:` n \star
`\stex_if_module_exists:` \underline{nTF} \star

conditional whether a module with the given URI exists

`\stex_activate_module:` N
`\stex_activate_module:` n

`\setmetatheory` `\setmetatheory` [$\langle archive\ id \rangle$] [$\langle module \rangle$]

sets the metatheory of all subsequent modules to the specified one

`\STEXexport` `\STEXexport` [$\langle code \rangle$]

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.

```
\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: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`

`\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 `_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

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`

17.5 Checking

`\stex_if_check_terms_p:` *★* conditional whether terms (types and definientia of symbols) should get syntactically checked. In HTML mode, this is always false, since they get written to the HTML anyway.
`\stex_if_check_terms:` *TF* *★*

`\stex_check_term: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>
-----------------------------------------	---------------------------------------

`_stex_notation_check:`

`_stex_notation_do_html:n`

`_stex_notation_make_args:`

`\stex_do_default_notation:`
`\stex_do_default_notation_op:`

<code>\STEXInternalNotation</code>	#1 : notation id
	#2 : operator precedence
	#3 : intent (WiP)
	#4 : arguments
	#5 : notation code
	#6 : continuation

`\argsep`

`\seqmap`

`\argmap`

`\argarraymap`

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.

<code>\dobrackets</code>	actually inserts parentheses around its argument.
--------------------------	---------------------------------------------------

<code>\withbrackets</code>	<code>\withbrackets {<left>} {<right>} {<body>}</code>
----------------------------	--------------------------------------------------------------------------

sets `<left>` and `<right>` as the parentheses to use in `<body>`.

<code>\dowithbrackets</code>	<code>\dowithbrackets {<left>} {<right>} {<body>}</code>
------------------------------	----------------------------------------------------------------------------

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`

`\spproofend`

`\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}{2025/11/11}{4.0.0}{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}{2025/11/11}{4.0.0}{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~4.0.0~*^^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 115.)
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 115.)
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 115.)
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 115.)

```
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 115.)

```
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 115.)

```
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 116.)

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 116.)

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 116.)
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 116.)

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 116.)

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 116.)

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 116.)

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 116.)

```

\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 117.)

```

\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 117.)

```

\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 117.)

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 117.)

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 117.)

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 117.)

```
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 117.)

```
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 118.)

```
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 118.)

```
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 118.)

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 118.)

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 118.)

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 118.)

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 119.)

`\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 119.)

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 119.)

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 stexcass#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 119.)

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`

(End of definition for `\stex_persist:n`. This function is documented on page 119.)

sfunction

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

```

586 \cs_new_protected:Nn \_stex_persist_read_now: {
587   \bool_if:NTF \c_stex_persist_mode_bool {
588     \bool_if:NTF \c_stex_persist_write_mode_bool
589       \__stex_persist_read_and_write:

```

```

590     {
591       \__stex_persist_load_file:n{\jobname.sms}
592     }
593   }{
594     \bool_if:NT \c_stex_persist_write_mode_bool \__stex_persist_write_only:
595   }
596 }
597
598 \cs_new_protected:Nn \__stex_persist_load_file:n {
599   \file_if_exist:nT{#1}{
600     \group_begin:
601     \cs_set:Npn \stex_persist:n ##1 {}
602     \cs_set:Npn \stex_persist:e ##1 {}
603     \stex_debug:nn{persist}{restoring~from~sms~file}
604     \catcode'\ =10\relax
605     \cs:w @ @ input \cs_end:#1\relax
606   \group_end:
607 }
608 }
609
610 \cs_new_protected:Nn \__stex_persist_write_only: {
611   \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
612   \AtEndDocument{ \iow_close:N \c__stex_persist_sms_iow }
613 }
614
615 \cs_new_protected:Nn \__stex_persist_read_and_write: {
616   \file_if_exist:nTF{\jobname.sms}{
617     \ior_open:Nn \g_tmpa_ior {\jobname.sms}
618     \iow_open:Nn \g_tmpa_iow {\jobname.sms2}
619     \ior_str_map_inline:Nn \g_tmpa_ior {
620       \iow_now:Nn \g_tmpa_iow {##1}
621     }
622     \iow_close:N \g_tmpa_iow
623     \ior_close:N \g_tmpa_ior
624     \__stex_persist_write_only:
625     \ior_open:Nn \g_tmpa_ior {\jobname.sms2}
626     \ior_str_map_inline:Nn \g_tmpa_ior {
627       \iow_now:Nn \c__stex_persist_sms_iow {##1}
628     }
629     \ior_close:N \g_tmpa_ior
630     \__stex_persist_load_file:n{\jobname.sms2}
631   }\__stex_persist_write_only:
632 }

```

26.9 Auxiliary Methods

```

633 <@@=stex_aux>

\_stex_do_deprecation:n

634 \cs_new:Nn \_stex_do_deprecation:n {
635   \str_if_empty:NF \l_stex_key_deprecate_str {
636     \msg_warning:nxxx{stex}{warning/deprecated}{#1}{\l_stex_key_deprecate_str}
637   }
638 }

```

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

`\stex_do_id:`

```
639 \cs_new_protected:Nn \stex_do_id: {  
640   \stex_if_smsmode:F {  
641     \str_if_empty:NF \l_stex_key_id_str {  
642       \exp_args:No \stex_ref_new_doc_target:n \l_stex_key_id_str  
643     }  
644   }  
645 }
```

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

Chapter 27

HTML Output

```
646 <@=stex_annotate>

\stex@backend We determine and \input the backend config file:

647 \ifcsize if@rustex\endcsname\else
648   \expandafter\newif\csname if@rustex\endcsname
649   \@rustexfalse
650 \fi

651
652 \stex_get_env:Nn\__stex_annotate_env_str{STEX_FORCE_PDF}
653 \exp_args:No \str_if_eq:nnTF \__stex_annotate_env_str {true} {
654   \def\stex@backend{pdflatex}
655 }{
656   \tl_if_exist:NF\stex@backend{
657     \if@rustex
658       \def\stex@backend{rustex}
659     \else
660       \cs_if_exist:NTF\HCode{
661         \def\stex@backend{tex4ht}
662       }{
663         \def\stex@backend{pdflatex}
664       }
665     \fi
666   }
667 }

668
669 \input{stex-backend-\stex@backend.cfg}

(End of definition for \stex@backend. This variable is documented on page 120.)

\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 120.)
sfunction

\ifstexhtml

670 \bool_new:N \l__stex_annotate_do_output_bool
671
672 \newif\ifstexhtml
673 \stex_if_html_backend:TF {
```

```

674 \stexhtmltrue
675 \bool_set_true:N \l__stex_annotate_do_output_bool
676 }{
677 \stexhtmlfalse
678 \bool_set_false:N \l__stex_annotate_do_output_bool
679 }

```

(End of definition for `\ifstexhtml`. This function is documented on page 120.)
sfunction

```

\stex_if_do_html_p:
\stex_if_do_html:TF
680 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
681 \bool_if:NTF \l__stex_annotate_do_output_bool
682 \prg_return_true: \prg_return_false:
683 }

```

(End of definition for `\stex_if_do_html:TF`. This function is documented on page 120.)
sfunction

```

\stex_suppress_html:n
684 \cs_new_protected:Nn \stex_suppress_html:n {
685 \stex_pseudogroup:nn{
686 \bool_set_false:N \l__stex_annotate_do_output_bool
687 #1
688 }{
689 \stex_if_do_html:T {
690 \bool_set_true:N \l__stex_annotate_do_output_bool
691 }
692 }
693 }

```

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

`\stex_annotate:nn` is provided by the backend config file.

(End of definition for `\stex_annotate:nn`. This function is documented on page 120.)
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 120.)
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 121.)
sfunction

`\STEXinvisible`

```
694 \cs_new_protected:Npn \STEXinvisible #1 {  
695   \stex_annotate_invisible:n { #1 }  
696 }
```

(End of definition for \STEXinvisible. This function is documented on page 121.)
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 121.)
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 121.)
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 121.)
sfunction

`\mmlintent`

```
\mmlarg  
697 \stex_if_html_backend:TF {  
698   \cs_new_protected:Npn \mmlintent #1 #2 {  
699     \stex_annotate:nn{mml:intent={#1}}{#2}  
700   }  
701   \cs_new_protected:Npn \mmlarg #1 #2 {  
702     \stex_annotate:nn{mml:arg={#1}}{#2}  
703   }  
704 }{  
705   \cs_new_protected:Npn \mmlintent #1 #2 { #2 }  
706   \cs_new_protected:Npn \mmlarg #1 #2 { #2 }  
707 }
```

(End of definition for \mmlintent and \mmlarg. These functions are documented on page 121.)
sfunction

We can now define `\stex_persist:n`:

```
708 <@@=stex_persist>  
709 \bool_if:NTF \c_stex_persist_write_mode_bool {  
710   \stex_if_html_backend:TF{  
711     \cs_new:Npn \stex_persist:n #1 {}  
712     \cs_new:Npn \stex_persist:e #1 {}  
713   }{  
714     \cs_new_protected:Nn \stex_persist:n {  
715       \iow_now:Nn \c__stex_persist_sms_iow {#1}  
716     }  
717     \cs_generate_variant:Nn \stex_persist:n {e}  
718   }  
719 }{  
720   \cs_new:Npn \stex_persist:n #1 {}  
721   \cs_new:Npn \stex_persist:e #1 {}  
722 }
```


and style patching:

```

723 <@@=stex_styles>
724 \stex_if_html_backend:TF{
725   \cs_new_protected:Nn \stex_style_apply: {}
726   \cs_new_protected:Nn \__stex_styles_patch:nnnn {}
727   \stex_keys_define:nnnn{ csspatch }{
728     \str_clear:N \l_stex_keys_cls_id
729     \str_clear:N \l_stex_keys_counter_id
730     \str_clear:N \l_stex_keys_counter_parent
731   }{
732     counter      .str_set_x:N = \l_stex_keys_counter_id,
733     parent       .str_set_x:N = \l_stex_keys_counter_parent,
734     unknown      .code:n      = {
735       \exp_args:NNo \str_set:Nn \l_stex_keys_cls_id {\l_keys_key_tl}
736     }
737   }{}
738   \cs_new_protected:Npn \__stex_styles_css_patch:nnnn #1 #2 {
739     \stex_keys_set:nn{ csspatch }{ #2 }
740     \group_begin:
741     \catcode'\ =12\relax
742     \catcode'^=12\relax
743     \__stex_styles_patch_i:nnn {#1}
744   }
745
746   \seq_new:N \g__stex_styles_counters_seq
747
748   \cs_new:Nn \__stex_styles_patch_html_c: {
749     \stex_html_literal:n{
750       <span~data-ftml-counter="\l_stex_keys_counter_id"~style="display:none;"~
751       data-ftml-counter-parent="\l_stex_keys_counter_parent"
752       ></span>
753     }
754   }
755
756   \cs_new:Nn \__stex_styles_patch_html: {
757     \stex_html_literal:n{
758       <span~data-ftml-style="\l_stex_keys_cls_id"~style="display:none;"~
759       data-ftml-counter="\l_stex_keys_counter_id"
760       ></span>
761     }
762   }
763
764   \cs_new_protected:Nn \__stex_styles_patch_i:nnn {
765     \str_if_empty:NTF \l_stex_keys_cls_id {
766       \str_set:Nn \l_stex_keys_cls_id { #1 }
767     }{
768       \str_set:Ne \l_stex_keys_cls_id { #1-\l_stex_keys_cls_id }
769     }
770     \exp_args:No \str_case:nnF \l_stex_keys_counter_parent {
771       {}{}
772       {part}{\str_set:Nn \l_stex_keys_counter_parent { 0 }}
773       {chapter}{\str_set:Nn \l_stex_keys_counter_parent { 1 }}
774       {section}{\str_set:Nn \l_stex_keys_counter_parent { 2 }}
775       {subsection}{\str_set:Nn \l_stex_keys_counter_parent { 3 }}

```

```

776     {subsubsection}{\str_set:Nn \l_stex_keys_counter_parent { 4 }}
777     {paragraph}{\str_set:Nn \l_stex_keys_counter_parent { 5 }}
778     {subparagraph}{\str_set:Nn \l_stex_keys_counter_parent { 6 }}
779   }{
780     \msg_error:nnx{stex}{error/notasection}\l_stex_keys_counter_parent
781   }
782   \str_set:Ne \l__stex_styles_first_str { &~.ftml-title-paragraph~ { display:inline-block
783   \str_if_empty:NF \l_stex_keys_counter_id {
784     %\str_put_left:Ne \l__stex_styles_first_str { counter-increment:~ ftml-\l_stex_keys_c
785     \exp_args:NNo \seq_if_in:NnF \g__stex_styles_counters_seq \l_stex_keys_counter_id {
786       \seq_gpush:No \g__stex_styles_counters_seq \l_stex_keys_counter_id
787       \cs_if_eq:ccTF{@onlypreamble}{@notprerr}{
788         \__stex_styles_patch_html_c:
789         % in body
790       }{
791         \exp_args:Ne \AtBeginDocument \__stex_styles_patch_html_c:
792         % in preamble
793       }
794     }
795     \cs_if_eq:ccTF{@onlypreamble}{@notprerr}{
796       \__stex_styles_patch_html:
797       % in body
798     }{
799       \exp_args:Ne \AtBeginDocument \__stex_styles_patch_html:
800       % in preamble
801     }
802   }
803   \exp_args:Ne \stex_css_literal:n { .ftml-\l_stex_keys_cls_id { \l__stex_styles_first_st
804
805   % TODO export counter reset somehow
806
807   \group_end:
808 }
809 }{
810   \cs_new_protected:Nn \__stex_styles_patch:nnnn {
811     \str_if_empty:NTF {#2}{
812       \tl_set:cn{__stex_styles_style_#1_start:}{#3}
813       \tl_set:cn{__stex_styles_style_#1_end:}{#4}
814     }{
815       \tl_set:cn{__stex_styles_style_#1_#2_start:}{#3}
816       \tl_set:cn{__stex_styles_style_#1_#2_end:}{#4}
817     }
818   }
819   \cs_new_protected:Nn \__stex_styles_css_patch:nnnn {}
820 }

```

Chapter 28

Math Archives

```
\c_stex_mathhub_files
\mathhub
821 (@@=stex_mathhub)
822 \str_if_empty:NTF\mathhub{
823   \stex_get_env:Nn \l__stex_mathhub_str {MATHHUB}
824   \str_if_empty:NTF \l__stex_mathhub_str {
825     \ior_open:NnTF \g_tmpa_ior{\stex_file_use:N \c_stex_home_file/.stex/mathhub.path}{
826       \group_begin:
827         \escapechar=-1\catcode'\=12
828         \ior_str_get:NN \g_tmpa_ior \l__stex_mathhub_str
829         \str_gset_eq:NN \l__stex_mathhub_str \l__stex_mathhub_str
830       \group_end:
831       \ior_close:N \g_tmpa_ior
832       \stex_debug:nn{mathhub}{MathHub-directory~determined~from~home~directory}
833     }{
834       \str_clear:N \l__stex_mathhub_str
835     }
836   }{
837     \stex_debug:nn{mathhub}{MathHub-directory~determined~from~environment~variable}
838   }
839 }{
840   \str_set_eq:NN \l__stex_mathhub_str \mathhub
841 }
842
843 \str_if_empty:NTF \l__stex_mathhub_str {
844   \msg_warning:nn{stex}{warning/nomathhub}
845   \stex_file_set:Ne \l_tmpa_seq {\stex_file_use:N \c_stex_home_file \tl_to_str:n{/MathHub}}
846   \seq_clear:N \c_stex_mathhub_files
847   \seq_push:Ne \c_stex_mathhub_files {\stex_file_use:N \l_tmpa_seq}
848 }{
849   \seq_clear:N \c_stex_mathhub_files
850   \seq_set_split:NnV \l_tmpa_seq , \l__stex_mathhub_str
851   \seq_map_inline:Nn \l_tmpa_seq {
852     \stex_file_resolve:Nn \l__stex_mathhub_str {#1}
853     \stex_if_file_absolute:NF \l__stex_mathhub_str {
854       \stex_file_resolve:Ne \l__stex_mathhub_str {
855         \stex_file_use:N \c_stex_pwd_file / #1
856       }

```

```

857     }
858     \seq_put_right:Ne \c_stex_mathhub_files {
859       \stex_file_use:N \l__stex_mathhub_str
860     }
861   }
862 }
863
864 \exp_args:NNe \str_set:Nn \mathhub {\seq_use:Nn \c_stex_mathhub_files ,}
865 \stex_debug:nn{mathhub}{MATHHUBS:~\mathhub}

```

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

`\stex_archive_id:N`

```

866 \cs_new:Npn \stex_archive_id:N {
867   \exp_after:wN \use_i:nnn
868 }

```

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

sfunction

`\stex_archive_base:N`

`\stex_archive_base:n`

```

869 \cs_new:Npn \stex_archive_base:N {
870   \exp_after:wN \use_ii:nnn
871 }
872
873 \cs_new:Nn \stex_archive_base:n {
874   \exp_args:NNe \use:nn \use_ii:nnn { \use:c {c_stex_mathhub_#1_archive} }
875 }

```

(End of definition for `\stex_archive_base:N` and `\stex_archive_base:n`. These functions are documented on page 124.)

sfunction

`\stex_archive_path:N`

`\stex_archive_path:n`

```

876 \cs_new:Npn \stex_archive_path:N {
877   \exp_after:wN \use_iii:nnn
878 }
879
880 \cs_new:Nn \stex_archive_path:n {
881   \exp_args:NNe \use:nn \use_iii:nnn { \use:c {c_stex_mathhub_#1_archive} }
882 }

```

(End of definition for `\stex_archive_path:N` and `\stex_archive_path:n`. These functions are documented on page 124.)

sfunction

`\stex_in_archive:nn`

```

883 \cs_new_protected:Nn \stex_in_archive:nn {
884   \stex_pseudogroup:nn{
885     \cs_set:Npn \l__stex_mathhub_cs ##1 {#2}
886     \tl_if_empty:nTF{#1}{
887       \tl_if_exist:NTF \l_stex_current_archive {
888         \exp_args:Ne \l__stex_mathhub_cs {\stex_archive_id:N \l_stex_current_archive }
889       }{

```

```

890     \l__stex_mathhub_cs {}
891   }
892   }{
893     \__stex_mathhub_set_current:n{#1}
894     \l__stex_mathhub_cs {#1}
895   }
896   }{
897     \stex_pseudogroup_restore:N \l_stex_current_archive
898     \cs_set:Npn \exp_not:N \l__stex_mathhub_cs ##1 {
899       \exp_args:No \exp_not:n {\l__stex_mathhub_cs {##1}}
900     }
901   }
902 }
903
904 \cs_set:Npn \l__stex_mathhub_cs #1 {}

```

Auxiliary macros for loading archives:

```

905 \cs_new_protected:Nn \__stex_mathhub_set_current:n {
906   \__stex_mathhub_require:n { #1 }
907   \stex_debug:nn{mathhub}{switching-to~archive~#1}
908   \tl_set_eq:Nc \l_stex_current_archive {
909     c_stex_mathhub_#1_archive
910   }
911 }
912
913 \cs_new_protected:Nn \__stex_mathhub_require:n {
914   \prop_if_exist:cF { c_stex_mathhub_#1_archive } {
915     \seq_if_empty:NTF \c_stex_mathhub_files {
916       \msg_fatal:nn{stex}{warning/nomathhub}
917     }{
918       \stex_debug:nn{mathhub}{Opening~archive:~#1}
919       \__stex_mathhub_do_manifest:nn { #1 }{
920         \msg_fatal:nnee{stex}{error/noarchive}
921         {#1}{\seq_use:Nn \c_stex_mathhub_files ,}
922       }
923     }
924   }
925 }

```

Attempts to find the archive's manifest file:

```

926 \cs_new_protected:Nn \__stex_mathhub_do_manifest:nn {
927   \seq_map_inline:Nn \c_stex_mathhub_files {
928     \__stex_mathhub_find_manifest:nn {##1}{#1}
929     \str_if_empty:NF \l__stex_mathhub_manifest_str \seq_map_break:
930   }
931   %\exp_args:Ne \__stex_mathhub_find_manifest:n {\stex_file_use:N \c_stex_mathhub_file / #1}
932   \str_if_empty:NTF \l__stex_mathhub_manifest_str { #2 }{
933     \__stex_mathhub_parse_manifest:n {#1}
934   }
935 }
936
937 \cs_new_protected:Nn \__stex_mathhub_find_manifest:nn {
938   \str_clear:N \l__stex_mathhub_manifest_str
939   \seq_set_split:Nnn \l_tmpa_seq / { #1 }
940   \seq_set_split:Nnn \l__stex_mathhub_seq / {#1 / #2}

```

```

941 \bool_set_true:N \l__stex_mathhub_bool
942 \bool_while_do:Nn \l__stex_mathhub_bool {
943   \tl_if_eq:NNTF \l__stex_mathhub_seq \l_tmpa_seq {
944     \bool_set_false:N \l__stex_mathhub_bool
945   }{
946     \__stex_mathhub_check_manifest:
947     \bool_if:NT \l__stex_mathhub_bool {
948       \seq_pop_right:NN \l__stex_mathhub_seq \l__stex_mathhub_tl
949     }
950   }
951 }
952 }
953
954 \cs_new_protected:Nn \__stex_mathhub_check_manifest: {
955   \__stex_mathhub_check_manifest:n {MANIFEST.MF}
956   \bool_if:NT \l__stex_mathhub_bool {
957     \__stex_mathhub_check_manifest:n {META-INF/MANIFEST.MF}
958     \bool_if:NT \l__stex_mathhub_bool {
959       \__stex_mathhub_check_manifest:n {meta-inf/MANIFEST.MF}
960     }
961   }
962 }
963
964 \cs_new_protected:Nn \__stex_mathhub_check_manifest:n {
965   \stex_debug:nn{mathhub}{Checking~\stex_file_use:N \l__stex_mathhub_seq / #1}
966   \file_if_exist:nT {\stex_file_use:N \l__stex_mathhub_seq / #1} {
967     \bool_set_false:N \l__stex_mathhub_bool
968     \str_set:Ne \l__stex_mathhub_manifest_str {\stex_file_use:N \l__stex_mathhub_seq / #1}
969   }
970 }

```

Parses the archive's manifest file:

```

971 \ior_new:N \c__stex_mathhub_manifest_ior
972
973 \str_set:Nn \l_tmpa_str {https://}
974
975 \exp_after:wN \cs_new:Npn \exp_after:wN \__stex_mathhub_replace_https:w \l_tmpa_str #1 \__s
976   http:// #1
977 }
978
979 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
980   \ior_open:Nn \c__stex_mathhub_manifest_ior \l__stex_mathhub_manifest_str
981   \str_set:Ne \l__stex_mathhub_file_str {\stex_file_use:N \l__stex_mathhub_seq}
982   \str_set:Nn \l__stex_mathhub_id_str {#1}
983   \str_set:Nn \l__stex_mathhub_base_str {http://mathhub.info}
984
985   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
986     \exp_args:NNNo \exp_args:NNNx
987     \seq_set_split:Nnn \l__stex_mathhub_seq \c_colon_str {\tl_to_str:n{##1}}
988     \seq_pop_left:NNT \l__stex_mathhub_seq \l__stex_mathhub_key {
989       \exp_args:NNo \str_set:Nn \l__stex_mathhub_key \l__stex_mathhub_key
990       \str_set:Ne \l__stex_mathhub_val {\seq_use:Nn \l__stex_mathhub_seq :}
991       \str_case:Nn \l__stex_mathhub_key {
992         {id} { \str_set_eq:NN \l__stex_mathhub_id_str \l__stex_mathhub_val }
993         {url-base} { \str_set_eq:NN \l__stex_mathhub_base_str \l__stex_mathhub_val }

```

```

994     }
995   }
996 }
997 \ior_close:N \c__stex_mathhub_manifest_ior
998 \exp_args:No \stex_str_if_starts_with:nnT \l__stex_mathhub_base_str {https://} {
999   \str_set:Ne \l__stex_mathhub_base_str { \exp_after:wN \__stex_mathhub_replace_https:w \
1000 }
1001 \tl_gset:ce { c_stex_mathhub_#1_archive } { {\l__stex_mathhub_id_str} {\l__stex_mathhub_b
1002 \stex_debug:nn{mathhub}{Result:~\use:c{c_stex_mathhub_#1_archive}}
1003 \str_if_eq:nnF {#1}{main}{
1004   \stex_persist:e {
1005     \__stex_mathhub_restore:nn{\l__stex_mathhub_id_str}{\l__stex_mathhub_base_str}
1006   }
1007 }
1008 }

```

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.

```

1009 \cs_set_protected:Nn \__stex_mathhub_restore:nn {
1010   \__stex_mathhub_do_manifest:nn { #1 }{}
1011   \tl_if_exist:cF { c_stex_mathhub_#1_archive }{
1012     \tl_set:ce{ c_stex_mathhub_#1_archive }{
1013       {\tl_to_str:n{#1}}
1014       {\tl_to_str:n{#2}}
1015     }
1016   }
1017 }
1018 }
1019

```

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

sfunction

```

\c_stex_no_archive_str
\c_stex_no_archive
1020 \str_const:Nn \c_stex_no_archive_str { no/archive }
1021
1022 \tl_const:Ne \c_stex_no_archive_uri {
1023   {\c_stex_no_archive_str}
1024   {\tl_to_str:n{http://unknown.source}}
1025   {}
1026 }
1027
1028 \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 125.)

```

\c_stex_main_archive
\l_stex_current_archive
1029 \seq_map_inline:Nn \c_stex_mathhub_files {
1030   \seq_set_split:Nnn \l_tmpa_seq / {#1}
1031   \stex_if_file_starts_with:NNT \c_stex_pwd_file \l_tmpa_seq {
1032     \seq_set_eq:NN \l_tmpb_seq \c_stex_pwd_file

```

```

1033 \seq_map_inline:Nn \l_tmpa_seq { \seq_pop_left:NN \l_tmpb_seq \l_tmpa_tl }
1034 \exp_args:Nne \__stex_mathhub_find_manifest:nn {#1} { \stex_file_use:N \l_tmpb_seq }
1035 \str_if_empty:NTF \l__stex_mathhub_manifest_str {
1036   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~archive}
1037 }{
1038   \__stex_mathhub_parse_manifest:n { main }
1039   \tl_set_eq:NN \c_stex_main_archive \c_stex_mathhub_main_archive
1040   \cs_undefine:N \c_stex_mathhub_main_archive
1041   \tl_set_eq:cN { c_stex_mathhub_ \stex_archive_id:N \c_stex_main_archive _archive }
1042   \c_stex_main_archive
1043   \prop_set_eq:NN \l_stex_current_archive \c_stex_main_archive
1044   \stex_debug:nn{mathhub}{Current~archive::~
1045     \stex_archive_id:N \c_stex_main_archive
1046   }
1047   \stex_persist:e {
1048     \__stex_mathhub_restore:nn{\stex_archive_id:N \c_stex_main_archive}{\stex_archive_b
1049     \tl_gset_eq:Nc \exp_not:N \c_stex_main_archive {c_stex_mathhub_ \stex_archive_id:N
1050     \tl_gset_eq:NN \exp_not:N \l_stex_current_archive \exp_not:N \c_stex_main_archive
1051   }
1052   %\bool_if:NT \c_stex_persist_write_mode_bool {
1053   %   \tl_put_right:Ne \stex_persist_read_now: {
1054   %     \stex_persist:n {{c_stex_mathhub_ \stex_archive_id:N \c_stex_main_archive _archi
1055   %     \tl_gset_eq:cN{c_stex_mathhub_ \stex_archive_id:N \c_stex_main_archive _manife
1056   %     \tl_gset_eq:NN \exp_not:N \l_stex_current_archive \exp_not:N \c_stex_main_archi
1057   %   }
1058   % }
1059   %}
1060 }
1061 \seq_map_break:
1062 }
1063 }
1064 \tl_if_exist:NF \c_stex_main_archive {
1065   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~directory}
1066 }

```

(End of definition for `\c_stex_main_archive` and `\l_stex_current_archive`. These variables are documented on page 125.)

```

\stex_source_path:n
\stex_source_path:nn
1067 \cs_new:Nn \stex_source_path:n {
1068   \tl_if_exist:NTF \l_stex_current_archive {
1069     \stex_archive_path:N \l_stex_current_archive / source \tl_if_empty:nF{#1}{/ #1}
1070   }{
1071     \stex_file_use:N \g_stex_current_file / .. \tl_if_empty:nF{#1}{/ #1}
1072   }
1073 }
1074 \cs_new:Nn \stex_source_path:nn {
1075   \tl_if_empty:nTF{#1}{
1076     \stex_source_path:n {#2}
1077   }{
1078     \tl_if_exist:cTF {c_stex_mathhub_ #1 _archive } {
1079       \stex_archive_path:n {#1} / source \tl_if_empty:nF{#2}{/ #2}
1080     }{
1081       \stex_file_use:N \g_stex_current_file / .. \tl_if_empty:nF{#2}{/ #2}

```



```

1082     }
1083   }
1084 }

```

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

```

sfunction

```

Chapter 29

URIs

```
1085 <@@=stex_uris>
```

```
\stex_use_archive_uri:n
```

```
1086 \cs_new:Nc \stex_use_archive_uri:n {  
1087   \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1  
1088 }
```

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

29.1 Document URIs

```
\stex_use_document_uri:N
```

```
\stex_use_document_uri:n
```

```
1089 \cs_new:Nn \stex_use_document_uri:N {  
1090   \exp_after:wN \__stex_uris_doc:nnnnn #1  
1091 }  
1092 \cs_new:Nn \stex_use_document_uri:n {  
1093   \__stex_uris_doc:nnnnn #1  
1094 }  
1095 \cs_new:Nc \__stex_uris_doc:nnnnn {  
1096   \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1  
1097   \exp_not:N \tl_if_empty:nF{#2}{  
1098     \c_ampersand_str\tl_to_str:n{p=}#2  
1099   }  
1100   \c_ampersand_str\tl_to_str:n{d=}#3  
1101   \c_ampersand_str\tl_to_str:n{l=}#4  
1102   \exp_not:N \tl_if_empty:nF{#5}{  
1103     \c_ampersand_str\tl_to_str:n{e=}#5  
1104   }  
1105 }
```

(End of definition for \stex_use_document_uri:N and \stex_use_document_uri:n. These functions are documented on page 126.)
sfunction

```
\stex_document_uri_archive:N
```

```
1106 \cs_new:Nn \stex_document_uri_archive:N {  
1107   \exp_after:wN \use_i:nnnnn #1
```

```

1108 }
(End of definition for \stex_document_uri_archive:N. This function is documented on page 126.)
sfunction

```

`\stex_document_uri_path:N`

```

1109 \cs_new:Nn \stex_document_uri_path:N {
1110   \exp_after:wN \use_ii:nnnnn #1
1111 }
(End of definition for \stex_document_uri_path:N. This function is documented on page 127.)
sfunction

```

`\stex_document_uri_name:N`

```

1112 \cs_new:Nn \stex_document_uri_name:N {
1113   \exp_after:wN \use_iii:nnnnn #1
1114 }
(End of definition for \stex_document_uri_name:N. This function is documented on page 127.)
sfunction

```

`\stex_document_uri_language:N`

```

1115 \cs_new:Nn \stex_document_uri_language:N {
1116   \exp_after:wN \use_iv:nnnnn #1
1117 }
(End of definition for \stex_document_uri_language:N. This function is documented on page 127.)
sfunction

```

`\stex_document_uri_element:N`

```

1118 \cs_new:Nn \stex_document_uri_element:N {
1119   \exp_after:wN \use_v:nnnnn #1
1120 }
(End of definition for \stex_document_uri_element:N. This function is documented on page 127.)
sfunction

```

`\stex_document_uri_with_language:Nn`

```

1121 \cs_new:Npn \stex_document_uri_with_language:Nn {
1122   \exp_after:wN \__stex_uris_with_language:nnnnnn
1123 }
1124 \cs_new:Nn \__stex_uris_with_language:nnnnnn {
1125   {#1}{#2}{#3}{#6}{}
1126 }
(End of definition for \stex_document_uri_with_language:Nn. This function is documented on page 127.)
sfunction

```

`\stex_new_document_element_uri:n`

```

1127 \cs_new:Npn \stex_new_document_element_uri:n {
1128   \exp_after:wN \__stex_uris_with_elem:nnnnnn \l_stex_current_document_uri
1129 }
1130 \cs_new:Nn \__stex_uris_with_elem:nnnnnn {
1131   {#1}{#2}{#3}{#4}{ \tl_if_empty:nTF{#5}{#6}{#5/#6}}
1132 }

```

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

sfunction

`\stex_document_uri_from_archive_file:Nn`

```

1133 \cs_new_protected:Nn \stex_document_uri_from_archive_file:Nn {
1134   \stex_file_set:Nn \l__stex_uris_file {#2}
1135   \stex_debug:nn{URI}{relative~path:~\stex_file_use:N \l__stex_uris_file}
1136   \__stex_uris_doc_from_archive_file:NN #1 \l__stex_uris_file
1137 }
1138
1139 \cs_new_protected:Nn \__stex_uris_doc_from_archive_file:NN {
1140   \__stex_uris_split_file:N #2
1141   \prop_if_exist:NTF \l_stex_current_archive {
1142     \str_set:Ne \l__stex_uris_archive {
1143       \stex_archive_id:N \l_stex_current_archive
1144     }
1145   }{
1146     \str_set_eq:NN \l__stex_uris_archive \c_stex_no_archive_str
1147   }
1148   \tl_set:Ne #1 {
1149     { \l__stex_uris_archive }
1150     { \stex_file_use:N \l__stex_uris_path_seq }
1151     { \l__stex_uris_name_str }
1152     { \l__stex_uris_lang_str }
1153   }{}
1154 }
1155 }
1156
1157 \cs_new_protected:Nn \__stex_uris_split_file:N {
1158   \seq_set_eq:NN \l__stex_uris_path_seq #1
1159   \seq_pop_right:NN \l__stex_uris_path_seq \l__stex_uris_name_str
1160   \seq_set_split:NnV \l_tmpa_seq . \l__stex_uris_name_str
1161   \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_lang_str % .tex?
1162   \seq_if_empty:NTF \l_tmpa_seq {
1163     \str_set_eq:NN \l__stex_uris_name_str \l__stex_uris_lang_str
1164     \str_set_eq:NN \l__stex_uris_lang_str \l_stex_current_language_str
1165   }{
1166     \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_lang_str % actual language maybe?
1167     \cs_if_eq:NNTF \l__stex_uris_lang_str \q_no_value {
1168       \str_set_eq:NN \l__stex_uris_lang_str \l_stex_current_language_str
1169     }{
1170       \prop_if_in:NoF \c_stex_languages_prop \l__stex_uris_lang_str {
1171         \seq_put_right:No \l_tmpa_seq \l__stex_uris_lang_str
1172         \str_set:Nn \l__stex_uris_lang_str {en}
1173       }
1174     }
1175     \str_set:Ne \l__stex_uris_name_str {\seq_use:Nn \l_tmpa_seq .}
1176   }
1177 }

```

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

sfunction

```

\c_stex_main_document_uri
\l_stex_current_document_uri
1178 \tl_if_exist:NTF \l_stex_current_archive {
1179   \str_set:Ne \l_tmpa_str { \stex_archive_path:N \l_stex_current_archive }
1180   \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
1181   \seq_set_eq:NN \l_tmpb_seq \c_stex_main_file
1182   \seq_map_inline:Nn \l_tmpa_seq {
1183     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_tl
1184   }
1185   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_tl
1186   \__stex_uris_doc_from_archive_file:NN \l_stex_current_document_uri \l_tmpb_seq
1187 }{
1188   \__stex_uris_doc_from_archive_file:NN \l_stex_current_document_uri \c_stex_main_file
1189 }
1190
1191 \tl_set_eq:NN \c_stex_main_document_uri \l_stex_current_document_uri
1192 \str_set:Ne \l_stex_current_language_str { \stex_document_uri_language:N \l_stex_current_do
1193
1194 \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 127.)

29.2 Module URIs

```

\stex_use_module_uri:N
\stex_use_module_uri:n
1195 \cs_new:Nn \stex_use_module_uri:N {
1196   \exp_after:wN \__stex_uris_module:nnn #1
1197 }
1198 \cs_new:Nn \stex_use_module_uri:n {
1199   \__stex_uris_module:nnn #1
1200 }
1201 \cs_new:Ne \__stex_uris_module:nnn {
1202   \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1
1203   \exp_not:N \tl_if_empty:nF{#2}{
1204     \c_ampersand_str\tl_to_str:n{p=}#2
1205   }
1206   \c_ampersand_str\tl_to_str:n{m=}#3
1207 }

```

(End of definition for `\stex_use_module_uri:N` and `\stex_use_module_uri:n`. These functions are documented on page 127.)

sfunction

```

\stex_module_uri_archive:N
1208 \cs_new:Nn \stex_module_uri_archive:N {
1209   \exp_after:wN \use_i:nnn #1
1210 }

```

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

sfunction

\stex_module_uri_path:N
\stex_module_uri_path:n

```
1211 \cs_new:Nn \stex_module_uri_path:N {
1212   \exp_after:wN \use_ii:nnn #1
1213 }
1214 \cs_new:Nn \stex_module_uri_path:n {
1215   \use_ii:nnn #1
1216 }
```

(End of definition for \stex_module_uri_path:N and \stex_module_uri_path:n. These functions are documented on page 127.)

sfunction

\stex_module_uri_name:N
\stex_module_uri_name:n

```
1217 \cs_new:Nn \stex_module_uri_name:N {
1218   \exp_after:wN \use_iii:nnn #1
1219 }
1220
1221 \cs_new:Nn \stex_module_uri_name:n {
1222   \use_iii:nnn #1
1223 }
```

(End of definition for \stex_module_uri_name:N and \stex_module_uri_name:n. These functions are documented on page 128.)

sfunction

\stex_module_uri_split_name:NNN
\stex_module_uri_split_name:NNn

```
1224 \cs_new_protected:Npn \stex_module_uri_split_name:NNN #1 {
1225   \exp_after:wN \__stex_uris_parent:NNnnn \exp_after:wN #1 \exp_after:wN
1226 }
1227
1228 \cs_new_protected:Nn \stex_module_uri_split_name:NNn {
1229   \__stex_uris_parent:NNnnn #1 #2 #3
1230 }
1231
1232 \cs_new_protected:Nn \__stex_uris_parent:NNnnn {
1233   \seq_set_split:Nnn #1 / {#5}
1234   \seq_pop_right:NN #1 #2
1235   \seq_if_empty:NTF #1 {
1236     \tl_set:Ne #1 { {#3} {#4} {#3}}
1237     \tl_clear:N #2
1238   }{
1239     \tl_set:Ne #1 { {#3} {#4} {\seq_use:Nn #1 /} }
1240   }
1241 }
```

(End of definition for \stex_module_uri_split_name:NNN and \stex_module_uri_split_name:NNn. These functions are documented on page 128.)

sfunction

\stex_module_uri_as_qm:n

```
1242 \cs_new:Nn \stex_module_uri_as_qm:n {
1243   \__stex_uris_as_qm:nnn #1
1244 }
1245 \cs_new:Nn \__stex_uris_as_qm:nnn {
1246   #1 / #2 ? #3
1247 }
```

(End of definition for \stex_module_uri_as_qm:n. This function is documented on page 128.)

sfunction

\stex_new_module_uri:n

```

1248 \cs_new:Npn \stex_new_module_uri:n {
1249   \stex_if_in_module:TF {
1250     \exp_after:wN \__stex_uris_new_nested_mod:nnnn \l_stex_current_module_uri
1251   }{
1252     \exp_after:wN \__stex_uris_new_mod:nnnnnn \l_stex_current_document_uri
1253   }
1254 }
1255
1256 \cs_new:Nn \__stex_uris_new_mod:nnnnnn {
1257   {#1}
1258   \str_if_eq:nnTF{#3}{#6}{
1259     {#2}{#3}
1260   }{
1261     {#2 \tl_if_empty:nF{#2}/ #3}{ \tl_to_str:n{ #6 } }
1262   }
1263 }
1264 \cs_new:Nn \__stex_uris_new_nested_mod:nnnn {
1265   {#1}
1266   {#2}
1267   {#3 / \tl_to_str:n{#4}}
1268 }

```

(End of definition for \stex_new_module_uri:n. This function is documented on page 128.)

sfunction

\stex_uri_from_pair:Nnn

```

1269 \bool_new:N \l_stex_relative_import_bool
1270 \cs_new_protected:Nn \stex_uri_from_pair:Nnn {
1271   \bool_set_false:N \l_stex_relative_import_bool
1272   \stex_debug:nn{uri}{resolving~<#2,#3>~in~\stex_use_document_uri:N \l_stex_current_document_uri}
1273   \seq_set_split:Nne \l_tmpa_seq ? { \tl_to_str:n {#3} }
1274   \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_name_str
1275   \str_clear:N \l__stex_uris_path_str
1276   \seq_if_empty:NF \l_tmpa_seq {
1277     \seq_pop_right:NN \l_tmpa_seq \l__stex_uris_path_str
1278   }
1279   \tl_if_empty:nTF { #2 }{
1280     \str_if_empty:NTF \l__stex_uris_path_str
1281     \__stex_uris_maybe_relative:N \__stex_uris_absolute:N
1282     #1
1283   }{
1284     \__stex_uris_pair_in_archive:Nn #1 { #2 }
1285   }
1286 }
1287 %\cs_generate_variant:Nn \stex_uri_from_pair:Nnn {Nee}
1288
1289 \cs_new_protected:Nn \__stex_uris_maybe_relative:N {
1290   \tl_set:Ne \l_tmpb_tl {
1291     \stex_document_uri_path:N \l_stex_current_document_uri
1292   }

```

```

1293 \tl_set:Ne \l_tmpa_tl {
1294   {
1295     \stex_document_uri_archive:N \l_stex_current_document_uri
1296   }
1297   {
1298     \l_tmpb_tl
1299     \exp_args:NNo \exp_args:Nne \str_if_eq:nnF \l__stex_uris_name_str {
1300       \stex_document_uri_name:N \l_stex_current_document_uri
1301     }{
1302       \str_if_empty:NF \l_tmpb_tl / \stex_document_uri_name:N \l_stex_current_document_uri
1303     }
1304   }
1305   { \l__stex_uris_name_str }
1306 }
1307 \stex_if_module_exists:NTF \l_tmpa_tl {
1308   \tl_set_eq:NN #1 \l_tmpa_tl
1309   \bool_set_true:N \l_stex_relative_import_bool
1310   \stex_debug:nn{uri}{returning~relative~\stex_use_module_uri:N #1}
1311 }{
1312   \__stex_uris_absolute:N #1
1313 }
1314 }
1315
1316 \cs_new_protected:Nn \__stex_uris_absolute:N {
1317   \tl_if_exist:NTF \l_stex_current_archive {
1318     \tl_set:Ne #1 {
1319       { \stex_archive_id:N \l_stex_current_archive }
1320       { \l__stex_uris_path_str }
1321       { \l__stex_uris_name_str }
1322     }
1323     \stex_debug:nn{uri}{returning~\stex_use_module_uri:N #1}
1324   }{
1325     \__stex_uris_pair_no_archive:N #1
1326   }
1327 }
1328
1329 \cs_new_protected:Nn \__stex_uris_pair_no_archive:N {
1330   \stex_file_resolve:Ne \l_tmpa_seq {
1331     \stex_file_use:N \g_stex_current_file / .. / \l__stex_uris_path_str
1332   }
1333   \tl_set:Ne #1 {
1334     { \c_stex_no_archive_str }
1335     { \stex_file_use:N \l_tmpa_seq }
1336     { \l__stex_uris_name_str }
1337   }
1338   \stex_debug:nn{uri}{returning~\stex_use_module_uri:N #1}
1339 }
1340
1341 \cs_new_protected:Nn \__stex_uris_pair_in_archive:Nn {
1342   \tl_set:Ne #1 {
1343     { \tl_to_str:n{ #2 } }
1344     { \l__stex_uris_path_str }
1345     { \l__stex_uris_name_str }
1346   }

```



```

1347 \stex_debug:nn{uri}{returning~\stex_use_module_uri:N #1}
1348 }

```

(End of definition for \stex_uri_from_pair:Nnn. This function is documented on page 128.)

```

sfunction

```

\stex_uri_from_pair:Nn

```

1349 \cs_new_protected:Nn \stex_uri_from_pair:Nn {
1350   \peek_charcode:NTF [ {
1351     \__stex_uris_parse_i:Nw #1
1352   }{
1353     \__stex_uris_parse_ii:Nw #1
1354   } #2 \__stex_uris_end:
1355 }
1356
1357 \cs_new_protected:Npn \__stex_uris_parse_i:Nw #1 [#2] #3 \__stex_uris_end: {
1358   \stex_uri_from_pair:Nnn #1 {#2} {#3}
1359 }
1360
1361 \cs_new_protected:Npn \__stex_uris_parse_ii:Nw #1 #2 \__stex_uris_end: {
1362   \stex_uri_from_pair:Nnn #1 {} {#2}
1363 }

```

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

```

sfunction

```

29.3 Symbol URIs

\stex_use_symbol_uri:N

\stex_use_symbol_uri:n

```

1364 \cs_new:Nn \stex_use_symbol_uri:N {
1365   \exp_after:wN \__stex_uris_symbol:nnnn #1
1366 }
1367 \cs_new:Nn \stex_use_symbol_uri:n {
1368   \__stex_uris_symbol:nnnn #1
1369 }
1370 \cs_new:Nc \__stex_uris_symbol:nnnn {
1371   \exp_not:N \stex_archive_base:n{#1} \tl_to_str:n{?a=} #1
1372   \exp_not:N \tl_if_empty:nF{#2}{
1373     \c_ampersand_str\tl_to_str:n{p=}#2
1374   }
1375   \c_ampersand_str\tl_to_str:n{m=}#3
1376   \c_ampersand_str\tl_to_str:n{s=}#4
1377 }

```

(End of definition for \stex_use_symbol_uri:N and \stex_use_symbol_uri:n. These functions are documented on page 128.)

```

sfunction

```

\stex_new_symbol_uri:n

```

1378 \cs_new:Nn \stex_new_symbol_uri:n {
1379   \l_stex_current_module_uri { \tl_to_str:n{ #1 } }
1380 }

```

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

sfunction

`\stex_new_symbol_uri:nn`

```
1381 \cs_new:Nn \stex_new_symbol_uri:nn {
1382   #1 { #2 }
1383 }
```

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

sfunction

`\stex_symbol_uri_archive:N`

```
1384 \cs_new:Nn \stex_symbol_uri_archive:N {
1385   \exp_after:wN \use_i:nnnn #1
1386 }
```

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

sfunction

`\stex_symbol_uri_path:N`

```
1387 \cs_new:Nn \stex_symbol_uri_path:N {
1388   \exp_after:wN \use_ii:nnnn #1
1389 }
```

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

sfunction

`\stex_symbol_uri_module:N`

`\stex_symbol_uri_module:n`

```
1390 \cs_new:Nn \stex_symbol_uri_module:N {
1391   \exp_after:wN \__stex_uris_first_three:nnnn #1
1392 }
1393 \cs_new:Nn \stex_symbol_uri_module:n {
1394   \__stex_uris_first_three:nnnn #1
1395 }
1396
1397 \cs_new:Nn \__stex_uris_first_three:nnnn {
1398   #{#1}#{#2}#{#3}
1399 }
```

(End of definition for `\stex_symbol_uri_module:N` and `\stex_symbol_uri_module:n`. These functions are documented on page 129.)

sfunction

`\stex_symbol_uri_name:N`

`\stex_symbol_uri_name:n`

```
1400 \cs_new:Nn \stex_symbol_uri_name:N {
1401   \exp_after:wN \use_iv:nnnn #1
1402 }
1403 \cs_new:Nn \stex_symbol_uri_name:n {
1404   \use_iv:nnnn #1
1405 }
```

(End of definition for `\stex_symbol_uri_name:N` and `\stex_symbol_uri_name:n`. These functions are documented on page 129.)

sfunction

Chapter 30

Documents

```

1406 (@@=stex_doc)

\STEXftmllink
\STEXlinkftml
\STEXsetlinkftml
1407 \cs_new_protected:Npn \STEXftmllink {
1408   \exp_args:Np \url{
1409     \stex_use_document_uri:N \l_stex_current_document_uri
1410   }
1411 }
1412
1413 \cs_new_protected:Npn \STEXsetlinkftml #1 {
1414   \tl_gset:Nn \g__stex_doc_ftml_link_text_tl { #1 }
1415 }
1416
1417 \tl_gset:Nn \g__stex_doc_ftml_link_text_tl{
1418   The~
1419   \stex_get_symbol:nF{FTML}{~}
1420   \tl_if_empty:NTF \l_stex_get_symbol_uri {FTML}{\sn{FTML}}
1421   {~ version ~ of ~ this ~ document ~ can ~ be ~ found ~ at\\
1422   \STEXftmllink
1423 }
1424
1425 \NewDocumentCommand \STEXlinkftml { 0{ } } {
1426   \ifstexhtml\else
1427     \begin{center}
1428       \emph{
1429         \tl_if_empty:NTF{#1}\g__stex_doc_ftml_link_text_tl{#1}
1430       }
1431     \end{center}
1432   \fi
1433 }
```

(End of definition for \STEXftmllink, \STEXlinkftml, and \STEXsetlinkftml. These functions are documented on page 130.)

sfunction

\stexdoctitle Initial definition (before \begin{document}):

```

1434 \tl_new:N \g__stex_doc_title_tl
1435
1436 \str_new:N \g_stex_document_kind_str
```

```

1437 \tl_new:N \g_stex_document_kind_parameters_tl
1438 \str_set:Nn \g_stex_document_kind_str{article}
1439 \stex_if_html_backend:T{
1440   \AtBeginDocument{
1441     \exp_args:Nne\stex_html_node:nnn{span}{
1442       data-ftml-document-kind=\g_stex_document_kind_str
1443       \g_stex_document_kind_parameters_tl
1444     }{}
1445   }
1446 }
1447
1448 \cs_new_protected:Npn \stexdoctitle #1 {
1449   \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1450   \global\def\stexdoctitle##1{}
1451 }

  At begin document, we switch to:

1452 \cs_new_protected:Nn \__stex_doc_set_title:n {
1453   \stex_if_smsmode:F{
1454     \gdef\stexdoctitle##1{}
1455     \stex_debug:nn{title}{Setting~title~to:~\tl_to_str:n{#1}}
1456     \tl_gset:Nn \g__stex_doc_title_tl { #1 }
1457     \__stex_doc_title_html:
1458   }
1459 }
1460
1461 \cs_new_protected:Nn \__stex_doc_title_html: {
1462   \stex_if_do_html:T{
1463     \stex_annotate_invisible:nn{data-ftml-doctitle={}}{ \hbox{\g__stex_doc_title_tl} }
1464   }
1465 }
1466
1467 \AtBeginDocument {
1468   \tl_if_empty:NTF \g__stex_doc_title_tl {
1469     \cs_set_eq:NN \stexdoctitle \__stex_doc_set_title:n
1470   }{
1471     \gdef\stexdoctitle#1{}
1472     \__stex_doc_title_html:
1473   }
1474
1475   \cs_set_eq:NN \__stex_doc_maketitle: \maketitle
1476   \protected\gdef\maketitle{
1477     \tl_if_empty:NF \@title {
1478       \exp_args:No \stexdoctitle \@title
1479     }
1480     \__stex_doc_maketitle:
1481   }
1482 }

```

(End of definition for \stexdoctitle. This function is documented on page 130.)

sfunction

30.1 Sectioning

`\l_stex_current_section_level_int`
`\l_stex_current_section_level_str`

```
1483 \int_new:N \l_stex_current_section_level_int
1484 \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 131.)

`\currentsectionlevel`
`\Currentsectionlevel`

```
1485 \cs_set_protected:Npn \currentsectionlevel {
1486   \stex_if_do_html:TF{
1487     \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
1488     \stex_annotate:nn{data-ftml-currentsectionlevel={},data-ftml-capitalize=false}{}
1489   }{
1490     \l_stex_current_section_level_str
1491   }
1492   \tl_if_exist:NT\xspace\xspace
1493 }
1494
1495 \cs_set_protected:Npn \Currentsectionlevel {
1496   \stex_if_do_html:TF{
1497     \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
1498     \stex_annotate:nn{data-ftml-currentsectionlevel={},data-ftml-capitalize=true}{}
1499   }{
1500     \exp_after:wN \str_uppercase:n \l_stex_current_section_level_str
1501   }
1502   \tl_if_exist:NT\xspace\xspace
1503 }
```

(End of definition for `\currentsectionlevel` and `\Currentsectionlevel`. These functions are documented on page 131.)

sfunction

`sfragment (env.)`

```
1504 \stex_keys_define:nnnn{ sfragment }{
1505   \tl_clear:N \l_stex_key_short_tl
1506 }{
1507   short .tl_set:N = \l_stex_key_short_tl
1508 }{id}
1509
1510 \NewDocumentEnvironment{sfragment}{0}{} m){
1511   \stex_keys_set:nn{sfragment}{#1}
1512   \__stex_doc_do_section:n{#2}
1513   \stex_do_id: % TODO
1514 }{
1515   \_sfragment_end:
1516 }
1517
1518 \cs_new_protected:Npn \__stex_doc_do_section:n {
1519   \int_case:nnF \l_stex_current_section_level_int {
1520     {0}{\cs_if_exist:NTF \thepart {\_sfragment_do_level:nn{part}}{
1521       \int_incr:N \l_stex_current_section_level_int
1522       \__stex_doc_do_section:n
```

```

1523     }}
1524     {1}{\cs_if_exist:NTF \thechapter {\_sfragment_do_level:nn{chapter}}{
1525         \int_incr:N \l_stex_current_section_level_int
1526         \__stex_doc_do_section:n
1527     }}
1528     {2}{\_sfragment_do_level:nn{section}}
1529     {3}{\_sfragment_do_level:nn{subsection}}
1530     {4}{\_sfragment_do_level:nn{subsubsection}}
1531     {5}{\_sfragment_do_level:nn{paragraph}}
1532 }{\_sfragment_do_level:nn{subparagraph}}
1533 }
1534
1535 \stex_if_html_backend:TF {
1536     \cs_new_protected:Nn \_sfragment_do_level:nn {
1537         \stexdoctitle{#2}
1538         \par
1539         \begin{stex_env_node}{section}{data-ftml-section={\int_use:N \l_stex_current_section_level_int}
1540             \noindent\stex_html_node:nnn{h1}{data-ftml-title={}}{
1541                 \stex_annotate_force_break:n{#2}
1542             }\par
1543         }
1544         \cs_new_protected:Nn \_sfragment_end: {
1545             \end{stex_env_node}
1546         }
1547     }{
1548         \cs_new_protected:Nn \_sfragment_do_level:nn {
1549             \stexdoctitle{#2}
1550             \tl_if_empty:NTF \l_stex_key_short_tl {
1551                 \use:c{#1}
1552             }{
1553                 \exp_args:Nne \use:nn{\use:c{#1}}{[\exp_args:No \exp_not:n \l_stex_key_short_tl]}
1554             }{#2}
1555
1556             \int_compare:nNnF \l_stex_current_section_level_int = 6{
1557                 \int_incr:N \l_stex_current_section_level_int
1558             }
1559             \tl_set:Nn\l_stex_current_section_level_str{#1}
1560         }
1561         \cs_new_protected:Nn \_sfragment_end: {
1562     }

```

`titlefragment (env.)`

```

1563 \bool_new:N \l__stex_doc_titlefragment_bool
1564
1565 \NewDocumentEnvironment{titlefragment}{0}{m}{
1566     \stex_keys_set:nn{sfragment}{#1}
1567     \cs_if_exist:NTF \maketitle {
1568         \bool_set_true:N \l__stex_doc_titlefragment_bool
1569         \title{#2}
1570         \tl_if_empty:NF \l_stex_key_short_tl {
1571             \cs_if_exist:NT \shorttitle {
1572                 \exp_args:No \shorttitle \l_stex_key_short_tl
1573             }
1574         }

```

```

1575     \stexdoctitle{#2}
1576     \maketitle
1577   }{
1578     \bool_set_false:N \l__stex_doc_titlefragment_bool
1579     \__stex_doc_do_section:n{#2}
1580     \_stex_do_id:
1581   }
1582 }{
1583   \bool_if:NF \l__stex_doc_titlefragment_bool \_sfragment_end:
1584 }

```

blindfragment (*env.*)

```

1585 \cs_new_protected:Nn \__stex_doc_skip_section_i: {
1586   \int_case:nn \l_stex_current_section_level_int {
1587     {0}{\cs_if_exist:NF \thepart {
1588       \int_incr:N \l_stex_current_section_level_int \__stex_doc_skip_section_i:
1589     }}
1590     {1}{\cs_if_exist:NF \thechapter {
1591       \int_incr:N \l_stex_current_section_level_int \__stex_doc_skip_section_i:
1592     }}
1593   }
1594   \int_compare:nNnF \l_stex_current_section_level_int = 6{
1595     \int_incr:N \l_stex_current_section_level_int
1596   }
1597 }
1598
1599 \stex_if_html_backend:TF {
1600   \cs_new_protected:Nn \__stex_doc_skip_section: {
1601     \__stex_doc_skip_section_i:
1602     \begin{stex_annotate_env}{data-ftml-skipsection={\int_use:N \l_stex_current_section_level_int}}
1603     \_stex_annotate_force_break:n{}
1604   }
1605 }{
1606   \cs_set_eq:NN \__stex_doc_skip_section: \__stex_doc_skip_section_i:
1607 }
1608
1609 \NewDocumentEnvironment{blindfragment}{}{
1610   \__stex_doc_skip_section:
1611 }{
1612   \stex_if_html_backend:T{
1613     \stex_annotate_invisible:n{~}
1614     \end{stex_annotate_env}
1615   }
1616 }

```

\skipfragment

```

1617 \cs_new_protected:Npn \skipfragment {
1618   \int_case:nnF \l_stex_current_section_level_int {
1619     {0}{\cs_if_exist:NTF \thepart {\stepcounter{part}}{
1620       \int_incr:N \l_stex_current_section_level_int
1621       \skipfragment
1622     }}
1623     {1}{\cs_if_exist:NTF \thechapter {\stepcounter{chapter}}{
1624       \int_incr:N \l_stex_current_section_level_int

```

```

1625     \skipfragment
1626   }}
1627   {2}{\stepcounter{section}}
1628   {3}{\stepcounter{subsection}}
1629   {4}{\stepcounter{subsubsection}}
1630   {5}{\stepcounter{paragraph}}
1631   }{\stepcounter{subparagraph}}
1632 }

```

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

sfunction

\setsectionlevel

```

1633 \cs_new_protected:Npn \setsectionlevel #1 {
1634   \str_case:nnF{#1}{
1635     {part}{\int_set:Nn \l_stex_current_section_level_int 0}
1636     {chapter}{\int_set:Nn \l_stex_current_section_level_int 1}
1637     {section}{\int_set:Nn \l_stex_current_section_level_int 2}
1638     {subsection}{\int_set:Nn \l_stex_current_section_level_int 3}
1639     {subsubsection}{\int_set:Nn \l_stex_current_section_level_int 4}
1640     {paragraph}{\int_set:Nn \l_stex_current_section_level_int 5}
1641   }{
1642     \int_set:Nn \l_stex_current_section_level_int 6
1643   }
1644   \stex_if_do_html:T{
1645     \cs_if_eq:NNTF\@onlypreamble\@notprerr{
1646       \stex_html_literal:n{<span~ data-ftml-sectionlevel="\int_use:N\l_stex_current_section_level_int}
1647     }{
1648       \AtBeginDocument{
1649         \stex_html_literal:n{<span~ data-ftml-sectionlevel="\int_use:N\l_stex_current_section_level_int}
1650       }
1651     }
1652   }
1653 }

```

In HTML mode, we make sure to record the top section level:

```

1654 \stex_if_html_backend:T{
1655   \cs_new_protected:Nn \__stex_doc_check_topsect: {
1656     \int_case:nnF \l_stex_current_section_level_int {
1657       {0}{\cs_if_exist:NTF \thepart {
1658         \stex_annotate_invisible:nn{data-ftml-sectionlevel=0}{}}
1659       }{
1660         \int_incr:N \l_stex_current_section_level_int
1661         \__stex_doc_check_topsect:
1662       }}
1663     {1}{\cs_if_exist:NTF \thechapter {
1664       \stex_annotate_invisible:nn{data-ftml-sectionlevel=1}{}}
1665     }{
1666       \int_incr:N \l_stex_current_section_level_int
1667       \__stex_doc_check_topsect:
1668     }}
1669   }{
1670     \stex_annotate_invisible:nn{data-ftml-sectionlevel={\int_use:N\l_stex_current_section_level_int}}
1671   }
1672 }

```



```

1673 \AtBeginDocument{\__stex_doc_check_topsect:
1674 \cs_if_exist:NTF \thechapter {
1675 \int_case:nnT \l_stex_current_section_level_int{
1676 {0}{ }
1677 {1}{ }
1678 }{
1679 \stex_css_literal:n {
1680 .ftml-section {
1681 &~.ftml-title-section { &::before {
1682 content:~counter(ftml-chapter)~"."~counter(ftml-section)~"";
1683 } }
1684 }
1685 .ftml-subsection {
1686 &~.ftml-title-subsection { &::before {
1687 content:~counter(ftml-chapter)~"."~counter(ftml-section)~"."~counter(ftml-sub
1688 } }
1689 }
1690 .ftml-subsubsection {
1691 &~.ftml-title-subsubsection { &::before {
1692 content:~counter(ftml-chapter)~"."~counter(ftml-section)~"."~counter(ftml-sub
1693 } }
1694 }
1695 }
1696 }
1697 }
1698 }
1699 }

```

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

sfunction

We make sure `\frontmatter` and `\backmatter` are always defined. **TODO: this seems like a hack and probably shouldn't be here**

```

1700 \bool_if:NF \c_stex_no_frontmatter_bool {
1701 \cs_new_protected:Nn \__stex_doc_x_matter: {
1702 \cs_if_exist:NTF\frontmatter{
1703 \let\__stex_doc_orig_frontmatter\frontmatter
1704 \let\frontmatter\relax
1705 }{
1706 \tl_set:Nn\__stex_doc_orig_frontmatter{
1707 \clearpage
1708 %\@mainmatterfalse
1709 \pagenumbering{roman}
1710 }
1711 }
1712 \cs_if_exist:NTF\backmatter{
1713 \let\__stex_doc_orig_backmatter\backmatter
1714 \let\backmatter\relax
1715 }{
1716 \tl_set:Nn\__stex_doc_orig_backmatter{
1717 \clearpage
1718 %\@mainmatterfalse
1719 \pagenumbering{roman}
1720 }
1721 }

```

```

1722 \newenvironment{frontmatter}{
1723   \__stex_doc_orig_frontmatter
1724 }{
1725   \cs_if_exist:NTF\mainmatter{
1726     \mainmatter
1727   }{
1728     \clearpage
1729     %\@mainmattertrue
1730     \pagenumbering{arabic}
1731   }
1732 }
1733 \newenvironment{backmatter}{
1734   \__stex_doc_orig_backmatter
1735 }{
1736   \cs_if_exist:NTF\mainmatter{
1737     \mainmatter
1738   }{
1739     \clearpage
1740     %\@mainmattertrue
1741     \pagenumbering{arabic}
1742   }
1743 }
1744 }
1745 \AtBeginDocument \__stex_doc_x_matter:
1746 }

```

30.2 Inter-Document References

```

1747 <@@=stex_refs>

The .sref-file:
1748 \iow_new:N \c__stex_refs_iow
1749 \AtBeginDocument{ \iow_open:Nn \c__stex_refs_iow {\jobname.sref} }
1750 \AtEndDocument{\iow_close:N \c__stex_refs_iow}
1751 \str_set:Ne \c__stex_refs_iow_str { \stex_file_use:N \c_stex_pwd_file / }

```

Keys:

```

1752 \stex_keys_define:nnnn{sref / 1}{
1753   \tl_clear:N \l_stex_key_pre_tl
1754   \tl_clear:N \l_stex_key_post_tl
1755   \tl_clear:N \l_stex_key_fallback_tl
1756 }{
1757   % TODO get rid of this
1758   fallback .code:n = {},
1759   pre      .code:n = {},
1760   post     .code:n = {}
1761 }{archive file}
1762 \stex_keys_define:nnnn{sref / 2}{-}{-}{archive file, title}

```

Target declarations:

\sreflabel

\stex_ref_new_doc_target:n

```

1763 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1764   \stex_if_smsmode:F{
1765     \tl_set:Ne \l__stex_refs_new_tl { \stex_new_document_element_uri:n { #1 } }

```

```

1766 \exp_args:Ne \label { \stex_use_document_uri:N \l__stex_refs_new_tl }
1767 \exp_args:Nne \STeXInternalNewSRefLabel{#1}{ \stex_use_document_uri:N \l__stex_refs_new
1768 \iow_now:Ne \@auxout {
1769   \STeXInternalNewSRefLabel{#1}{ \stex_use_document_uri:N \l__stex_refs_new_tl }
1770 }
1771 \iow_now:Ne \c__stex_refs_iow {
1772   \exp_not:N \STeXInternalSRefLabel
1773   { #1 }
1774   { \stex_use_document_uri:N \l__stex_refs_new_tl }
1775   { \exp_not:o \@currentHref}
1776   { \exp_not:o \@currentlabelname }
1777 }
1778 }
1779 }
1780
1781 \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 79.)

sfunction

\stex_ref_new_sym_target:n

```

1782 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1783   \stex_if_smsmode:F{
1784     \cs_if_exist:cTF{r@\tl_to_str:n{#1}}{
1785       \cs_if_exist:cT{__stex_sref_aux_sym_ \tl_to_str:n{#1}}{
1786         \cs_undefine:c{__stex_sref_aux_sym_ \tl_to_str:n{#1}}
1787         \exp_args:Ne\label { \tl_to_str:n{#1} }
1788       }
1789     }{ \exp_args:Ne\label { \tl_to_str:n{#1} } }
1790   }
1791 }

```

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

sfunction

Inserting References:

\sref

```

1792 \NewDocumentCommand \sref { 0{} m 0{} }{
1793   \stex_keys_set:nn { sref / 1 }{ #1 }
1794   \tl_clear:N \l__stex_refs_tmp
1795   \str_if_empty:NTF \l_stex_key_file_str {
1796     \__stex_refs_local:n
1797   }{
1798     \__stex_refs_remote:nn{#3}
1799   }{#2}
1800 }
1801
1802 \cs_new_protected:Nn \__stex_refs_local:n {
1803   \stex_debug:nn{sref}{Checking-#1}
1804   \tl_if_exist:cTF{ g_stex_sref_label_ #1 }{
1805     %% TODO INSERT FTML ANNOTATION
1806     \exp_args:Ne \__stex_refs_local_ref:n {\use:c{ g_stex_sref_label_ #1 }}
1807   }{

```

```

1808 \tl_if_empty:NTF \l_stex_key_fallback_tl { ??? }{ \l_stex_key_fallback_tl }
1809 }
1810 }
1811
1812 \cs_new_protected:Nn \__stex_refs_local_ref:n {
1813 \cs_if_exist:cTF{autoref}{
1814 \l_stex_key_pre_tl \autoref{#1} \l_stex_key_post_tl
1815 }{
1816 \message{^^J^^J
1817 sTeX~Warning: \c_backslash_str sref ~ with ~ local ~ target ~ used ~ without ~ hyperr
1818 ^^J^^J}
1819 \l_stex_key_pre_tl \ref{#1} \l_stex_key_post_tl
1820 }
1821 }
1822
1823 \cs_new_protected:Nn \__stex_refs_remote:nn {
1824 \str_set:Nn \l__stex_refs_key { #2 }
1825 \tl_set:Nn \l__stex_refs_in { #1 }
1826 \exp_args:No \stex_in_archive:nn \l_stex_key_archive_str {
1827 \exp_args:NNo \stex_document_uri_from_archive_file:Nn \l__stex_refs_uri {\l_stex_key_fi
1828 \stex_debug:nn{sref}{Checking~#2~from~\stex_use_document_uri:N \l__stex_refs_uri}
1829 \tl_if_eq:NNTF \l__stex_refs_uri \c_stex_main_document_uri {
1830 \stex_debug:nn{sref}{From~current~document;~delegating~to~\string\autoref}
1831 \__stex_refs_local:n{ #2 }
1832 }{
1833 \str_set:Ne \l__stex_refs_uri { \stex_use_document_uri:N \l__stex_refs_uri }
1834 \str_set:Ne \l__stex_refs_file { \exp_args:No \stex_source_path:n {\l_stex_key_file_s
1835 \exp_args:No \IfFileExists \l__stex_refs_file {
1836 \__stex_refs_find:
1837 }{ }
1838 \tl_if_empty:NTF \l__stex_refs_tmp {
1839 \stex_debug:nn{sref}{Not~found}
1840 \tl_if_empty:NTF \l_stex_key_fallback_tl { ??? }{ \l_stex_key_fallback_tl }
1841 }{
1842 \str_set_eq:NN \l__stex_refs_uri \l__stex_refs_tmp
1843 \stex_debug:nn{sref}{Found~ \l__stex_refs_uri}
1844 \__stex_refs_maybe_in:
1845 }
1846 }
1847 }
1848 }
1849
1850 \cs_new_protected:Nn \__stex_refs_find: {
1851 \setbox0\vbox\bgroup
1852 \cs_set_eq:NN \STeXInternalSRefLabel \__stex_refs_check_i:nnnn
1853 \use:c{@ @ input}{\l__stex_refs_file}
1854 \exp_args:NNe \use:nn
1855 \egroup {
1856 \tl_set:Nn \exp_not:N \l__stex_refs_tmp { \l__stex_refs_tmp }
1857 }
1858 }
1859
1860 \cs_new_protected:Nn \__stex_refs_check_i:nnnn {
1861 \exp_args:No \str_if_eq:nnT{ \l__stex_refs_key }{#1}{

```

```

1862     \exp_args:Nno \stex_str_if_starts_with:nnT{#2}{\l__stex_refs_uri}{
1863       \str_set:Nn \l__stex_refs_tmp {#2}
1864       \endinput
1865     }
1866   }
1867 }
1868
1869 \cs_new_protected:Nn \__stex_refs_maybe_in: {
1870   \tl_if_empty:NTF\l__stex_refs_in {
1871     \cs_if_exist:cTF{r@\l__stex_refs_uri}{
1872       \exp_args:No \__stex_refs_local_ref:n \l__stex_refs_uri
1873     }{
1874       \tl_if_empty:NTF \l__stex_refs_default_file {
1875         \exp_args:No \__stex_refs_local_ref:n \l__stex_refs_uri
1876       }{
1877         \tl_set_eq:NN \l_stex_key_title_tl \l__stex_refs_default_title
1878         \str_set_eq:NN \l__stex_refs_file \l__stex_refs_default_file
1879         \str_if_eq:NNTF \l__stex_refs_default_file \c__stex_refs_iow_str {
1880           \exp_args:No \__stex_refs_local_ref:n \l__stex_refs_uri
1881         }{
1882           \stex_debug:nn{sref}{in~title=\exp_args:No\exp_not:n\l_stex_key_title_tl,file=\l__
1883             \__stex_refs_in:
1884           }
1885         }
1886       }
1887     }{
1888       \stex_debug:nn{sref}{in~\exp_args:No\exp_not:n\l__stex_refs_in}
1889       \exp_args:No \stex_in_archive:nn \l_stex_key_archive_str {
1890         \str_set:Ne \l__stex_refs_file { \exp_args:No \stex_source_path:n {\l_stex_key_file_s
1891       }
1892       \exp_args:Nno \stex_keys_set:nn{ sref / 2 }{ \l__stex_refs_in }
1893       \__stex_refs_in:
1894     }
1895   }
1896
1897 \cs_new_protected:Nn \__stex_refs_in: {
1898   \tl_clear:N \l__stex_refs_tmp
1899   \exp_args:No \IfFileExists \l__stex_refs_file {
1900     \__stex_refs_find_in:
1901   }{
1902     \stex_debug:nn{sref}{not~found:~\l__stex_refs_file}
1903   }
1904   \tl_if_empty:NTF\l__stex_refs_tmp{
1905     \stex_debug:nn{sref}{Not~found}
1906     \tl_if_empty:NTF \l_stex_key_fallback_tl { ??? }{ \l_stex_key_fallback_tl }
1907   }{
1908     \stex_debug:nn{sref}{found:~\meaning\l__stex_refs_tmp}
1909     \tl_set:Ne \l__stex_refs_tmp {
1910       \exp_after:wN \__stex_refs_in_text:nn \l__stex_refs_tmp
1911     }
1912     \cs_if_exist:NT \href { \exp_args:No \href \l__stex_refs_uri } \l__stex_refs_tmp
1913   }
1914 }
1915

```

```

1916 \cs_new:Nn \__stex_refs_in_text:nn {
1917   \__stex_refs_in_text:w #1 \__stex_refs_stop:
1918   \tl_if_empty:nF{#2}{~(\exp_not:n{#2})}
1919   \tl_if_empty:NF \l_stex_key_title_tl {
1920     {}~in~\exp_args:No \exp_not:n \l_stex_key_title_tl
1921   }
1922 }
1923
1924 \cs_new:Npn \__stex_refs_in_text:w #1.#2 \__stex_refs_stop: {
1925   \__stex_refs_uppercase:n #1~#2~
1926 }
1927
1928 \cs_new:Nn \__stex_refs_uppercase:n { \uppercase{#1}}
1929
1930 \cs_new_protected:Nn \__stex_refs_find_in: {
1931   \stex_debug:nn{sref}{finding~\l__stex_refs_uri;~in~\l__stex_refs_file...?}
1932   \setbox0\vbox\bgroup
1933   \catcode'\^J=9\relax
1934   \cs_set_eq:NN \STeXInternalSRefLabel \__stex_refs_check_in:nnnn
1935   \use:c{@ @ input}{\l__stex_refs_file}
1936   \exp_args:NNe \use:nn
1937   \egroup {
1938     \tl_set:Nn \exp_not:N \l__stex_refs_tmp { \exp_args:No \exp_not:n \l__stex_refs_tmp }
1939   }
1940 }
1941
1942 \cs_new_protected:Nn \__stex_refs_check_in:nnnn {
1943   \exp_args:No \str_if_eq:nnT{ \l__stex_refs_uri }{#2}{
1944     \tl_set:Nn \l__stex_refs_tmp { {#3}{#4} }
1945   }
1946   \endinput
1947 }

```

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

sfunction
 TODO

```

1948 %\int_new:N \l__stex_refs_unnamed_counter_int
1949
1950
1951
1952
1953 \cs_new:Npn \STeXInternalSRefLabel #1 #2 #3 #4 {}
1954
1955 \cs_new_protected:Npn \STeXInternalNewSRefLabel #1 #2 {
1956   \tl_gset:ce{ g_stex_sref_label_ #1 }{ \tl_to_str:n{ #2 } }
1957 }
1958
1959 \tl_new:N \l__stex_refs_default_title
1960 \str_new:N \l__stex_refs_default_file
1961 \str_new:N \l__stex_refs_default_archive
1962
1963 \newcommand\srefsetin[3][[]]{
1964   \str_set:Ne \l__stex_refs_default_file { #2 }
1965   \tl_if_empty:nTF{#1}{

```

```

1966 \str_set:Ne \l__stex_refs_default_archive { \stex_archive_id:N \c_stex_main_archive }
1967 }{
1968 \str_set:Nn \l__stex_refs_default_archive { #1 }
1969 }
1970 \exp_args:No \stex_in_archive:nn \l__stex_refs_default_archive {
1971 \str_set:Ne \l__stex_refs_default_file { \exp_args:No \stex_source_path:n {\l_stex_key_
1972 }
1973 \str_set:Nn \l__stex_refs_default_title { #3 }
1974 }
1975 \NewDocumentCommand \extref { 0{} m m}{???}
1976
1977 \cs_new_protected:Nn \stex_ref_new_symbol:n {}
1978
1979
1980 \NewDocumentCommand \srefsym { m m }{
1981 \stex_get_symbol:n { #1 }
1982 \exp_args:Ne \srefsymuri { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }{ #2 }
1983 }
1984
1985 \cs_new_protected:Npn \srefsymuri #1 {
1986 \cs_if_exist:cTF{r@tl_to_str:n{#1}}{
1987 \cs_if_exist:NT \hyperlink { \hyperlink{#1} }
1988 }{
1989 \cs_if_exist:NT \href { \href{#1} }
1990 }
1991 }

```

30.3 Inputting From MathHub Resources

```

1992 <@@=stex_inputs>

\inputref

1993 \NewDocumentCommand \inputref { 0{} m}{
1994 \stex_in_archive:nn{#1}{
1995 \stex_document_uri_from_archive_file:Nn \l__stex_inputs_uri {#2}
1996 \__stex_inputs_ref:n{#2}
1997 }
1998 }
1999
2000 \stex_if_html_backend:TF{
2001 \str_set:Nn \c__stex_inputs_ref_pre_str {<span~data-ftml-inputref="}
2002 \str_set:Nn \c__stex_inputs_ref_post_str {"~><!--<!--></span>}
2003 \cs_new_protected:Nn \__stex_inputs_ref:n {
2004 \IfFileExists{\stex_source_path:n {#1}}{
2005 \relax
2006 \exp_args:Ne\stex_html_literal:n{
2007 \c__stex_inputs_ref_pre_str
2008 \stex_use_document_uri:N\l__stex_inputs_uri
2009 \c__stex_inputs_ref_post_str
2010 }
2011 }{
2012 \stex_input_with_hooks:Ne\l__stex_inputs_uri {\stex_source_path:n {#1}}
2013 }

```

```

2014 }
2015 }{
2016 \cs_new_protected:Nn \__stex_inputs_ref:n {
2017 \begingroup
2018 \inputreftrue
2019 \let \l_stex_metatheory_uri \c_stex_default_metatheory
2020 %\seq_clear:N \l_stex_all_modules_seq
2021 \stex_undefine:N \l_stex_current_module_uri
2022 \stex_debug:nn{inputref}{Inputrefing~document~\stex_use_document_uri:N \l__stex_input
2023 \stex_input_with_hooks:Ne\l__stex_inputs_uri{\stex_source_path:n {#1}}
2024 \endgroup
2025 }
2026 }

```

(End of definition for \inputref. This function is documented on page 132.)

sfunction

\mhinput

```

2027 \NewDocumentCommand \mhinput { 0{} m}{
2028 \stex_in_archive:nn {#1} {
2029 \stex_document_uri_from_archive_file:Nn \l__stex_inputs_uri {#2}
2030 \ifinputref
2031 \stex_input_with_hooks:Ne\l__stex_inputs_uri{\stex_source_path:n {#2}}
2032 \else
2033 \inputreftrue
2034 \stex_input_with_hooks:Ne\l__stex_inputs_uri{\stex_source_path:n {#2}}
2035 \inputreffalse
2036 \fi
2037 }
2038 }

```

(End of definition for \mhinput. This function is documented on page 132.)

sfunction

\ifinputref

```

2039 \newif \ifinputref \inputreffalse

```

(End of definition for \ifinputref. This function is documented on page 132.)

sfunction

\IfInputref

```

2040 \stex_if_html_backend:TF{
2041 \newcommand \IfInputref[2]{
2042 \stex_annotate:nn{data-ftml-ifinputref=true}{#1}
2043 \stex_annotate:nn{data-ftml-ifinputref=false}{#2}
2044 }
2045 }{
2046 \newcommand \IfInputref[2]{
2047 \ifinputref #1 \else #2 \fi
2048 }
2049 }

```

(End of definition for \IfInputref. This function is documented on page 132.)

sfunction

\libinput

```
2050 \newcommand \libinput [2] [] {
2051   \stex_in_archive:nn{#1}{
2052     \__stex_inputs_up_archive:nn{#2}{tex}
2053     \stex_debug:nn{lib}{Result:~\seq_use:Nn \l__stex_inputs_libinput_files_seq ,}
2054     \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
2055       \msg_error:nnnn{stex}{error/nofile}{\libinput}{#2.tex}
2056     }{
2057       \seq_map_function:NN \l__stex_inputs_libinput_files_seq \input
2058     }
2059   }
2060 }
```

(End of definition for \libinput. This function is documented on page 132.)

sfunction

\libusepackage

```
2061 \newcommand\libusepackage[2] []{
2062   \__stex_inputs_up_archive:nn{#2}{sty}
2063   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2064     \str_set:Ne \l__stex_inputs_tmp_str {\seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
2065     \exp_args:Nne \use:n {\usepackage[#1]} {
2066       \str_range:Nnn\l__stex_inputs_tmp_str 1 {-5}
2067     }
2068   }{
2069     \msg_fatal:nnnn{stex}{error/nofile}{\libusepackage}{#2.sty}
2070   }
2071 }
```

(End of definition for \libusepackage. This function is documented on page 132.)

sfunction

\libusetikzlibrary

```
2072 \newcommand \libusetikzlibrary [2] [] {
2073   \cs_if_exist:NF \usetikzlibrary {
2074     \msg_error:nnx{stex}{error/notikz}{\tl_to_str:n{\libusetikzlibrary}}
2075   }
2076   \tl_if_empty:NTF{#1}{
2077     \__stex_inputs_usetikzlibrary:n{#2}
2078   }{
2079     \stex_in_archive:nn{#1}{\__stex_inputs_usetikzlibrary:n{#2}}
2080   }
2081 }
2082
2083 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary:n{
2084   \__stex_inputs_up_archive:nn{tikzlibrary#1}{code.tex}
2085   \int_compare:nNnTF {\seq_count:N \l__stex_inputs_libinput_files_seq} = 1 {
2086     \exp_args:Nne \__stex_inputs_usetikzlibrary_i:nn{#1}{ \seq_item:Nn \l__stex_inputs_libinput_files_seq 1}
2087   }{
2088     \msg_fatal:nnnn{stex}{error/nofile}{\libusetikzlibrary}{tikzlibrary#1.code.tex}
2089   }
2090 }
2091
2092 \cs_new_protected:Nn \__stex_inputs_usetikzlibrary_i:nn {
```

```

2093 \pgfkeys@spdef\pgf@temp{#1}
2094 \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
2095 \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@
2096 \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
2097 \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
2098 \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
2099 \catcode'\@=11
2100 \catcode'\|=12
2101 \catcode'\$=3
2102 \pgfutil@InputIfFileExists{#2}{-}{-}
2103 \catcode'\@=\csname tikz@library@#1@atcode\endcsname
2104 \catcode'\|=\csname tikz@library@#1@barcode\endcsname
2105 \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
2106 }

```

(End of definition for \libusetikzlibrary. This function is documented on page 132.)

sfunction

\addmhbibresource

```

2107 \newcommand \addmhbibresource [2] [] {
2108 \stex_in_archive:nn{#1}{
2109 \__stex_inputs_up_archive:nn{#2}{bib}
2110 \stex_debug:nn{lib}{Result:-\seq_use:Nn \l__stex_inputs_libinput_files_seq ,}
2111 \seq_if_empty:NTF \l__stex_inputs_libinput_files_seq {
2112 \msg_error:nnnn{stex}{error/nofile}{\addmhbibresource}{#2.bib}
2113 }{
2114 \seq_map_function:NN \l__stex_inputs_libinput_files_seq \addbibresource
2115 }
2116 }
2117 }

```

(End of definition for \addmhbibresource. This function is documented on page 133.)

sfunction

__stex_inputs_up_archive:nn Iterates up the current archive's directory in search for lib files with name #1.#2

```

2118 \cs_new_protected:Nn \__stex_inputs_up_archive:nn {
2119 \tl_if_exist:NF \l_stex_current_archive {
2120 \msg_error:nnn{stex}{error/notinarchive}\libinput
2121 }
2122 \seq_clear:N \l__stex_inputs_libinput_files_seq
2123 \seq_set_split:Nne \l__stex_inputs_path_seq / {\stex_archive_path:N \l_stex_current_archi
2124 \seq_set_split:Nne \l__stex_inputs_id_seq / {\stex_archive_id:N \l_stex_current_archive}
2125 \seq_map_inline:Nn \l__stex_inputs_id_seq {
2126 \seq_pop_right:NN \l__stex_inputs_path_seq \l_tmpa_tl
2127 }
2128 \stex_debug:nn{lib}{Checking~#1~in~\seq_use:Nn \l__stex_inputs_path_seq /}
2129
2130 \bool_while_do:nn { ! \seq_if_empty_p:N \l__stex_inputs_id_seq }{
2131 \str_set:N \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / meta-
2132 \stex_debug:nn{lib}{Checking~\l__stex_inputs_path_str}
2133 \IfFileExists{ \l__stex_inputs_path_str }{
2134 \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_
2135 \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
2136 }

```

```

2137     }{}
2138     \seq_pop_left:NN \l__stex_inputs_id_seq \l__stex_inputs_path_str
2139     \seq_put_right:No \l__stex_inputs_path_seq \l__stex_inputs_path_str
2140   }
2141
2142   \str_set:Ne \l__stex_inputs_path_str {\stex_file_use:N \l__stex_inputs_path_seq / lib / #
2143   \stex_debug:nn{lib}{Checking~\l__stex_inputs_path_str}
2144   \IfFileExists{ \l__stex_inputs_path_str }{
2145     \exp_args:NNo \seq_if_in:NnF \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_st
2146     \seq_put_right:No \l__stex_inputs_libinput_files_seq \l__stex_inputs_path_str
2147   }
2148   }{}
2149 }

```

(End of definition for __stex_inputs_up_archive:nn.)

\mhgraphics
\cmhgraphics

```

2150 \ltx@ifpackageloaded{graphicx}{\use:n}{\AtEndOfPackageFile{graphicx}}{
2151   \define@key{Gin}{archive}{
2152     \stex_in_archive:nn{#1}{}
2153     \str_set:Ne \Gin@mhrepos {
2154       \stex_source_path:nn{#1}{}
2155     }
2156   }
2157   \providecommand\mhgraphics[2] [] {
2158     \str_set:Ne \Gin@mhrepos {
2159       \stex_source_path:n{}
2160     }
2161     \setkeys{Gin}{#1}
2162     \includegraphics[#1]{ \Gin@mhrepos / #2 }
2163   }
2164   \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
2165 }

```

(End of definition for \mhgraphics and \cmhgraphics. These functions are documented on page 133.)

sfunction

\lstinputmhlisting
\clstinputmhlisting

```

2166 \ltx@ifpackageloaded{listings}{\use:n}{\AtEndOfPackageFile{listings}}{
2167   \define@key{lst}{archive}{
2168     \stex_in_archive:nn{#1}{}
2169     \str_set:Ne \lst@mhrepos{
2170       \stex_source_path:nn{#1}{}
2171     }
2172   }
2173   \newcommand\lstinputmhlisting[2] [] {%
2174     \str_set:Ne \lst@mhrepos{
2175       \stex_source_path:n{}
2176     }
2177     \setkeys{lst}{#1}%
2178     \lstinputlisting[#1]{\lst@mhrepos / #2}}
2179   \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
2180 }

```

(End of definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 133.)

sfunction

```

\mhtikzinput
\cmhtikzinput
2181 \ltx@ifpackageloaded{tikzinput}{\use:n}{\AtEndOfPackageFile{tikzinput}}{
2182   \define@key{Gin}{archive}{
2183     \stex_in_archive:nn{#1}{}
2184     \str_set:Ne \Gin@mhrepos{
2185       \stex_source_path:nn{#1}{}
2186     }
2187     \str_set:Ne \l__stex_inputs_gin_repo_str {#1}
2188   }
2189   \newcommand\mhtikzinput[2][]{%
2190     \str_clear:N \l__stex_inputs_gin_repo_str
2191     \str_set:Ne \Gin@mhrepos{
2192       \stex_source_path:n{ }
2193     }
2194     \setkeys{Gin}{#1}%
2195     \exp_args:No \stex_in_archive:nn \l__stex_inputs_gin_repo_str {
2196       \tikzinput[#1]{\Gin@mhrepos / #2}
2197     }
2198   }
2199   \newcommand\cmhtikzinput[2][]{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
2200 }

```

(End of definition for `\mhtikzinput` and `\cmhtikzinput`. These functions are documented on page 133.)

sfunction

Chapter 31

Modules

```
2201 (@@=stex_modules)
```

A module consists of four separate macros, storing its *symbols*, *incoming morphisms* (e.g. `\importmodules`), *notations*, and “initialization code”, respectively:

```
2202 \cs_new:Nn \__stex_modules_macro_short:nnn {
2203   #1 ? #2 ? #3
2204 }
2205 \cs_new:Nn \_stex_symbol_macro:N {
2206   c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _symbols_prop
2207 }
2208 \cs_new:Nn \_stex_symbol_macro:n {
2209   c_stex_module_ \__stex_modules_macro_short:nnn #1 _symbols_prop
2210 }
2211 \cs_new:Nn \_stex_morphisms_macro:N {
2212   c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _morphisms_prop
2213 }
2214 \cs_new:Nn \_stex_morphisms_macro:n {
2215   c_stex_module_ \__stex_modules_macro_short:nnn #1 _morphisms_prop
2216 }
2217 \cs_new:Nn \_stex_notations_macro:N {
2218   c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _notations_prop
2219 }
2220 \cs_new:Nn \_stex_notations_macro:n {
2221   c_stex_module_ \__stex_modules_macro_short:nnn #1 _notations_prop
2222 }
2223 \cs_new:Nn \_stex_module_code_macro:N {
2224   c_stex_module_ \exp_after:wN \__stex_modules_macro_short:nnn #1 _code
2225 }
2226 \cs_new:Nn \_stex_module_code_macro:n {
2227   c_stex_module_ \__stex_modules_macro_short:nnn #1 _code
2228 }
```

`\l_stex_current_module_uri` initially undefined.

(End of definition for `\l_stex_current_module_uri`. This variable is documented on page 134.)

`\l_stex_all_modules_seq`

```
2229 \seq_new:N \l_stex_all_modules_seq
```

(End of definition for `\l_stex_all_modules_seq`. This variable is documented on page 134.)

smodule (*env*.)

```

2230 \tl_new:N \l_stex_metatheory_uri
2231
2232 \stex_keys_define:nnnn{smodule}{
2233   \str_clear:N \l_stex_key_sig_str
2234 }{
2235   meta .code:n = {
2236     \tl_if_empty:nTF {#1}{
2237       \tl_clear:N \l_stex_metatheory_uri
2238     }{
2239       \stex_uri_from_pair:Nn \l_stex_metatheory_uri { #1 }
2240     }
2241   },
2242   %ns .code:n = {
2243     % \stex_uri_resolve:Ne \l_stex_current_ns_uri { #1 }
2244     %} ,
2245   %lang .code:n = {
2246     % \stex_set_language:n { #1 }
2247     %} ,
2248   sig .str_set_x:N = \l_stex_key_sig_str ,
2249   %creators .code:n = {} , % todo ?
2250   %contributors .code:n = {} , % todo ?
2251 }{id, title, style, deprecate}
2252
2253 \stex_if_html_backend:T {
2254   \cs_new_protected:Nn \__stex_modules_html_annots: {
2255     \exp_args:Ne \stex_annotate_env {
2256       data-ftml-module={\stex_module_uri_name:N \l_stex_current_module_uri}
2257       %data-ftml-language={ \l_stex_current_language_str},
2258       \str_if_empty:NF\l_stex_key_sig_str {, data-ftml-signature={\l_stex_key_sig_str}}
2259       \tl_if_empty:NF \l_stex_metatheory_uri {,
2260         data-ftml-metatheory={\stex_use_module_uri:N \l_stex_metatheory_uri}
2261       }
2262     }
2263     \stex_annotate_invisible:n{}
2264   }
2265 }
2266
2267 \stex_new_stylable_env:nnnnnn {module} {0{} m}{
2268   \stex_keys_set:nn { smodule }{ #1 }
2269   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
2270   \tl_if_empty:NF \thistitle {
2271     \exp_args:No \stexdoctitle \thistitle
2272   }
2273   \stex_module_setup:n { #2 }
2274
2275   \stex_if_do_html:T \__stex_modules_html_annots:
2276   \stex_if_smsmode:F {
2277     \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_current_module_uri }
2278     \str_set:Nn \thismodulename {#2}
2279     \stex_style_apply:
2280   }
2281   \stex_smsmode_do:
2282 }{

```

```

2283 \stex_close_module:
2284 \stex_if_smsmode:F \stex_style_apply:
2285 \stex_if_do_html:T{ \endstex_annotate_env }
2286 }{}{}{s}

```

\stex_module_setup:n

```

2287 \cs_new_protected:Npn \stex_module_setup:n {
2288   \stex_if_in_module:TF \__stex_modules_nested:n \__stex_modules_top:n
2289 }
2290
2291 \cs_new_protected:Nn \__stex_modules_top:n {
2292   \str_if_empty:NTF \l_stex_key_sig_str
2293     \stex_module_setup_top_nosig:n \__stex_modules_top_sig:n {#1}
2294   \g_stex_every_module_tl
2295   \tl_if_empty:NF \l_stex_metatheory_uri {
2296     \stex_debug:nn{modules}{loading~metatheory~\stex_use_module_uri:N \l_stex_metatheory_uri}
2297     \__stex_modules_load_meta:
2298   }
2299 }

```

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

sfunction

\stex_module_setup_top_nosig:n

```

2300 \cs_new_protected:Nn \stex_module_setup_top_nosig:n {
2301   \stex_metagroup_new:
2302   \tl_set:Ne \l__stex_modules_uri { \stex_new_module_uri:n {#1} }
2303   \stex_debug:nn{module}{URI:\stex_use_module_uri:N \l__stex_modules_uri}
2304   \exp_args:No \stex_if_module_exists:NTF \l__stex_modules_uri {
2305     \stex_debug:nn{modules}{(already exists)}
2306   }{
2307     \tl_gclear:c{ \stex_module_code_macro:N \l__stex_modules_uri }
2308     \prop_gclear:c{ \stex_morphisms_macro:N \l__stex_modules_uri }
2309     \prop_gclear:c{ \stex_symbol_macro:N \l__stex_modules_uri }
2310     \prop_gclear:c{ \stex_notations_macro:N \l__stex_modules_uri }
2311   }
2312   \tl_set_eq:NN \l_stex_current_module_uri \l__stex_modules_uri
2313   \seq_put_right:No \l_stex_all_modules_seq \l__stex_modules_uri
2314 }

```

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

sfunction

__stex_modules_load_meta: loads the metatheory:

```

2315 \bool_new:N \l_stex_in_meta_bool
2316
2317 \cs_new_protected:Nn \__stex_modules_load_meta: {
2318   \exp_after:wN \stex_execute_in_module:n \exp_after:wN {
2319     \exp_after:wN \__stex_modules_load_meta_i:n \exp_after:wN { \l_stex_metatheory_uri }
2320   }
2321 }
2322
2323 \cs_new_protected:Nn \__stex_modules_load_meta_i:n {
2324   \bool_if:NTF \l_stex_in_meta_bool {

```

```

2325 \stex_activate_module:n {#1}
2326 }{
2327 \bool_set_true:N \l_stex_in_meta_bool
2328 \stex_activate_module:n {#1}
2329 \bool_set_false:N \l_stex_in_meta_bool
2330 }
2331 }

```

(End of definition for _stex_modules_load_meta:.)

_stex_modules_top_sig:n sets up a new module with a signature:

```

2332 \cs_new_protected:Nn \_stex_modules_top_sig:n {
2333 \stex_debug:nn{modules}{Signature:\l_stex_key_sig_str}
2334 \stex_metagroup_new:
2335 \tl_set:Nc \l__stex_modules_uri { \stex_new_module_uri:n {#1} }
2336 \stex_debug:nn{module}{URI:\stex_use_module_uri:N \l__stex_modules_uri}
2337 \stex_if_module_exists:NTF \l__stex_modules_uri {
2338 \stex_debug:nn{modules}{(already exists)}
2339 \stex_activate_module:N \l__stex_modules_uri
2340 }{
2341 \stex_debug:nn{modules}{(needs loading)}
2342 \_stex_modules_load_sig:
2343 }
2344 \tl_set_eq:NN \l_stex_current_module_uri \l__stex_modules_uri
2345 }
2346
2347 \cs_new_protected:Nn \_stex_modules_load_sig: {
2348 \stex_file_split_off_lang:NN \l__stex_modules_sigfile \g_stex_current_file
2349 \tl_set:Nc \l__stex_modules_sig {
2350 \exp_args:NNo \stex_document_uri_with_language:Nn
2351 \l_stex_current_document_uri \l_stex_key_sig_str
2352 }
2353 \stex_debug:nn{modules}{Loading~signature~\stex_use_document_uri:N \l__stex_modules_sig;~
2354 \exp_args:NNe \stex_file_in_smsmode:Nn \l__stex_modules_sig {
2355 \stex_file_use:N \l__stex_modules_sigfile . \l_stex_key_sig_str . tex
2356 }
2357 \stex_activate_module:N \l__stex_modules_uri
2358 }

```

(End of definition for _stex_modules_top_sig:n.)

_stex_modules_nested:n sets up a new nested module:

```

2359 \cs_new_protected:Nn \_stex_modules_nested:n {
2360 \stex_metagroup_new:
2361 \stex_debug:nn{module}{Nested~module~#1~in~\stex_use_module_uri:N \l_stex_current_module_uri}
2362 \tl_set:Nc \l__stex_modules_uri { \stex_new_module_uri:n{#1} }
2363 \stex_debug:nn{module}{URI:\stex_use_module_uri:N \l__stex_modules_uri}
2364 \exp_args:No \stex_if_module_exists:NTF \l__stex_modules_uri {
2365 \stex_debug:nn{modules}{(already exists)}
2366 }{
2367 \tl_gclear:c{ \_stex_module_code_macro:N \l__stex_modules_uri }
2368 \prop_gclear:c{ \_stex_morphisms_macro:N \l__stex_modules_uri }
2369 \prop_gclear:c{ \_stex_symbol_macro:N \l__stex_modules_uri }
2370 \prop_gclear:c{ \_stex_notations_macro:N \l__stex_modules_uri }
2371 }

```



```

2372 \tl_set_eq:NN \l_stex_current_module_uri \l__stex_modules_uri
2373 \seq_put_left:No \l_stex_all_modules_seq \l__stex_modules_uri
2374 }

```

(End of definition for __stex_modules_nested:n.)

\stex_close_module:

```

2375 \cs_new_protected:Nn \stex_close_module: {
2376   \bool_if:NT \c_stex_persist_write_mode_bool \__stex_modules_persist_module:
2377   \stex_debug:nn{module}{
2378     Closing-module~\stex_use_module_uri:N \l_stex_current_module_uri^^J
2379     Code:~\cs_meaning:c{ \stex_module_code_macro:N \l_stex_current_module_uri }^^J
2380     Imports:~ \exp_after:wN \prop_to_keyval:N \cs:w \stex_morphisms_macro:N \l_stex_current_module_uri
2381     Declarations:~ \exp_after:wN \prop_to_keyval:N \cs:w \stex_symbol_macro:N \l_stex_current_module_uri
2382     Notations:~ \exp_after:wN \prop_to_keyval:N \cs:w \stex_notations_macro:N \l_stex_current_module_uri
2383   }
2384 }

```

Persist and restore modules in the .sms-file:

```

2385 \cs_new_protected:Nn \__stex_modules_persist_module: {
2386   \stex_persist:e {
2387     \__stex_modules_restore_module:nnnn {\l_stex_current_module_uri}{
2388       \exp_after:wN \prop_to_keyval:N \cs:w
2389         \stex_morphisms_macro:N \l_stex_current_module_uri
2390       \cs_end:
2391     }{
2392       \exp_after:wN \prop_to_keyval:N \cs:w
2393         \stex_symbol_macro:N \l_stex_current_module_uri
2394       \cs_end:
2395     }{
2396       \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
2397       \exp_after:wN \exp_after:wN \exp_after:wN
2398       { \cs:w \stex_module_code_macro:N \l_stex_current_module_uri \cs_end: }
2399     }{}
2400     \prop_map_function:cN{\stex_notations_macro:N \l_stex_current_module_uri}
2401       \__stex_modules_persist_not_i:nn
2402     \exp_not:N \STEXRestoreNotEnd {}
2403   }
2404 }
2405
2406 \cs_new:Nn \__stex_modules_persist_not_i:nn {
2407   \exp_not:n{#2}
2408 }
2409
2410
2411 \cs_new:Nn \__stex_modules_stringify_uri:nnn {
2412   {\tl_to_str:n{#1}}
2413   {\tl_to_str:n{#2}}
2414   {\tl_to_str:n{#3}}
2415 }
2416 \cs_new:Nn \__stex_modules_stringify_uri:nnnn {
2417   {\tl_to_str:n{#1}}
2418   {\tl_to_str:n{#2}}
2419   {\tl_to_str:n{#3}}
2420   {\tl_to_str:n{#4}}

```

```

2421 }
2422
2423 \cs_new_protected:Nn \__stex_modules_restore_module:nnnn {
2424   \tl_set:Nc \l__stex_modules_uri { \__stex_modules_stringify_uri:nnn #1 }
2425   \stex_debug:nn{persist}{restoring~\stex_use_module_uri:N \l__stex_modules_uri}
2426   \prop_gset_from_keyval:cn{ \stex_morphisms_macro:N \l__stex_modules_uri }{#2}
2427   \prop_gset_from_keyval:cn{ \stex_symbol_macro:N \l__stex_modules_uri }{#3}
2428   \prop_map_inline:cn{ \stex_symbol_macro:N \l__stex_modules_uri }{
2429     \stex_ref_new_symbol:n{#1?##1}
2430   }
2431   \def \l_tmpa_tl { #4 }
2432   \exp_args:Nno \tl_gset:cn{ \stex_module_code_macro:N \l__stex_modules_uri } \l_tmpa_tl
2433   \prop_gc_clear:c{ \stex_notations_macro:N \l__stex_modules_uri }
2434   \group_begin:
2435     \catcode' =8\relax
2436     \catcode' =12\relax
2437     \__stex_modules_restore_notcs:n
2438   }
2439
2440 \quark_new:N \STEXRestoreNotsEnd
2441
2442 \cs_new_protected:Nn \__stex_modules_restore_notcs:n {
2443   \__stex_modules_restore_notcs_i:n
2444 }
2445
2446 \cs_new_protected:Nn \__stex_modules_restore_notcs_i:n {
2447   \tl_if_eq:nnTF{#1}{\STEXRestoreNotsEnd}{
2448     \group_end:
2449   }{
2450     \__stex_modules_restore_notcs_ii:nnnn {#1}
2451   }
2452 }
2453
2454 \cs_new_protected:Nn \__stex_modules_restore_notcs_ii:nnnn {
2455   \tl_set:Nc \l__stex_modules_tl {{{#4}{#5}}}
2456   % eliminate ## => #
2457   \exp_after:wN \def \exp_after:wN \l__stex_modules_tl \exp_after:wN {\l__stex_modules_tl}
2458   \exp_args:NNe\use:nn\prop_gput:cnn{
2459     { \stex_notations_macro:N \l__stex_modules_uri }
2460     { \stex_use_symbol_uri:n{#1}\tl_to_str:n{?#2}}{
2461       { \__stex_modules_stringify_uri:nnnn #1}{\tl_to_str:n{#2}}{#3}
2462     }
2463   }
2464 }
2465 \__stex_modules_restore_notcs_i:n
2466 }

```

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

sfunction

\stex_every_module:n

```

2467 \tl_new:N \g_stex_every_module_tl
2468 \cs_new_protected:Nn \stex_every_module:n {
2469   \tl_gput_right:Nn \g_stex_every_module_tl { #1 }
2470 }

```

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

sfunction

`\stex_do_up_to_module:n`

`\stex_do_up_to_module:e`

```
2471 \cs_new_protected:Nn \stex_do_up_to_module:n {
2472   \stex_metagroup_do_in:n {#1}
2473 }
2474 \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 134.)

sfunction

`\stex_execute_in_module:n`

`\stex_execute_in_module:e`

```
2475 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:TF {
2476   \tl_gput_right:cn { \_stex_module_code_macro:N \l_stex_current_module_uri } { #1 }
2477   \stex_debug:nn{modules}{extending~activation~for~\stex_use_module_uri:N \l_stex_current_m
2478   \stex_do_up_to_module:n { #1 }
2479 }{ #1 }}
2480 \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 134.)

sfunction

`\stex_if_in_module_p:`

`\stex_if_in_module:TF`

```
2481 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
2482   \tl_if_exist:NTF \l_stex_current_module_uri
2483   \prg_return_true: \prg_return_false:
2484 }
```

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

sfunction

`\stex_if_module_exists_p:N`

`\stex_if_module_exists:NTF`

`\stex_if_module_exists_p:n`

`\stex_if_module_exists:nTF`

```
2485 \prg_new_conditional:Nnn \stex_if_module_exists:N {p, T, F, TF} {
2486   \tl_if_exist:cTF { \_stex_module_code_macro:N #1 }
2487   \prg_return_true: \prg_return_false:
2488 }
2489 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
2490   \tl_if_exist:cTF { \_stex_module_code_macro:n {#1} }
2491   \prg_return_true: \prg_return_false:
2492 }
```

(End of definition for `\stex_if_module_exists:NTF` and `\stex_if_module_exists:nTF`. These functions are documented on page 135.)

sfunction

`\stex_activate_module:N`

`\stex_activate_module:n`

```
2493 \cs_new_protected:Nn \stex_activate_module:N {
2494   \exp_args:No \stex_activate_module:n #1
2495 }
2496
2497 \cs_new_protected:Nn \stex_activate_module:n {
2498   \seq_if_in:NnF \l_stex_all_modules_seq {#1} {
2499     \stex_debug:nn{modules}{Activating~module~\stex_use_module_uri:n{#1}}
```

```

2500 \seq_put_right:Nn \l_stex_all_modules_seq {#1}
2501 \stex_pseudogroup:nn{
2502   \tl_set:Nn \l_stex_current_module_uri {#1}
2503   \tl_if_exist:cF { \_stex_module_code_macro:N \l_stex_current_module_uri }{
2504     \msg_error:nnx{stex}{error/unknownmodule}{\stex_use_module_uri:n {#1} }
2505   }
2506   \_stex_activate_symbols:
2507   \_stex_activate_notations:
2508   \stex_debug:nn{modules}{Executing:~\expandafter\meaning\csname\_stex_module_code_macro:N \l_stex_current_module_uri }
2509   \use:c{ \_stex_module_code_macro:N \l_stex_current_module_uri }
2510 }{
2511   \stex_pseudogroup_restore:N \l_stex_current_module_uri
2512 }
2513 \stex_debug:nn{modules}{Activated~module~\stex_use_module_uri:n {#1} }
2514 }
2515 }

```

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

sfunction

`\setmetatheory`

```

2516 \NewDocumentCommand \setmetatheory {0{} m}{
2517   \stex_debug:nn{metatheory}{Setting~metatheory~[#1]#2}
2518   \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
2519   \stex_debug:nn{metatheory}{Here:\stex_use_module_uri:N \l_stex_import_uri
2520 }
2521   \tl_set_eq:NN \l_stex_metatheory_uri \l_stex_import_uri
2522   \begingroup
2523     \stex_require_module:N \l_stex_metatheory_uri
2524   \endgroup
2525   \stex_smsmode_do:
2526 }

```

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

sfunction

`\STEXexport`

```

2527 \cs_new_protected:Npn \STEXexport {
2528   \ExplSyntaxOn
2529   \__stex_modules_export:n
2530 }
2531 \cs_new_protected:Nn \__stex_modules_export:n {
2532   \stex_ignore_spaces_and_pars:#1\ExplSyntaxOff
2533   \tl_gput_right:cn { \_stex_module_code_macro:N \l_stex_current_module_uri } {
2534     \stex_ignore_spaces_and_pars: #1
2535   }
2536   \stex_smsmode_do:
2537 }
2538
2539 \stex_deactivate_macro:Nn \STEXexport {module~environments}
2540 \stex_every_module:n {\stex_reactivate_macro:N \STEXexport}

```

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

sfunction

```

\stex_structural_feature_module:nn
\stex_structural_feature_module_end:
2541 \cs_new_protected:Nn \stex_structural_feature_module:nn {
2542   \stex_if_do_html:TF {
2543     \exp_args:Nne \begin{stex_annotate_env} {
2544       data-ftml-feature-#2={#1}
2545       \str_if_empty:NF \l_stex_macroname_str {,
2546         data-ftml-macroname={\l_stex_macroname_str}
2547     }
2548   }
2549   \stex_annotate_invisible:n{ }
2550 } \group_begin:
2551 \stex_module_setup:n {#1}
2552 }
2553
2554 \cs_new_protected:Nn \stex_structural_feature_module_end: {
2555   \tl_gset_eq:NN \g_stex_last_feature_str \l_stex_current_module_str
2556   \stex_close_module:
2557   \stex_if_do_html:TF{
2558     \end{stex_annotate_env}
2559 } \group_end:
2560 }

(End of definition for \stex_structural_feature_module:nn and \stex_structural_feature_module_end:
. These functions are documented on page 135.)
sfunction

```

31.1 SMS Mode

```

2561 <@@=stex_smsmode>

\stex_sms_allow:N

2562 \tl_new:N \g__stex_smsmode_allowed_tl
2563
2564 \cs_new_protected:Nn \stex_sms_allow:N {
2565   \tl_gput_right:Nn \g__stex_smsmode_allowed_tl {#1}
2566 }

(End of definition for \stex_sms_allow:N. This function is documented on page 136.)
sfunction

\stex_sms_allow_escape:N

2567 \tl_new:N \g__stex_smsmode_allowed_escape_tl
2568
2569 \cs_new_protected:Nn \stex_sms_allow_escape:N {
2570   \tl_gput_right:Nn \g__stex_smsmode_allowed_escape_tl {#1}
2571 }

(End of definition for \stex_sms_allow_escape:N. This function is documented on page 136.)
sfunction

\stex_sms_allow_env:n

2572 \seq_new:N \g__stex_smsmode_allowedenvs_seq
2573
2574 \cs_new_protected:Nn \stex_sms_allow_env:n {

```

```

2575 \seq_gput_right:Ne \g__stex_smsmode_allowedenvs_seq {\tl_to_str:n{#1}}
2576 }

```

(End of definition for \stex_sms_allow_env:n. This function is documented on page 136.)

sfunction

Some initial allowed macros:

```

2577 \stex_sms_allow:N \makeatletter
2578 \stex_sms_allow:N \makeatother
2579 \stex_sms_allow:N \ExplSyntaxOn
2580 \stex_sms_allow:N \ExplSyntaxOff
2581 \stex_sms_allow:N \rustexBREAK
2582 \stex_sms_allow_env:n{smodule}
2583 \stex_sms_allow_escape:N \setmetatheory
2584 \stex_sms_allow_escape:N \STEXexport

```

\stex_sms_allow_import:Nn

```

2585 \tl_new:N \g__stex_smsmode_allowed_import_tl
2586 \tl_new:N \g__stex_smsmode_import_setup_tl
2587
2588 \cs_new_protected:Nn \stex_sms_allow_import:Nn {
2589   \tl_gput_right:Nn \g__stex_smsmode_allowed_import_tl {#1}
2590   \tl_gput_right:Nn \g__stex_smsmode_import_setup_tl {#2}
2591 }

```

(End of definition for \stex_sms_allow_import:Nn. This function is documented on page 136.)

sfunction

\stex_sms_allow_import_env:nn

```

2592 \seq_new:N \g__stex_smsmode_allowed_import_env_seq
2593
2594 \cs_new_protected:Nn \stex_sms_allow_import_env:nn {
2595   \seq_gput_right:Ne \g__stex_smsmode_allowed_import_env_seq {\tl_to_str:n{#1}}
2596   \tl_gput_right:Nn \g__stex_smsmode_import_setup_tl {#2}
2597 }
2598
2599 %\stex_sms_allow_import_env:nn{smodule}{}

```

(End of definition for \stex_sms_allow_import_env:nn. This function is documented on page 136.)

sfunction

\g_stex_sms_import_code

```

2600 \tl_new:N \g_stex_sms_import_code

```

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

\stex_if_smsmode_p:

\stex_if_smsmode:TF

```

2601 \bool_new:N \g__stex_smsmode_bool
2602
2603 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2604   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2605 }

```

(End of definition for \stex_if_smsmode:TF. This function is documented on page 136.)

sfunction

\stex_file_in_smsmode:Nn

```
2606
2607 \seq_new:N \l__stex_smsmode_cycles_seq
2608
2609 \cs_new_protected:Nn \stex_file_in_smsmode:Nn {
2610   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2611   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2612   \tl_clear:N \g_stex_sms_import_code
2613   \seq_if_in:NnF \l__stex_smsmode_cycles_seq {#2} {
2614     \group_begin:
2615       \seq_push:Nn \l__stex_smsmode_cycles_seq {#2}
2616       \let \l_stex_metatheory_uri \c_stex_default_metatheory
2617       \seq_clear:N \l_stex_all_modules_seq
2618       \stex_undefine:N \l_stex_current_module_uri
2619       \cs_set:Npn \stex_check_term:nn ##1 ##2 {}
2620       \stex_debug:nn{sms}{Loading~#2~in~\stex_use_document_uri:N #1}
2621       \stex_with_file_hooks:Nnn #1 {#2}{
2622         \__stex_smsmode_in_smsmode:n {
2623           \group_begin:
2624             \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_imports:N
2625             \g__stex_smsmode_import_setup_tl
2626             \__stex_smsmode_start_smsmode:n{#2}
2627           \group_end:
2628           %{
2629             \g_stex_sms_import_code
2630           %}
2631           %\group_begin:
2632             \let \__stex_smsmode_do_aux_curr:N \__stex_smsmode_do_aux_normal:N
2633             \__stex_smsmode_start_smsmode:n{#2}
2634           %\group_end:
2635         }
2636       }
2637     \group_end:
2638   }
2639 }
2640
2641 \cs_new_protected:Nn \__stex_smsmode_in_smsmode:n {
2642   \stex_suppress_html:n {
2643     \vbox_set:Nn \l_tmpa_box {
2644       \bool_set_true:N \g__stex_smsmode_bool
2645       #1
2646     }
2647   }
2648 }
2649
2650 \quark_new:N \q__stex_smsmode_break
2651
2652 \cs_new_protected:Nn \__stex_smsmode_start_smsmode:n {
2653   \everyeof{\q__stex_smsmode_break\exp_not:N}
2654   \let\stex_smsmode_do:\__stex_smsmode_smsmode_do:
2655   \exp_after:wN \exp_after:wN \exp_after:wN
2656   \stex_smsmode_do:
2657   \cs:w @ @ input\cs_end: "#1" \relax
2658 }
```

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

sfunction

`\stex_smsmode_do:`

```

2659 \let\stex_smsmode_do:\relax
2660
2661 \cs_new_protected:Nn \__stex_smsmode_smsmode_do: { \__stex_smsmode_do:w }
2662
2663 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2664   \exp_args:Ne \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2665     %\__stex_smsmode_check_cs:NNn \__stex_smsmode_do_aux:N \__stex_smsmode_do:w { #1 }
2666     \__stex_smsmode_check_cs:N {#1}
2667   }{
2668     \__stex_smsmode_do:w #1
2669   }
2670 }
2671
2672 \cs_new_protected:Nn \__stex_smsmode_check_cs:N {
2673   %\stex_debug:nn{temp}{Checking \exp_not:N#1 \relax}
2674   \exp_after:wN\if\exp_after:wN\relax\noexpand #1 ~
2675   \exp_after:wN \__stex_smsmode_do_aux:N \exp_after:wN#1\else
2676   \exp_after:wN \__stex_smsmode_do:w \fi
2677 }
2678
2679 %\cs_new_protected:Nn \__stex_smsmode_check_cs:NNn {
2680 %  \message{^^JHERE: \tl_to_str:n{#1~and~#2~and~#3}}
2681 %  \exp_after:wN\if\exp_after:wN\relax\exp_not:N#3
2682 %  \exp_after:wN#1\exp_after:wN#3\else
2683 %  \exp_after:wN#2\fi
2684 %}
2685
2686 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2687   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
2688     \__stex_smsmode_do_aux_curr:N #1
2689   }
2690 }
2691
2692 \cs_new_protected:Nn \__stex_smsmode_do_aux_normal:N {
2693   \tl_if_in:NnTF \g__stex_smsmode_allowed_tl {#1} {
2694     \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}}
2695     #1\__stex_smsmode_do:w
2696   }{
2697     \tl_if_in:NnTF \g__stex_smsmode_allowed_escape_tl {#1} {
2698       \stex_debug:nn{sms}{Executing~escaped~\tl_to_str:n{#1}}
2699       #1
2700     }{
2701       \cs_if_eq:NNTF \begin #1 {
2702         \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowedenvs_seq
2703       }{
2704         \cs_if_eq:NNTF \end #1 {
2705           \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowedenvs_seq
2706         }{
2707           \__stex_smsmode_do:w
2708         }

```



```

2709     }
2710   }
2711 }
2712 }
2713
2714 \cs_new_protected:Nn \__stex_smsmode_do_aux_imports:N {
2715   \tl_if_in:NnTF \g__stex_smsmode_allowed_import_tl {#1} {
2716     \stex_debug:nn{sms}{Executing~\tl_to_str:n{#1}~in~import}
2717     #1
2718   }{
2719     \cs_if_eq:NNTF \begin #1 {
2720       \__stex_smsmode_check_begin:Nn \g__stex_smsmode_allowed_import_env_seq
2721     }{
2722       \cs_if_eq:NNTF \end #1 {
2723         \__stex_smsmode_check_end:Nn \g__stex_smsmode_allowed_import_env_seq
2724       }{
2725         \__stex_smsmode_do:w
2726       }
2727     }
2728   }
2729 }
2730
2731 \cs_new_protected:Nn \__stex_smsmode_check_begin:Nn {
2732   \seq_if_in:NeTF #1 { \tl_to_str:n{#2} }{
2733     \stex_debug:nn{sms}{Environment~#2}
2734     \begin{#2}
2735   }{
2736     \__stex_smsmode_do:w
2737   }
2738 }
2739 \cs_new_protected:Nn \__stex_smsmode_check_end:Nn {
2740   \seq_if_in:NeTF #1 { \tl_to_str:n{#2} }{
2741     \stex_debug:nn{sms}{End~Environment~#2}
2742     \end{#2}\__stex_smsmode_do:w
2743   }{
2744     \__stex_smsmode_do:w
2745   }
2746 }

```

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

sfunction

31.2 Inheritance and Morphisms

```

2747 <@@=stex_import>

\stex_require_module:N
\stex_require_module_noerr:N
\stex_require_module_noerr:n
2748 \cs_new_protected:Nn \stex_require_module:N {
2749   \stex_if_module_exists:NF #1 {
2750     \__stex_import_load_module:NF #1 {
2751       \msg_error:nnx{stex}{error/unknownmodule}{\stex_use_module_uri:N #1}
2752     }
2753   }
2754   \stex_activate_module:N #1

```

```

2755 }
2756
2757 \cs_new_protected:Nn \stex_require_module_noerr:N {
2758   \stex_if_module_exists:NF #1 {
2759     \__stex_import_load_module:NF #1 {}
2760     \str_if_empty:NF \l__stex_import_file_str {
2761       \stex_activate_module:N #1
2762     }
2763   }
2764 }
2765
2766 \cs_new_protected:Nn \stex_require_module_noerr:n {
2767   \tl_set:Nn \l__stex_import_uri { #1 }
2768   \stex_require_module_noerr:N \l__stex_import_uri
2769 }
2770
2771 \cs_new_protected:Npn \__stex_import_load_module:NF #1 #2 {
2772   \tl_set_eq:NN \l__stex_import_uri #1
2773   \tl_set:Ne \l__stex_import_archive_str { \stex_module_uri_archive:N #1 }
2774   \tl_set:Ne \l__stex_import_path_str { \stex_module_uri_path:N #1 }
2775   \tl_set:Ne \l__stex_import_name_str { \stex_module_uri_name:N #1 }
2776   \str_if_eq:NNTF \l__stex_import_archive_str \c_stex_no_archive_str {
2777     \str_set_eq:NN \l__stex_import_pre_str \l__stex_import_path_str
2778     \__stex_import_load_check:n {#2}
2779   }{
2780     \exp_args:No \stex_in_archive:nn \l__stex_import_archive_str {
2781       \str_set:Ne \l__stex_import_pre_str {
2782         \exp_args:Ne \stex_source_path:n{ \l__stex_import_path_str }
2783       }
2784       \exp_args:No \__stex_import_load_check:n {#2}
2785     }
2786   }
2787 }
2788
2789 \cs_new_protected:Nn \__stex_import_load_check:n {
2790   \str_clear:N \l__stex_import_file_str
2791   \str_set_eq:NN \l__stex_import_language_str \l_stex_current_language_str
2792   \__stex_import_check_file:nnn{ / \l__stex_import_name_str .tex }{}{
2793     \__stex_import_check_file:nnn{/ \l__stex_import_name_str . \l_stex_current_language_str
2794     \__stex_import_check_file:nnn{/ \l__stex_import_name_str .en.tex}{
2795       \str_set:Nn \l__stex_import_language_str {en}
2796     }{
2797       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq / \l__stex_import_path_str
2798       \seq_pop_right:NN \l_tmpa_seq \l__stex_import_name_str
2799       \str_set:Ne \l__stex_import_path_str { \seq_use:Nn \l_tmpa_seq /}
2800       \__stex_import_check_file:nnn{.tex}{}{
2801         \__stex_import_check_file:nnn{. \l_stex_current_language_str .tex}{}{
2802           \__stex_import_check_file:nnn{.en.tex}{
2803             \str_set:Nn \l__stex_import_language_str {en}
2804           }{
2805             #1
2806           }
2807         }
2808       }

```

```

2809     }
2810   }
2811 }
2812 \str_if_empty:NF \l__stex_import_file_str {
2813   \stex_if_smsmode:TF{
2814     \exp_args:NNo \exp_args:Nne \str_if_eq:nnTF{\l__stex_import_file_str}{\stex_file_use:
2815       \stex_debug:nn{imports}{Skipping~current~file}
2816     }{
2817       \IfFileExists{ \l__stex_import_file_str }{
2818         \__stex_import_load_file:
2819       }{}
2820     }
2821   }{
2822     \IfFileExists{ \l__stex_import_file_str }{
2823       \__stex_import_load_file:
2824     }{}
2825   }
2826 }
2827 }
2828
2829 \cs_new_protected:Npn \__stex_import_check_file:nnn #1 #2 {
2830   \stex_debug:nn{imports}{Checking~ \l__stex_import_pre_str #1}
2831   \IfFileExists{ \l__stex_import_pre_str #1 }{
2832     \stex_debug:nn{imports}{Success}
2833     \str_set:Ne \l__stex_import_file_str { \l__stex_import_pre_str #1 }
2834     #2
2835   }
2836 }
2837
2838 \cs_new_protected:Nn \__stex_import_load_file: {
2839   \tl_set:Ne \l__stex_import_doc_uri {
2840     { \l__stex_import_archive_str }
2841     { \l__stex_import_path_str }
2842     { \l__stex_import_name_str }
2843     { \l__stex_import_language_str }
2844   }{}
2845 }
2846 \exp_args:NNo \stex_file_in_smsmode:Nn \l__stex_import_doc_uri \l__stex_import_file_str
2847 \stex_if_module_exists:NF \l__stex_import_uri {
2848   \msg_error:nnx{stex}{error/unknownmodule}{\stex_use_module_uri:N \l__stex_import_uri}
2849 }
2850 }

```

(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 137.)

sfunction

`\usemodule`

```

2851 \stex_new_stylable_cmd:nnnn {usemodule} { 0{} m } {
2852   \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
2853   \stex_require_module:N \l_stex_import_uri
2854   \stex_debug:nn{usemodule}{Done.}
2855   \stex_if_do_html:T {
2856     \hbox{\stex_annotate_invisible:nn

```

```

2857     {data-ftml-usemodule=\stex_use_module_uri:N \l_stex_import_uri } {}
2858   }
2859   \stex_if_smsmode:F{
2860     \group_begin:
2861     \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_import_uri }
2862     \str_set:Ne \thismodulename {\stex_module_uri_name:N \l_stex_import_uri}
2863     \tl_clear:N \thisstyle
2864     \stex_style_apply:
2865     \group_end:
2866   }
2867   }{}%\aftergroup\ignorespaces}

```

(End of definition for \usemodule. This function is documented on page 137.)

sfunction

\importmodule

```

2868 \stex_new_stylable_cmd:nnnn{importmodule} { 0{ } m } {
2869   \__stex_import_import_module:nn {#1}{#2}
2870   \stex_smsmode_do:
2871 }{\aftergroup\ignorespaces}
2872
2873 \stex_deactivate_macro:Nn \importmodule {module~environments}
2874 \stex_every_module:n {\stex_reactivate_macro:N \importmodule}
2875 \stex_sms_allow_escape:N \importmodule
2876
2877 \cs_new_protected:Nn \__stex_import_import_module:nn {
2878   \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
2879   \stex_require_module:N \l_stex_import_uri
2880   \stex_execute_in_module:e{
2881     \stex_activate_module:n { \l_stex_import_uri }
2882   }
2883   \stex_add_morphism:nonn
2884   {}{\l_stex_import_uri}{import}{}
2885   \stex_if_do_html:T {
2886     \stex_annotate_invisible:nn
2887     {data-ftml-import=\stex_use_module_uri:N \l_stex_import_uri } {}
2888   }
2889   \stex_if_smsmode:F{
2890     \group_begin:
2891     \tl_set:Nn \thisarchive {#1}
2892     \str_set:Ne \thismoduleuri {\stex_use_module_uri:N \l_stex_import_uri }
2893     \str_set:Ne \thismodulename {\stex_module_uri_name:N \l_stex_import_uri}
2894     \tl_clear:N \thisstyle
2895     \stex_style_apply:
2896     \group_end:
2897   }
2898 }

```

In sms mode, we change the definition:

```

2899 \cs_new_protected:Nn \__stex_import_import_module_presms:nn {
2900   \stex_uri_from_pair:Nnn \l_stex_import_uri { #1 }{ #2 }
2901   \bool_if:NF \l_stex_relative_import_bool {
2902     \tl_gput_right:Ne \g_stex_sms_import_code {
2903       \stex_require_module_noerr:n { \l_stex_import_uri }
2904     }

```

```

2905 }
2906 }
2907
2908 \stex_sms_allow_import:Nn \importmodule {
2909   \stex_reactivate_macro:N \importmodule
2910   \let \_stex_import_import_module:nn \_stex_import_import_module_presms:nn
2911 }

```

(End of definition for \importmodule. This function is documented on page 137.)

sfunction

\stex_add_morphism:nnnn

```

2912 \cs_new_protected:Nn \stex_add_morphism:nnnn {
2913   \exp_args:Nne \prop_gput:cnn{ \stex_morphisms_macro:N \l_stex_current_module_uri }{
2914     \tl_if_empty:nTF{#1}{[\stex_use_module_uri:n {#2}]}{#1}
2915     }{{#1}{#2}{#3}{#4}}
2916 }
2917 \cs_generate_variant:Nn \stex_add_morphism:nnnn {nonn,oooe}

```

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

sfunction

31.3 Theory Morphisms

2918 <@@=stex_morphisms>

\stex_structural_feature_morphism:nnnnn
 ructural_feature_morphism_with_macros:nnnnn
 \stex_structural_feature_morphism_end:

```

2919 \tl_new:N \l_stex_current_domain_uri
2920 \bool_new:N \l__stex_morphisms_with_macros_bool
2921
2922 \cs_new_protected:Npn \stex_structural_feature_morphism_with_macros:nnnnn {
2923   \bool_set_true:N \l__stex_morphisms_with_macros_bool
2924   \__stex_morphisms_morphism:nnnnn
2925 }
2926 \cs_new_protected:Npn \stex_structural_feature_morphism:nnnnn {
2927   \bool_set_false:N \l__stex_morphisms_with_macros_bool
2928   \__stex_morphisms_morphism:nnnnn
2929 }
2930
2931 \cs_new_protected:Nn \__stex_morphisms_morphism:nnnnn {
2932   \tl_clear:N \l_stex_current_domain_uri
2933   \stex_debug:nn{morphisms}{new~morphism:{#1}{#2}{#3}{#4}{#5}}
2934   \tl_if_empty:nTF{#3}{
2935     \stex_get_mathstructure:n{#4}
2936     \tl_set_eq:NN \l_stex_current_domain_uri \l_stex_get_structure_module_uri
2937   }
2938   \tl_if_empty:NT \l_stex_current_domain_uri {
2939     \stex_uri_from_pair:Nnn \l_stex_import_uri { #3 }{ #4 }
2940     \group_begin:
2941     \stex_require_module:N \l_stex_import_uri
2942     \exp_args:Nne \use:nn \group_end: {
2943       \tl_set:Nn \exp_not:N\l_stex_current_domain_uri {\l_stex_import_uri}
2944     }
2945   }

```

```

2946 \tl_if_empty:nT{#1}{
2947   \msg_error:nn{stex}{error/morphism-needs-name}
2948 }
2949 \str_set:Nn \l_tmpa_str {#1}
2950
2951 \stex_debug:nn{morphisms}{#2:~\l_tmpa_str;~for~\stex_use_module_uri:N \l_stex_current_dom
2952
2953 \str_set:Nn \l__stex_morphisms_feature_str {#2}
2954 \str_set_eq:NN \l_stex_feature_name_str \l_tmpa_str
2955
2956 \stex_if_do_html:TF {
2957   \begin{stex_annotate_env} {
2958     data-ftml-feature-#2={\l_tmpa_str},
2959     data-ftml-domain={\stex_use_module_uri:N \l_stex_current_domain_uri}
2960     #5
2961   }
2962   \stex_annotate_invisible:n{ }
2963 } \group_begin:
2964 \__stex_morphisms_setup:
2965 \__stex_morphisms_reactivate:
2966 \stex_metagroup_new:
2967 }
2968
2969 \cs_new_protected:Nn \__stex_morphisms_setup: {
2970   \seq_clear:N \l_stex_morphism_symbols_seq
2971   \seq_clear:N \l_stex_morphism_renames_seq
2972   \seq_clear:N \l_stex_morphism_morphisms_seq
2973   \seq_clear:N \l_stex_morphism_assigns_seq
2974   \__stex_morphisms_do_decls:
2975   \exp_args:No \__stex_morphisms_do:n \l_stex_current_domain_uri
2976 }
2977
2978 \cs_new_protected:Nn \__stex_morphisms_do_decls: {
2979   \exp_args:No \stex_iterate_symbols:nn \l_stex_current_domain_uri {
2980     \exp_args:NNe \seq_put_right:Nn \l_stex_morphism_symbols_seq {
2981       { \stex_new_symbol_uri:nn{##1}{##3} }
2982       { {##2}{##4}{##5}{##6}{##7}{##8}##9 }
2983     }
2984   }
2985 }
2986
2987
2988 \cs_new_protected:Nn \__stex_morphisms_do:n {
2989   %\seq_clear:N \l__stex_morphisms_dones_seq
2990   \prop_map_inline:cn {\_stex_morphisms_macro:n{#1}}{
2991     %\seq_if_in:NnF \l__stex_morphisms_dones_seq {##2}{
2992       % \seq_push:Nn \l__stex_morphisms_dones_seq {##2}
2993       \stex_debug:nn{morphisms}{Doing ~ \tl_to_str:n{##2}}
2994       \__stex_morphisms_do_morph:nnnn ##2
2995     % }
2996   }
2997 }
2998
2999 \cs_new_protected:Nn \__stex_morphisms_do_morph:nnnn {

```

```

3000     \tl_if_empty:nF{#3}{
3001         \seq_put_right:Nn \l_stex_morphism_morphisms_seq {{#1}{#2}{#3}}
3002     }
3003     \__stex_morphisms_do:n{#2}
3004 }
3005
3006
3007 \cs_new_protected:Nn \stex_structural_feature_morphism_end: {
3008     \tl_gset_eq:NN \l_stex_current_domain_uri \l_stex_current_domain_uri
3009     \seq_gset_eq:NN \l_stex_morphism_symbols_seq \l_stex_morphism_symbols_seq
3010     \seq_gset_eq:NN \l_stex_morphism_renames_seq \l_stex_morphism_renames_seq
3011     \seq_gset_eq:NN \l_stex_morphism_assigns_seq \l_stex_morphism_assigns_seq
3012     \seq_gset_eq:NN \l_stex_morphism_morphisms_seq \l_stex_morphism_morphisms_seq
3013     \bool_gset_eq:NN \l__stex_morphisms_with_macros_bool \l__stex_morphisms_with_macros_bool
3014     \stex_if_do_html:TF{
3015         \end{stex_annotate_env}
3016     }\group_end:
3017     \__stex_morphisms_do_elaboration:
3018 }
3019
3020 \cs_new_protected:Nn \__stex_morphisms_do_elaboration: {
3021     \stex_debug:nn{morphisms}{
3022         Elaborating:^^J\meaning \l_stex_morphism_symbols_seq
3023         ^^J^^J
3024         Renamings:^^J
3025         \meaning \l_stex_morphism_renames_seq
3026         ^^J^^J
3027         Assignments:^^J
3028         \meaning \l_stex_morphism_assigns_seq
3029     }
3030
3031     \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
3032         \__stex_morphisms_elab:nn ##1
3033     }
3034
3035     \stex_add_morphism:oooo
3036         \l_stex_feature_name_str
3037         \l_stex_current_domain_uri
3038         \l__stex_morphisms_feature_str
3039         {\seq_map_function:NN \l_stex_morphism_renames_seq \__stex_morphisms_rename:n}
3040 }
3041
3042 \cs_new:Nn \__stex_morphisms_rename:n{
3043     \__stex_morphisms_rename:nn #1
3044 }
3045
3046 \cs_new:Nn \__stex_morphisms_rename:nn{
3047     {#1}{\use_ii:nn#2}
3048 }
3049
3050 \cs_new_protected:Nn \__stex_morphisms_elab:nn {
3051     \tl_clear:N \l__stex_morphisms_tmp
3052     \seq_map_inline:Nn \l_stex_morphism_renames_seq {
3053         \__stex_morphisms_elab_check_rename:nnn{#1}##1

```

```

3054 }
3055 \tl_if_empty:NTF \l__stex_morphisms_tmp {
3056   \exp_args:Ne \__stex_morphisms_elab_i:nnn {\stex_symbol_uri_name:n {#1}}{#1}
3057 }{
3058   \stex_debug:nn{morphisms}{Generating~1~\l__stex_morphisms_tmp}
3059   \use_i:nn{ \exp_args:Nno \use:nn {\__stex_morphisms_add_symbol:nnnnnnnN {#1}} \l__stex
3060 }#2
3061
3062 \exp_args:Nno\stex_iterate_notations:nn\l_stex_current_domain_uri {
3063   \str_if_eq:nnT{#1}{##1}{
3064     \exp_args:Ne \stex_add_notation:nnnnn {
3065       \exp_args:Ne \stex_new_symbol_uri:n {\exp_after:wN \use_ii:nn \l__stex_morphisms_tm
3066 }{##2}{##3}{##4}{##5}
3067     }
3068   }
3069 }
3070
3071 \cs_new_protected:Nn \__stex_morphisms_elab_i:nnn {
3072   \tl_set:Ne \l__stex_morphisms_tmp {{}}{\l_stex_feature_name_str / #1}}
3073   \stex_debug:nn{morphisms}{Generating~2~\l_stex_feature_name_str / #1}
3074   \bool_if:NTF \l__stex_morphisms_with_macros_bool {
3075     \exp_args:Nnno \__stex_morphisms_add_symbol:nnnnnnnnN {#2} {#3}
3076   }{
3077     \exp_args:Nnno \__stex_morphisms_add_symbol:nnnnnnnnN {#2} {}
3078   }
3079   {\l_stex_feature_name_str / #1}
3080 }
3081
3082 \cs_new_protected:Npn \__stex_morphisms_add_symbol:nnnnnnnnN #1 {
3083   \tl_clear:N \l__stex_morphisms_tmp_b
3084   \seq_map_inline:Nn \l_stex_morphism_assigns_seq {
3085     \__stex_morphisms_elab_check_assign:nnnnn{#1}##1
3086   }
3087   \tl_if_empty:NTF \l__stex_morphisms_tmp_b
3088   \stex_add_symbol:nnnnnnnN
3089   \__stex_morphisms_add_symbol_i:nnnnnnnnN
3090 }
3091
3092 \cs_new_protected:Npn \__stex_morphisms_add_symbol_i:nnnnnnnnN #1 #2 #3 #4 #5 {
3093   \stex_add_symbol:nnnnnnnN {#1}{#2}{#3}{#4}{assigned}
3094 }
3095
3096 \cs_new_protected:Nn \__stex_morphisms_elab_check_assign:nnnnn {
3097   \tl_if_eq:nnT{#1}{#2}{#3}{#4}{#5}{
3098     \seq_map_break:n{
3099       \tl_set:Nn \l__stex_morphisms_tmp_b {assigned}
3100     }
3101   }
3102 }
3103
3104 \cs_new_protected:Nn \__stex_morphisms_elab_check_rename:nnn {
3105   \tl_if_eq:nnT{#1}{#2}{
3106     \seq_map_break:n{
3107       \tl_set:Nn \l__stex_morphisms_tmp {#3}

```



```

3108     }
3109   }
3110 }
3111
3112 \cs_new_protected:Nn \__stex_morphisms_do_for_list: {
3113   \seq_clear:N \l_stex_fors_seq
3114   \clist_map_inline:Nn \l_stex_key_for_clist {
3115     \exp_args:N\stex_get_in_morphism:n{\tl_to_str:n{##1}}
3116     \seq_put_right:No \l_stex_fors_seq
3117       {\l_stex_get_symbol_uri}
3118   }
3119 }
3120
3121
3122 \cs_new_protected:Nn \__stex_morphisms_definiens_impl:nn {
3123   \tl_if_empty:nF {#1} {
3124     \stex_get_in_morphism:n { #1 }
3125     \tl_set_eq:NN \l_stex_current_def_uri \l_stex_get_symbol_uri
3126   }
3127   \tl_if_empty:NT \l_stex_current_def_uri {
3128     \msg_error:nn{stex}{error/definiensfor}
3129   }
3130   \stex_debug:nn{definiens}{Checking-\stex_use_symbol_uri:N \l_stex_current_def_uri }
3131   \stex_if_do_html:TF{
3132     \exp_after:wN \stex_metagroup_do_in:n \exp_after:wN {
3133       \exp_after:wN \seq_put_right:Nn \exp_after:wN \l_stex_morphism_assigns_seq
3134       \exp_after:wN { \l_stex_current_def_uri }
3135     }
3136     \mode_leave_vertical: \stex_annotate:nn{data-ftml-assign={\stex_use_symbol_uri:N \l_st
3137       \stex_annotate_force_break:n{
3138         \stex_annotate:nn{data-ftml-definiens={}}{#2}
3139       }
3140     }
3141   }{
3142     \exp_args:No \__stex_morphisms_add_definiens:nn \l_stex_current_def_uri {#2}
3143     \stex_if_smsmode:F{#2}
3144   }
3145 }
3146
3147 \cs_new_protected:Nn \__stex_morphisms_add_definiens:nn {
3148   \tl_set:Nn \l_stex_get_symbol_uri {#1}
3149   \stex_debug:nn{morphisms}{Adding~definiens~\tl_to_str:n{#1~#2}}
3150   %\__stex_morphisms_set_definiens_macros: #1\__stex_morphisms_break:
3151   \stex_assign_do:n{#2}
3152 }
3153
3154 %\cs_new_protected:Npn \__stex_morphisms_set_definiens_macros: #1?#2?#3\__stex_morphisms_br
3155 % \str_set:Nn \l_stex_get_symbol_uri { #1?#2} % TODO
3156 % \str_set:Nn \l_stex_get_symbol_name_str {#3}
3157 % \exp_args:Nne\use:nn{\__stex_morphisms_set_definiens_macros_i:nnnnnnn}{
3158 %   \prop_item:Nn \l_stex_morphism_symbols_prop {[#1?#2]/[#3]}
3159 % }
3160 %}
3161

```

```

3162 %\cs_new_protected:Nn \__stex_morphisms_set_definiens_macros_i:nnnnnnn {
3163 % \tl_set:Nn \l_stex_get_symbol_def_tl{#4}
3164 %}
3165
3166
3167 \cs_new_protected:Nn \__stex_morphisms_rename_all: {
3168 % TODO
3169 }
3170
3171
3172 \cs_new_protected:Nn \stex_structural_feature_morphism_check_total: {
3173 \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
3174 \__stex_morphisms_total_check:nn ##1
3175 }
3176 }
3177
3178 \cs_new_protected:Nn \__stex_morphisms_total_check:nn {
3179 \__stex_morphisms_total_check:nnnnnnN {#1} #2
3180 }
3181
3182 \cs_new_protected:Nn \__stex_morphisms_total_check:nnnnnnN {
3183 \tl_if_empty:nT{#5}{
3184 \seq_if_in:NnF \l_stex_morphism_assigns_seq {#1} {
3185 \msg_error:nnxx{stex}{error/needsdefiniens}{\stex_use_symbol_uri:n {#1}}{total-morphi
3186 }
3187 }
3188 }
3189
3190 \cs_new:Npn \__stex_morphisms_split_qm:w #1 ? #2 ? #3 { #3 }
3191
3192
3193
3194 \cs_new_protected:Npn \__stex_morphisms_reactivate: {
3195 \stex_deactivate_macro:Nn \symdecl {module~environments}
3196 \stex_deactivate_macro:Nn \textsymdecl {module~environments}
3197 \stex_deactivate_macro:Nn \symdef {module~environments}
3198 \stex_deactivate_macro:Nn \notation {module~environments}
3199 \stex_deactivate_macro:Nn \importmodule {module~environments}
3200 \stex_deactivate_macro:Nn \requiremodule {module~environments}
3201 \stex_deactivate_macro:Nn \smodule {outside-of~morphisms}
3202 \stex_reactivate_macro:N \assign
3203 \stex_reactivate_macro:N \assignMorphism
3204 \stex_reactivate_macro:N \renamedec1
3205 \cs_set_eq:NN \_stex_do_for_list: \__stex_morphisms_do_for_list:
3206 \cs_set_eq:NN \_stex_add_definiens:nn \__stex_morphisms_add_definiens:nn
3207 \cs_set_eq:NN \_stex_definiens_impl:nn \__stex_morphisms_definiens_impl:nn
3208 }

```

(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 137.)

sfunction

\stex_iterate_morphisms:nn

```

3209 \cs_new_protected:Nn \stex_iterate_morphisms:nn {
3210   \seq_clear:N \l__stex_morphisms_mods_seq
3211   \bool_set_true:N \l__stex_morphisms_continue_bool
3212   \cs_set:Npn \__stex_morphisms_cs:nnnn ##1 ##2 ##3 ##4 ##5 {
3213     #2
3214     \bool_if:NT \l__stex_morphisms_continue_bool {
3215       \str_if_eq:nnTF{##1}{[#2]}{
3216         \tl_put_right:Nn \l__stex_morphisms_todo_tl {
3217           \__stex_morphisms_iterate:nn{##5}{##2}
3218         }
3219       }{
3220         \tl_put_right:Nn \l__stex_morphisms_todo_tl {
3221           \__stex_morphisms_iterate:nn{##5 / ##1}{##2}
3222         }
3223       }
3224     }
3225   }
3226   \cs_set:Npn \stex_iterate_break:n ##1 {
3227     \bool_set_false:N \l__stex_morphisms_continue_bool
3228     \prop_map_break:n{##1}
3229   }
3230   \__stex_morphisms_iterate:nn{}{#1}
3231 }
3232 \cs_new_protected:Nn \__stex_morphisms_iterate:nn {
3233   \tl_clear:N \l__stex_morphisms_todo_tl
3234   \seq_if_in:NnF \l__stex_morphisms_mods_seq {#1 #2}{
3235     \seq_put_right:Nn \l__stex_morphisms_mods_seq {#1 #2}
3236     \prop_map_inline:cn{\__stex_morphisms_macro:n{#2}}{
3237       \__stex_morphisms_cs:nnnn ##2 {#1}
3238       % TODO
3239       % ##1: name or [mpath]
3240       % ##2 = {####1}{####2}{####3}{####4}
3241       % ####1 = name
3242       % ####2 = mpath
3243       % ####3 = type
3244       % ####4 = {origname}{newname}*
3245     }
3246     \bool_if:NT \l__stex_morphisms_continue_bool \l__stex_morphisms_todo_tl
3247   }
3248 }

```

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

sfunction

`\stex_get_in_morphism:n`

```

3249 \cs_new_protected:Nn \stex_get_in_morphism:n {
3250   \stex_debug:nn{morphisms}{Getting-in-morphism:~#1}
3251   \tl_clear:N \l_stex_get_symbol_uri
3252   \str_set:Nn \l__stex_morphisms_get_str {#1}
3253   \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
3254     \__stex_morphisms_get_check:nn ##1
3255   }
3256   \tl_if_empty:NT \l_stex_get_symbol_uri {
3257     \seq_map_inline:Nn \l_stex_morphism_renames_seq {

```

```

3258     \__stex_morphisms_renamed_check:nn##1
3259   }
3260   \tl_if_empty:NT \l_stex_get_symbol_uri {
3261     \msg_error:nnxx{stex}{error/unknownsymbolin}{#1}{
3262       morphism~\l_stex_feature_name_str
3263     }
3264   }
3265 }
3266 }
3267
3268
3269 \cs_new_protected:Nn \__stex_morphisms_get_check:nn {
3270   \__stex_morphisms_check_name:nnnn #1 #2
3271 }
3272
3273 \cs_new_protected:Nn \__stex_morphisms_check_name:nnnn {
3274   % TODO module name, path, archive, macroname ?
3275   \exp_args:No \str_if_eq:nnTF \l__stex_morphisms_get_str {#4} {
3276     \tl_set:Nn \l_stex_get_symbol_uri {{#1}{#2}{#3}{#4}}
3277     \__stex_morphisms_set:nnnnnnN
3278   }{
3279     \__stex_morphisms_macro:nnnnnnN{{#1}{#2}{#3}{#4}}
3280   }
3281 }
3282
3283 \cs_new_protected:Nn \__stex_morphisms_set:nnnnnnN {
3284   \seq_map_break:n{
3285     \str_set:Nn \l_stex_get_symbol_macro_str{#1}
3286     \int_set:Nn \l_stex_get_symbol_arity_int {#2}
3287     \tl_set:Nn \l_stex_get_symbol_args_tl {#3}
3288     \tl_set:Nn \l_stex_get_symbol_def_tl {#4}
3289     \tl_set:Nn \l_stex_get_symbol_type_tl {#5}
3290     \tl_set:Nn \l_stex_get_symbol_return_tl {#6}
3291     \tl_set:Nn \l_stex_get_symbol_invoke_cs {#7}
3292   }
3293 }
3294
3295 \cs_new_protected:Npn \__stex_morphisms_macro:nnnnnnN #1 #2 {
3296   \exp_args:No \str_if_eq:nnTF \l__stex_morphisms_get_str {#2}{
3297     \tl_set:Nn \l_stex_get_symbol_uri{#1}
3298     \__stex_morphisms_set:nnnnnnN
3299   }\__stex_morphisms_gobble:nnnnnnN
3300   {#2}
3301 }
3302
3303 \cs_new_protected:Nn \__stex_morphisms_gobble:nnnnnnN {}
3304
3305 \cs_new_protected:Nn \__stex_morphisms_renamed_check:nn {
3306   \stex_debug:nn{here}{\tl_to_str:n{{#1}{#2}}:~\l_stex_morphisms_get_str}
3307   \__stex_morphisms_renamed_check:nnnnn #1 #2
3308 }
3309
3310 \cs_new_protected:Nn \__stex_morphisms_renamed_check:nnnnn {
3311   \exp_args:No \str_if_eq:nnTF \l__stex_morphisms_get_str{#5}{

```

```

3312 \seq_map_break:n{
3313   \str_set:Nn \l__stex_morphisms_get_str {#4}
3314   \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
3315     \__stex_morphisms_get_check:nn ##1
3316   }
3317 }
3318 }{
3319   \exp_args:No \str_if_eq:nnT \l__stex_morphisms_get_str{#6}{
3320     \seq_map_break:n{
3321       \str_set:Nn \l__stex_morphisms_get_str {#4}
3322       \seq_map_inline:Nn \l_stex_morphism_symbols_seq {
3323         \__stex_morphisms_get_check:nn ##1
3324       }
3325     }
3326   }
3327 }
3328 }

```

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

sfunction

copymodule (env.)

```

3329 \stex_new_stylable_env:nnnnnnn {copymodule}{m 0{} m}{
3330   \__stex_morphisms_begin_copy:Nnnn\stex_structural_feature_morphism:nnnnn {#1}{#2}{#3}
3331 }{
3332   \stex_if_smsmode:F {
3333     \stex_style_apply:
3334   }
3335   \stex_structural_feature_morphism_end:
3336 }{}{}{}
3337
3338 \stex_new_stylable_env:nnnnnnn {copymodule*}{m 0{} m}{
3339   \cs_set:Npn \stex_style_apply: {
3340     \stex_apply_patch_begin:n{copymodule}
3341   }
3342   \__stex_morphisms_begin_copy:Nnnn\stex_structural_feature_morphism_with_macros:nnnnn {#1}
3343 }{
3344   \cs_set:Npn \stex_style_apply: {
3345     \stex_apply_patch_end:n{copymodule}
3346   }
3347   \stex_if_smsmode:F {
3348     \stex_style_apply:
3349   }
3350   \stex_structural_feature_morphism_end:
3351 }{}{}{}
3352
3353 \cs_new_protected:Nn \__stex_morphisms_begin_copy:Nnnn {
3354   #1{#2}{morphism}{#3}{#4}{,data-ftml-total=false}
3355   \stex_if_smsmode:F {
3356     \tl_set:Nn \thiscopyname { #2 }
3357     \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
3358     \stex_style_apply:
3359   }
3360   \stex_smsmode_do:

```

```

3361 }
3362
3363
3364 \stex_deactivate_macro:Nn \copymodule {module~environments}
3365 \stex_every_module:n {
3366   \stex_reactivate_macro:N \copymodule
3367 }
3368 \stex_sms_allow_env:n{copymodule}
3369
3370 \exp_after:wN \stex_deactivate_macro:Nn \csname copymodule*\endcsname {module~environments}
3371 \stex_every_module:n {
3372   \exp_after:wN\stex_reactivate_macro:N \csname copymodule*\endcsname
3373 }
3374 \stex_sms_allow_env:n{copymodule*}
3375
3376 \stex_new_stylable_cmd:nnnn{copymod}{s m O{} m m}{
3377   \IfBooleanTF#1\stex_structural_feature_morphism_with_macros:nnnnn
3378   \stex_structural_feature_morphism:nnnnn{#2}{morphism}{#3}{#4}{,data-ftml-total={false}}
3379   \clist_map_function:nN{#5}\__stex_morphisms_parse_assign:n
3380   \stex_if_smsmode:F {
3381     \tl_set:Nn \thiscopyname { #2 }
3382     \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
3383     \stex_style_apply:
3384   }
3385   \stex_structural_feature_morphism_end:
3386   \stex_smsmode_do:
3387 }{}
3388
3389 \stex_deactivate_macro:Nn \copymod {module~environments}
3390 \stex_every_module:n {
3391   \stex_reactivate_macro:N \copymod
3392 }
3393 \stex_sms_allow_escape:N\copymod

```

interpretmodule (env.)

```

3394 \stex_new_stylable_env:nnnnnnn {interpretmodule}{m O{} m}{
3395   \__stex_morphisms_begin_interpret:Nnnn\stex_structural_feature_morphism:nnnnn {#1}{#2}{#3}
3396 }{}
3397   \stex_structural_feature_morphism_check_total:
3398   \stex_if_smsmode:F {
3399     \stex_style_apply:
3400   }
3401   \stex_structural_feature_morphism_end:
3402 }{}{}{}
3403
3404 \stex_new_stylable_env:nnnnnnn {interpretmodule*}{m O{} m}{
3405   \cs_set:Npn \stex_style_apply: {
3406     \_stex_apply_patch_begin:n{interpretmodule}
3407   }
3408   \__stex_morphisms_begin_interpret:Nnnn\stex_structural_feature_morphism_with_macros:nnnnn
3409 }{}
3410   \cs_set:Npn \stex_style_apply: {
3411     \_stex_apply_patch_end:n{interpretmodule}
3412   }

```

```

3413 \stex_structural_feature_morphism_check_total:
3414 \stex_if_smsmode:F {
3415   \stex_style_apply:
3416 }
3417 \stex_structural_feature_morphism_end:
3418 }{}{}{}
3419
3420 \cs_new_protected:Nn \__stex_morphisms_begin_interpret:Nnnn {
3421   #1{#2}{morphism}{#3}{#4}{,data-ftml-total=true}
3422   \stex_if_smsmode:F {
3423     \tl_set:Nn \thiscopyname { #2 }
3424     \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
3425     \stex_style_apply:
3426   }
3427   \stex_smsmode_do:
3428 }
3429
3430 \stex_deactivate_macro:Nn \interpretmodule {module~environments}
3431 \stex_every_module:n {
3432   \stex_reactivate_macro:N \interpretmodule
3433 }
3434 \stex_sms_allow_env:n{interpretmodule}
3435
3436 \exp_after:wN \stex_deactivate_macro:Nn \csname interpretmodule*\endcsname {module~environments}
3437 \stex_every_module:n {
3438   \exp_after:wN \stex_reactivate_macro:N \csname interpretmodule*\endcsname
3439 }
3440 \stex_sms_allow_env:n{interpretmodule*}
3441
3442 \stex_new_stylable_cmd:nnnn{interpretmod}{s m 0{} m m}{
3443   \IfBooleanTF#1\stex_structural_feature_morphism_with_macros:nnnnn
3444   \stex_structural_feature_morphism:nnnnn{#2}{morphism}{#3}{#4}{,data-ftml-total=true}
3445   \clist_map_function:nN{#5}\__stex_morphisms_parse_assign:n
3446   \stex_if_smsmode:F {
3447     \tl_set:Nn \thiscopyname { #2 }
3448     \tl_set:Nn \thismoduleuri {\stex_use_module_uri:N \l_stex_current_domain_uri}
3449     \stex_style_apply:
3450   }
3451   \stex_structural_feature_morphism_check_total:
3452   \stex_structural_feature_morphism_end:
3453   \stex_smsmode_do:
3454 }{}
3455
3456 \stex_deactivate_macro:Nn \interpretmod {module~environments}
3457 \stex_every_module:n {
3458   \stex_reactivate_macro:N \interpretmod
3459 }
3460 \stex_sms_allow_escape:N\interpretmod

```

\assign

```

3461 \stex_new_stylable_cmd:nnnn {assign} { m m }{
3462   \stex_get_in_morphism:n{#1}
3463   \stex_if_do_html:TF{
3464     \stex_annotate_invisible:n{\hbox{$\_stex_assign_do:n{#2}$}}

```

```

3465   }{
3466     \_stex_assign_do:n{#2}
3467   }
3468   \stex_smsmode_do:
3469 }{}
3470 \stex_sms_allow_escape:N\assign
3471 \stex_deactivate_macro:Nn \assign {morphism~environments}
3472
3473 \cs_new_protected:Nn \_stex_assign_do:n{
3474   \stex_debug:nn{assign}{Assigning~\stex_use_symbol_uri:N \l_stex_get_symbol_uri~to~\tl_to_
3475   \stex_check_term:nn{assignment}{#1}
3476   \stex_if_do_html:T{
3477     \stex_annotate_invisible:nn{data-ftml-assign={\stex_use_symbol_uri:N \l_stex_get_symbol
3478     \_stex_annotate_force_break:n{
3479       \mode_if_math:TF{\stex_annotate:nn{data-ftml-definiens={}}{#1}}{\hbox{$\stex_annota
3480     }
3481   }
3482 }
3483 \exp_after:wN \stex_metagroup_do_in:n \exp_after:wN {
3484   \exp_after:wN \seq_put_right:Nn \exp_after:wN \l_stex_morphism_assigns_seq
3485   \exp_after:wN { \l_stex_get_symbol_uri }
3486 }
3487 }
3488
3489 \cs_new_protected:Nn \_stex_morphisms_parse_assign:n {
3490   \str_clear:N \l__stex_morphisms_name_str
3491   \str_clear:N \l__stex_morphisms_newname_str
3492   \tl_clear:N \l__stex_morphisms_ass_tl
3493   \stex_debug:nn{morphisms}{Parsing~#1}
3494   \exp_args:NNe \seq_set_split:Nnn \l__stex_morphisms_seq {\tl_to_str:n{@}} {#1}
3495   \int_compare:nNnTF {\seq_count:N \l__stex_morphisms_seq} = 1 {
3496     \stex_debug:nn{morphisms}{No~@}
3497     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_next_tl
3498   }{
3499     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3500     \stex_debug:nn{morphisms}{Name:~\l__stex_morphisms_name_str}
3501     \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3502     \tl_set:Ne \l__stex_morphisms_next_tl {\seq_use:Nn \l__stex_morphisms_seq @}
3503   }
3504   \exp_args:NNNo \seq_set_split:Nnn \l__stex_morphisms_seq = \l__stex_morphisms_next_tl
3505   \str_if_empty:NTF \l__stex_morphisms_name_str {
3506     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_name_str
3507     \exp_args:NNo \str_set:Nn \l__stex_morphisms_name_str \l__stex_morphisms_name_str
3508     \tl_set:Ne \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3509   }{
3510     \seq_pop_left:NN \l__stex_morphisms_seq \l__stex_morphisms_newname_str
3511     \exp_args:NNo \str_set:Nn \l__stex_morphisms_newname_str \l__stex_morphisms_newname_str
3512     \tl_set:Ne \l__stex_morphisms_ass_tl {\seq_use:Nn \l__stex_morphisms_seq =}
3513   }
3514   \_stex_morphisms_do_parsed_assign:
3515 }
3516
3517 \cs_new_protected:Nn \_stex_morphisms_do_parsed_assign: {
3518   \exp_args:No \stex_get_in_morphism:n \l__stex_morphisms_name_str

```



```

3519 \str_if_empty:NF \l__stex_morphisms_newname_str {
3520   \stex_debug:nn{morphisms}{renaming~\l__stex_morphisms_name_str;~to~\l__stex_morphisms_n
3521   \exp_after:wN \__stex_morphisms_do_parsed_newname: \l__stex_morphisms_newname_str \__st
3522 }
3523 \tl_if_empty:NF \l__stex_morphisms_ass_tl {
3524   \stex_debug:nn{morphisms}{assigning~\l__stex_morphisms_name_str;~to~\exp_args:No \tl_to
3525   \exp_args:No \_stex_assign_do:n \l__stex_morphisms_ass_tl
3526 }
3527 }
3528
3529 \cs_new_protected:Nn \__stex_morphisms_do_parsed_newname: {
3530   \peek_charcode:NTF [ {
3531     \__stex_morphisms_do_parsed_newname:w
3532   }{
3533     \__stex_morphisms_do_parsed_newname:w []
3534   }
3535 }
3536
3537 \cs_new_protected:Npn \__stex_morphisms_do_parsed_newname:w [#1] #2 \__stex_morphisms_end:
3538   \_stex_renamedec1_do:nn{#1}{#2}
3539 }

```

(End of definition for \assign. This function is documented on page 137.)

sfunction

\renamedec1

```

3540 \stex_new_stylable_cmd:nnnn {renamedec1} { m 0{} m }{
3541   \stex_get_in_morphism:n{#1}
3542   \_stex_renamedec1_do:nn{#2}{#3}
3543   \stex_smsmode_do:
3544 }{}
3545 \stex_sms_allow_escape:N\renamedec1
3546 \stex_deactivate_macro:Nn \renamedec1 {morphism~environments}
3547
3548 \cs_new_protected:Nn \_stex_renamedec1_do:nn {
3549   \stex_debug:nn{renamedec1}{Renaming~\stex_use_symbol_uri:N \l_stex_get_symbol_uri~to~[#1]
3550   \stex_if_do_html:T{
3551     \exp_args:Ne \stex_annotate_invisible:nn{
3552       data-ftml-rename={\stex_use_symbol_uri:N \l_stex_get_symbol_uri},
3553       data-ftml-macroname={#2}
3554     }
3555     \str_if_empty:NF{#1}{ ,data-ftml-to={#1} }
3556   }{}
3557 }
3558 \stex_metagroup_do_in:e {
3559   \seq_push:Nn \exp_not:N \l_stex_morphism_renames_seq
3560   { {\l_stex_get_symbol_uri}{#2}{
3561     \tl_if_empty:nTF{#1}{
3562       \l_stex_feature_name_str/\stex_symbol_uri_name:N \l_stex_get_symbol_uri
3563     }{#1}
3564   } }
3565 }

```

(End of definition for \renamedec1. This function is documented on page 137.)

sfunction

\assignMorphism

```

3566 \stex_new_stytable_cmd:nnnn {assignMorphism} { m m }{
3567   \str_clear:N \l__stex_morphisms_morphism_dom_str
3568   \exp_args:No \stex_iterate_morphisms:nn\l_stex_current_domain_uri{
3569     \stex_debug:nn{assignMorphism}{
3570       Checking:~#1~vs:^^J##1^^J##2^^J##3^^J##4
3571     }
3572     \str_if_eq:nnTF{#1}{##1}{
3573       \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3574     }{
3575       \stex_str_if_ends_with:nnT{##2}{#1}{
3576         \__stex_morphisms_do_morph_assign:nnn{##1}{##2}{##4}
3577       }
3578     }
3579   }
3580   \str_if_empty:NT \l__stex_morphisms_morphism_dom_str {
3581     \msg_error:nnn{stex}{error/nomorphism}{#1}
3582   }
3583   \bool_set_false:N \l_tmpa_bool
3584   \stex_iterate_morphisms:nn \l_stex_current_module_str {
3585     \stex_debug:nn{assignMorphism}{
3586       Checking:~#2~vs:^^J##1^^J##2^^J##3^^J##4
3587     }
3588     \str_if_eq:nnTF{#2}{##1}{
3589       \stex_debug:nn{assignMorphism}{match!}
3590       \stex_iterate_break:n{
3591         \stex_annotate_invisible:nn{
3592           data-ftml-assignmorphismfrom={\l__stex_morphisms_morphism_dom_str}
3593           data-ftml-assignmorphismto={\l_stex_current_module_str?##1}
3594         }{}
3595         \bool_set_true:N \l_tmpa_bool
3596       }
3597     }{
3598       \stex_str_if_ends_with:nnT{##2}{#2}{
3599         \stex_debug:nn{assignMorphism}{match!}
3600         \stex_iterate_break:n{
3601           \stex_annotate_invisible:nn{
3602             data-ftml-assignmorphismfrom={\l__stex_morphisms_morphism_dom_str},
3603             data-ftml-assignmorphismto={\l_stex_current_module_str?##1}
3604           }{}
3605           \bool_set_true:N \l_tmpa_bool
3606         }
3607       }
3608     }
3609   }
3610   \bool_if:NF \l_tmpa_bool {
3611     \msg_error:nnn{stex}{error/nomorphism}{#2}
3612   }
3613 }{}
3614 \stex_sms_allow_escape:N \assignMorphism
3615 \stex_deactivate_macro:Nn \assignMorphism {morphism-environments}
3616
3617 \cs_new_protected:Nn \__stex_morphisms_do_morph_assign:nnn {
3618   \stex_iterate_break:n{

```

```

3619     \str_set:Ne \l__stex_morphisms_morphism_dom_str { \l_stex_current_domain_str ? #1 }
3620     \stex_debug:nn{assignMorphism}{match!}
3621     \stex_iterate_symbols:nn{#2}{
3622         \stex_debug:nn{assignMorphism}{removing-##1?##3}
3623         % TODO: non-trivial assignments
3624         \prop_remove:Nn \l_stex_morphism_symbols_prop {
3625             [##1]/[##3]
3626         }
3627     }
3628 }
3629 }

```

(End of definition for \assignMorphism. This function is documented on page 138.)

sfunction

Chapter 32

Symbols

3630 <@@=stex_syms>

32.1 Declarations

Keys used in various symbol declarations:

```
3631 \stex_keys_define:nnnn{symargs}{
3632   \str_clear:N \l_stex_key_args_str
3633   \str_clear:N \l_stex_key_role_str
3634   \str_clear:N \l_stex_key_reorder_str
3635   \str_clear:N \l_stex_key_assoc_str
3636 }{
3637   args      .str_set:N = \l_stex_key_args_str ,
3638   reorder   .str_set:N = \l_stex_key_reorder_str ,
3639   assoc     .choices:nn = {bin,binl,binr,pre,conj,pwconj}
3640             {\str_set:Ne \l_stex_key_assoc_str \l_keys_choice_tl},
3641   role      .str_set:N = \l_stex_key_role_str
3642 }{}
3643
3644 \stex_keys_define:nnnn{decl}{
3645   \str_clear:N \l_stex_key_name_str
3646   \str_clear:N \l_stex_key_args_str
3647   \tl_clear:N \l_stex_key_type_tl
3648   \tl_clear:N \l_stex_key_def_tl
3649   \tl_clear:N \l_stex_key_return_tl
3650   \str_clear:N \l_stex_key_wikidata_str
3651   \clist_clear:N \l_stex_key_argtypes_clist
3652 }{
3653   name      .str_set:N = \l_stex_key_name_str ,
3654   return    .tl_set:N = \l_stex_key_return_tl ,
3655   argtypes  .clist_set:N = \l_stex_key_argtypes_clist ,
3656   wikidata  .str_set:N = \l_stex_key_wikidata_str ,
3657   type      .tl_set:N = \l_stex_key_type_tl ,
3658   def       .tl_set:N = \l_stex_key_def_tl ,
3659   align     .code:n = {},
3660   gf        .code:n = {}
3661 }{style,deprecate,symargs}
```

\symdecl

```

3662 \str_new:N \l_stex_macroname_str
3663
3664 \stex_new_stylable_cmd:nnnn {symdecl} { s m 0{}} {
3665   \stex_keys_set:nn{decl}{#3}
3666   \IfBooleanTF #1 {
3667     \str_clear:N \l_stex_macroname_str
3668   }{
3669     \str_set:Ne \l_stex_macroname_str { #2 }
3670   }
3671   \__stex_syms_top:n{#2}
3672   \stex_if_smsmode:F \__stex_syms_style:
3673   \stex_smsmode_do:
3674 }{}
3675
3676 \stex_deactivate_macro:Nn \symdecl {module~environments}
3677 \stex_every_module:n {\stex_reactivate_macro:N \symdecl}
3678 \stex_sms_allow_escape:N \symdecl
3679
3680 \cs_new_protected:Nn \__stex_syms_style: {
3681   \group_begin:
3682     \tl_set:Ne \thisdecluri {\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
3683     \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3684     \tl_set_eq:NN \thistype \l_stex_key_type_tl
3685     \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
3686     \tl_set_eq:NN \thisargs \l_stex_key_args_str
3687     \tl_clear:N \thisstyle
3688     \stex_style_apply:
3689   \group_end:
3690 }

```

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

sfunction

`\symdef`

```

3691 \stex_new_stylable_cmd:nnnn {symdef} { m 0{ } m } {
3692   \stex_keys_set:nn{symdef}{#2}
3693   \str_set:Ne \l_stex_macroname_str { #1 }
3694   \__stex_syms_top:n{#1}
3695   \stex_debug:nn{symdef}{Doing~\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
3696   \str_set_eq:NN \l_stex_get_symbol_uri \l_stex_current_symbol_uri
3697   \stex_notation_parse:n{#3}
3698   \stex_if_check_terms:T{ \_stex_notation_check: }
3699   \_stex_notation_add:
3700   \stex_if_do_html:T{
3701     \_stex_notation_do_html:n{\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
3702   }
3703   \stex_if_smsmode:F \__stex_syms_def_style:
3704   \stex_smsmode_do:
3705 }{}
3706
3707 \stex_deactivate_macro:Nn \symdef {module~environments}
3708 \stex_every_module:n {\stex_reactivate_macro:N \symdef}
3709 \stex_sms_allow_escape:N \symdef
3710

```

```

3711 \cs_new_protected:Nn \__stex_syms_def_style: {
3712   \group_begin:
3713   \tl_set:Nx \thisdecluri {\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
3714   \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3715   \tl_set_eq:NN \thistype \l_stex_key_type_tl
3716   \tl_set_eq:NN \thisdefiniens \l_stex_key_def_tl
3717   \tl_set_eq:NN \thisargs \l_stex_key_args_str
3718   \tl_clear:N \thisstyle
3719   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
3720   \def\thisnotation{
3721     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}}{
3722     \stex_notation_make_args:
3723   }
3724 }
3725 \stex_style_apply:
3726 \group_end:
3727 }

```

(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 139.)

sffunction

`\textsymdecl`

```

3728 \stex_keys_define:nnnn{textsymdecl}{
3729   \str_clear:N \l_stex_key_name_str
3730   \tl_clear:N \l_stex_key_type_tl
3731   \tl_clear:N \l_stex_key_def_tl
3732 }{
3733   name      .str_set:N = \l_stex_key_name_str ,
3734   type      .tl_set:N  = \l_stex_key_type_tl ,
3735   def       .tl_set:N  = \l_stex_key_def_tl,
3736   gf        .code:n    = {}
3737 }{style,deprecate}
3738
3739 \stex_new_stylable_cmd:nnnn {textsymdecl} {m O{} m} {
3740   \stex_keys_set:nn{symdef}{-}
3741   \stex_keys_set:nn{textsymdecl}{#2}
3742   \str_set:Ne \l_stex_macroname_str { #1 }
3743   \str_if_empty:NT \l_stex_key_name_str {
3744     \str_set:Nn \l_stex_key_name_str {#1}
3745   }%{
3746   % \str_set:Ne \l_stex_key_name_str {\l_stex_key_name_str-sym}
3747   %}
3748   \str_set:Nn \l_stex_key_role_str {textsymdecl}
3749   \tl_set:Ne \l_stex_current_symbol_uri {
3750     \exp_args:No \stex_new_symbol_uri:n \l_stex_key_name_str
3751   }
3752   \stex_symdecl_do:
3753   \stex_check_terms:
3754   \exp_args:Nne \use:nn {\stex_add_symbol:nnnnnnN}{
3755     {\l_stex_macroname_str}
3756     {\l_stex_key_name_str}
3757     {0}{-}

```

```

3758     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
3759     {}% type
3760     {\use:c{#1name_nospace}}}% return
3761     \stex_invoke_text_symbol:
3762 }
3763 \exp_args:Ne \stex_ref_new_symbol:n
3764 { \stex_use_symbol_uri:N \l_stex_current_symbol_uri }
3765 \stex_if_do_html:T {
3766     \stex_symdecl_html:
3767 }
3768
3769 \int_set:Nn \l_stex_get_symbol_arity_int 0
3770 \tl_clear:N \l_stex_key_op_tl
3771 \str_clear:N \l_stex_key_intent_str
3772 \str_clear:N \l_stex_key_prec_str
3773 \tl_set_eq:NN \l_stex_get_symbol_uri \l_stex_current_symbol_uri
3774 \stex_notation_parse:n{\hbox{#3}}
3775 \stex_notation_add:
3776 \stex_if_do_html:T {
3777     \def\comp{\_comp}
3778     \stex_notation_do_html:n{ \stex_use_symbol_uri:N \l_stex_current_symbol_uri }
3779 }
3780 \stex_execute_in_module:e{
3781     \__stex_syms_set_textsymdecl_macro:nnn{#1}{ \stex_use_symbol_uri:N \l_stex_current_symbol_uri }
3782     \exp_not:n{#3}}
3783 }
3784
3785 \stex_if_smsmode:F{
3786     \group_begin:
3787     \tl_set:Ne \thisdecluri { \stex_use_symbol_uri:N \l_stex_current_symbol_uri }
3788     \tl_set_eq:NN \thisdeclname \l_stex_key_name_str
3789     \tl_clear:N \thisstyle
3790     \stex_style_apply:
3791     \group_end:
3792 }
3793 \stex_smsmode_do:
3794 }{}
3795
3796 \stex_deactivate_macro:Nn \textsymdecl {module~environments}
3797 \stex_every_module:n { \stex_reactivate_macro:N \textsymdecl }
3798 \stex_sms_allow_escape:N \textsymdecl
3799
3800 \cs_new_protected:Nn \__stex_syms_set_textsymdecl_macro:nnn {
3801     \cs_set_protected:cpn{#1name_nospace}{#3}
3802     \cs_set_protected:cpn{#1name}{
3803         \mode_if_vertical:T{\hbox_unpack:N\c_empty_box}
3804         \mode_if_math:T{\hbox{\let\xspace\relax #3}
3805         \mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
3806     }
3807 }

```

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

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.

```

3808 \cs_new_protected:Nn \__stex_syms_top:n {
3809   \str_if_empty:NT \l_stex_key_name_str {
3810     \str_set:Ne \l_stex_key_name_str { #1 }
3811   }
3812   \tl_set:Ne \l_stex_current_symbol_uri {
3813     \exp_args:No \stex_new_symbol_uri:n \l_stex_key_name_str
3814   }
3815
3816   \stex_symdecl_do:
3817   \_stex_check_terms:
3818   \__stex_syms_add_decl:
3819   \stex_if_do_html:T \_stex_symdecl_html:
3820 }

```

(End of definition for __stex_syms_top:n.)

Adds the symbol to the current module:

```

3821 \cs_new_protected:Nn \__stex_syms_add_decl: {
3822   \exp_args:Nne \use:nn {\stex_add_symbol:nnnnnnN}{
3823     {\l_stex_macroname_str}
3824     {\l_stex_key_name_str}
3825     {\int_use:N \l_stex_get_symbol_arity_int}
3826     {\l_stex_get_symbol_args_tl}
3827     {\tl_if_empty:NF \l_stex_key_def_tl{DEFED} }
3828     {}
3829     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
3830     \stex_invoke_symbol:
3831   }
3832   \exp_args:Ne \stex_ref_new_symbol:n
3833     {\stex_use_symbol_uri:N \l_stex_current_symbol_uri}
3834 }

```

\stex_symdecl_do:

```

3835 \cs_new_protected:Nn \stex_symdecl_do: {
3836   \_stex_do_deprecation:n \l_stex_key_name_str
3837   \__stex_syms_parse_arity:
3838   \__stex_syms_do_args:
3839 }
3840
3841 \cs_new:Nn \_stex_return_args:nn {
3842   {\svar{ARGUMENT_#1}\_stex_eat_exclamation_point:}
3843 }
3844
3845 \cs_new_protected:Nn \__stex_syms_do_args: {
3846   \tl_clear:N \l_stex_get_symbol_args_tl
3847   \int_step_inline:nn \l_stex_get_symbol_arity_int {
3848     \tl_put_right:Nn \l_stex_get_symbol_args_tl {##1}
3849     \tl_put_right:Ne \l_stex_get_symbol_args_tl {
3850       \str_item:Nn \l_stex_key_args_str {##1}
3851     }
3852   }
3853 }

```


Constructs the arity string of the symbol:

```

3854 \int_new:N \l_stex_assoc_args_count
3855
3856 \cs_new_protected:Nn \__stex_syms_parse_arity: {
3857   \int_zero:N \l_stex_get_symbol_arity_int
3858   \int_zero:N \l_stex_assoc_args_count
3859   \str_map_inline:Nn \l_stex_key_args_str {
3860     \str_case:nnF ##1 {
3861       0 { \str_map_break: }
3862       1 { \str_map_break:n{
3863         \int_set:Nn \l_stex_get_symbol_arity_int {1}
3864         \str_set:Nn \l_stex_key_args_str {i}
3865       } }
3866       2 { \str_map_break:n{
3867         \int_set:Nn \l_stex_get_symbol_arity_int {2}
3868         \str_set:Nn \l_stex_key_args_str {ii}
3869       } }
3870       3 { \str_map_break:n{
3871         \int_set:Nn \l_stex_get_symbol_arity_int {3}
3872         \str_set:Nn \l_stex_key_args_str {iii}
3873       } }
3874       4 { \str_map_break:n{
3875         \int_set:Nn \l_stex_get_symbol_arity_int {4}
3876         \str_set:Nn \l_stex_key_args_str {iiii}
3877       } }
3878       5 { \str_map_break:n{
3879         \int_set:Nn \l_stex_get_symbol_arity_int {5}
3880         \str_set:Nn \l_stex_key_args_str {iiiii}
3881       } }
3882       6 { \str_map_break:n{
3883         \int_set:Nn \l_stex_get_symbol_arity_int {6}
3884         \str_set:Nn \l_stex_key_args_str {iiiiii}
3885       } }
3886       7 { \str_map_break:n{
3887         \int_set:Nn \l_stex_get_symbol_arity_int {7}
3888         \str_set:Nn \l_stex_key_args_str {iiiiiii}
3889       } }
3890       8 { \str_map_break:n{
3891         \int_set:Nn \l_stex_get_symbol_arity_int {8}
3892         \str_set:Nn \l_stex_key_args_str {iiiiiiii}
3893       } }
3894       9 { \str_map_break:n{
3895         \int_set:Nn \l_stex_get_symbol_arity_int {9}
3896         \str_set:Nn \l_stex_key_args_str {iiiiiii}
3897       } }
3898       i {\int_incr:N \l_stex_get_symbol_arity_int}
3899       b {\int_incr:N \l_stex_get_symbol_arity_int}
3900       a {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3901       B {\int_incr:N \l_stex_get_symbol_arity_int \int_incr:N \l_stex_assoc_args_count}
3902     }{
3903       \msg_error:nnxx{stex}{error/wrongargs}{-}{##1}
3904     }
3905   }
3906 }
```

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

sfunction

`_stex_symdecl_html`:

```

3907 \cs_new_protected:Nn \_stex_symdecl_html: {
3908   \stex_annotate_invisible:n{
3909     \hbox{\exp_args:Ne \stex_annotate:nn {
3910       data-ftml-symdecl = { \l_stex_key_name_str},
3911       data-ftml-args = {\l_stex_key_args_str}
3912       \str_if_empty:NF \l_stex_macroname_str {,
3913         data-ftml-macroname={\l_stex_macroname_str}
3914       }
3915       \str_if_empty:NF \l_stex_key_wikidata_str {,
3916         data-ftml-wikidata={\l_stex_key_wikidata_str}
3917       }
3918       \str_if_empty:NF \l_stex_key_assoc_str {,
3919         data-ftml-assoctype={\l_stex_key_assoc_str}
3920       }
3921       \str_if_empty:NF \l_stex_key_reorder_str {,
3922         data-ftml-reorderargs={\l_stex_key_reorder_str}
3923       }
3924       \str_if_empty:NF \l_stex_key_role_str {,
3925         data-ftml-role={\l_stex_key_role_str}
3926       }
3927     }\stex_annotate_force_break:n{
3928       \bool_set_true:N \stex_in_invisible_html_bool
3929       \tl_if_empty:NF \l_stex_key_type_tl {
3930         $\stex_annotate:nn{data-ftml-type={}}{\l_stex_key_type_tl}$
3931       }
3932       \tl_if_empty:NF \l_stex_key_def_tl {
3933         $\stex_annotate:nn{data-ftml-definiens={}}{\l_stex_key_def_tl}$
3934       }
3935       \tl_if_empty:NF \l_stex_key_return_tl{
3936         \exp_args:Nno \use:n{
3937           \cs_generate_from_arg_count:NNnn \l__stex_syms_cs
3938           \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
3939           \tl_set:Ne \l__stex_syms_args_tl {\_stex_map_args:N \_stex_return_args:nn}
3940           $\stex_annotate:nn{data-ftml-returntype={}}{
3941             \exp_after:wN \l__stex_syms_cs \l__stex_syms_args_tl!
3942           }$
3943         }
3944       \clist_if_empty:NF \l_stex_key_argtypes_clist {
3945         \stex_annotate:nn{data-ftml-argtypes={}}{\_stex_annotate_force_break:n{
3946           \clist_map_inline:Nn \l_stex_key_argtypes_clist {
3947             $\stex_annotate:nn{data-ftml-type={}}{##1}$
3948           }
3949         }}
3950       }
3951     }}
3952   }}
3953 }
```

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

sfunction

\stex_add_symbol:nnnnnnN

```

3954 \cs_new_protected:Nn \stex_add_symbol:nnnnnnN {
3955   \stex_debug:nn{declaration}{New~declaration:~\stex_use_module_uri:N \l_stex_current_modul
3956   Macro:#1^^JArity:#3~(#4)^^J
3957   Def:~\tl_to_str:n{#5}^^J
3958   Type:~\tl_to_str:n{#6}^^J
3959   Returns:~\tl_to_str:n{#7}^^J
3960   Invokes:~\tl_to_str:n{#8}
3961 }
3962 \prop_gput:cnn{ \_stex_symbol_macro:N \l_stex_current_module_uri }
3963 {#2}{{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}}
3964 \tl_if_empty:nF{#1}{
3965   \stex_do_up_to_module:n {
3966     \__stex_syms_activate_i:nnnnnnnn{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}
3967   }
3968 }
3969 }

```

Generate semantic macros etc.:

```

3970 \cs_new_protected:Nn \_stex_activate_symbols: {
3971   \stex_debug:nn{activating}{All~symbols:~\cs_meaning:c{ \_stex_symbol_macro:N \l_stex_curr
3972   \prop_map_inline:cn{ \_stex_symbol_macro:N \l_stex_current_module_uri }{
3973     \__stex_syms_maybe_activate:nnnnnnnn ##2
3974   }
3975 }
3976
3977 \cs_new_protected:Nn \__stex_syms_maybe_activate:nnnnnnnn {
3978   \str_if_empty:nF{#1}{
3979     \__stex_syms_activate_i:nnnnnnnn {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}
3980   }
3981 }
3982
3983 \cs_new_protected:Nn \__stex_syms_activate_i:nnnnnnnn {
3984   \stex_debug:nn{activating}{macro~#1:\stex_use_module_uri:N \l_stex_current_module_uri ? #
3985   \tl_to_str:n{{#3}{#4}{#5}{#6}{#7}{#8}}
3986 }
3987 \cs_set:cpe{#1} {
3988   \_stex_invoke_symbol:nnnnnnN
3989   {\exp_args:No \stex_new_symbol_uri:nn {\l_stex_current_module_uri} {#2} }
3990   \exp_not:n{{#3}{#4}{#5}{#6}{#7}{#8}}
3991 }
3992 }

```

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

sfunction

\stex_has_definiens_p:N

\stex_has_definiens:N \overline{TF}

\stex_has_definiens_p:n

\stex_has_definiens:n \overline{TF}

```

3993 \prg_new_conditional:Nnn \stex_has_definiens:N { p, T, F, TF } {
3994   \exp_args:No \stex_has_definiens:nTF #1 \prg_return_true: \prg_return_false:
3995 }
3996 \prg_new_conditional:Nnn \stex_has_definiens:n { p, T, F, TF } {
3997   \exp_args:Nne \use:nn{\__stex_syms_has_definiens:nnnnnnnn}{\exp_args:Nne \prop_item:cn{
3998     \exp_args:Ne \_stex_symbol_macro:n {\stex_symbol_uri_module:n{#1}}
3999   }{

```

```

4000     \stex_symbol_uri_name:n {#1}
4001   }}
4002 }
4003
4004 \cs_new:Nn \__stex_syms_has_definiens:nnnnnnnN {
4005   \tl_if_empty:nTF{#5}\prg_return_false:\prg_return_true:
4006 }

```

(End of definition for `\stex_has_definiens:NTF` and `\stex_has_definiens:nTF`. These functions are documented on page 139.)

sfunction

32.2 Retrieval

```

\stex_iterate_symbols:n
  \stex_iterate_break:
  \stex_iterate_break:n
4007 \cs_new_protected:Nn \stex_iterate_symbols:n {
4008   \stex_pseudogroup_with:nn{\__stex_syms_sym_cs:nnnnnnnnN\stex_iterate_break:\stex_iterate_
4009     \cs_set:Npn \__stex_syms_sym_cs:nnnnnnnnN
4010     ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
4011     \cs_set:Npn \stex_iterate_break: {
4012       \prop_map_break:n{\seq_map_break:}
4013     }
4014     \cs_set:Npn \stex_iterate_break:n ##1 {
4015       \prop_map_break:n{\seq_map_break:n{##1}}
4016     }
4017     \seq_map_inline:Nn \l_stex_all_modules_seq {
4018       \prop_map_inline:cn{\_stex_symbol_macro:n {##1}}{
4019         \__stex_syms_sym_cs:nnnnnnnnN {##1} #####2
4020       }
4021     }
4022   }
4023 }

```

(End of definition for `\stex_iterate_symbols:n`, `\stex_iterate_break:`, and `\stex_iterate_break:n`. These functions are documented on page 140.)

sfunction

```

\stex_iterate_symbols:nn
4024 \cs_new_protected:Nn \stex_iterate_symbols:nn {
4025   \seq_clear:N \l__stex_syms_mods_seq
4026   \stex_pseudogroup_with:nn{\__stex_syms_sym_cs:nnnnnnnnN}{
4027     \cs_set:Npn \__stex_syms_sym_cs:nnnnnnnnN
4028     ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #2 }
4029     \clist_map_function:nN {#1} \__stex_syms_it_decl_i:n
4030   }
4031 }
4032
4033 \cs_new_protected:Nn \__stex_syms_it_decl_i:n {
4034   \seq_if_in:NnF \l__stex_syms_mods_seq {#1} {
4035     \seq_put_left:Nn \l__stex_syms_mods_seq {#1}
4036     \prop_map_inline:cn{\_stex_morphisms_macro:n {#1}}{
4037       \__stex_syms_it_decl_check:nnnn ##2
4038     }

```

```

4039     \prop_map_inline:cn{\_stex_symbol_macro:n {#1} }{
4040       \__stex_syms_sym_cs:nnnnnnnnN {#1} ##2
4041     }
4042   }
4043 }
4044
4045 \cs_new_protected:Nn \__stex_syms_it_decl_check:nnnn {
4046   \tl_if_empty:nT{#1}{
4047     \__stex_syms_it_decl_i:n {#2}
4048   }
4049 }

```

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

sfunction

\stex_get_symbol:n

\stex_get_symbol:nF

```

4050 \int_new:N \l_stex_get_symbol_arity_int
4051
4052 \cs_new_protected:Nn \stex_get_symbol:n {
4053   \stex_get_symbol:nF{ #1 }{
4054     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4055   }
4056 }
4057
4058 \cs_new_protected:Npn \stex_get_symbol:nF #1 #2 {
4059   \tl_clear:N \l_stex_get_symbol_uri
4060   \cs_if_exist:cTF { #1 }{
4061     \cs_set_eq:Nc \l__stex_syms_cs { #1 }
4062     % command name
4063     \exp_args:Ne \tl_if_empty:nTF { \cs_argument_spec:N \l__stex_syms_cs }{
4064       % ...that takes no arguments
4065       \exp_args:Ne \cs_if_eq:NNTF {\tl_head:N \l__stex_syms_cs}
4066         \stex_invoke_symbol:nnnnnnN
4067         \__stex_syms_get_symbol_from_cs:
4068         {\__stex_syms_get_symbol_from_string:n { #1 }}
4069     }{
4070       \__stex_syms_get_symbol_from_string:n { #1 }
4071     }
4072   }{
4073     \__stex_syms_get_symbol_from_string:n { #1 }
4074   }
4075   \tl_if_empty:NT \l_stex_get_symbol_uri { #2 }
4076 }
4077
4078 \cs_new_protected:Nn \__stex_syms_get_symbol_from_cs: {
4079   \stex_pseudogroup_with:nn{\_stex_invoke_symbol:nnnnnnN}{
4080     \cs_set:Npn \_stex_invoke_symbol:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 {
4081       \tl_set:Nn \l_stex_get_symbol_uri { ##1 }
4082       \int_set:Nn \l_stex_get_symbol_arity_int {##2}
4083       \tl_set:Nn \l_stex_get_symbol_args_tl {##3}
4084       \tl_set:Nn \l_stex_get_symbol_def_tl {##4}
4085       \tl_set:Nn \l_stex_get_symbol_type_tl {##5}
4086       \tl_set:Nn \l_stex_get_symbol_return_tl {##6}
4087       \tl_set:Nn \l_stex_get_symbol_invoke_cs {##7}

```

```

4088     }
4089     \l__stex_syms_cs
4090 }
4091 }
4092
4093 \cs_new_protected:Nn \__stex_syms_get_symbol_from_string:n {
4094     \stex_debug:nn{symbols}{Getting~from~string~#1...}
4095     \seq_set_split:Nnn \l__stex_syms_seq ? {#1}
4096     \seq_pop_right:NN \l__stex_syms_seq \l__stex_syms_name
4097     \seq_if_empty:NTF \l__stex_syms_seq {
4098         \exp_args:No \__stex_syms_get_from_one_string:n {#1}
4099     }{
4100         \exp_args:NNe \exp_args:Nno \__stex_syms_get_symbol_from_modules:nn {
4101             \seq_use:Nn \l__stex_syms_seq ?
4102         } \l__stex_syms_name
4103     }
4104 }
4105
4106 \cs_new_protected:Nn \__stex_syms_sym_from_str_i:nnnn {
4107     \bool_lazy_any:nTF{
4108         {\str_if_eq_p:nn{#2}{#3}}
4109         {\str_if_eq_p:nn{#2}{#4}}
4110         {\stex_str_if_ends_with_p:nn{#4}{/#2}}
4111     }{
4112         \__stex_syms_sym_i_finish:nnnnnnN{#1}{#4}
4113     }{
4114         \__stex_syms_sym_i_gobble:nnnnnn
4115     }
4116 }
4117 \cs_new_protected:Nn \__stex_syms_sym_i_gobble:nnnnnn {}
4118
4119 \cs_new_protected:Nn \__stex_syms_sym_i_finish:nnnnnnN {
4120     \prop_map_break:n{\seq_map_break:n{
4121         \tl_set:Ne \l_stex_get_symbol_uri { \stex_new_symbol_uri:nn {#1} {#2}}
4122         \int_set:Nn \l_stex_get_symbol_arity_int {#3}
4123         \tl_set:Nn \l_stex_get_symbol_args_tl {#4}
4124         \tl_set:Nn \l_stex_get_symbol_def_tl {#5}
4125         \tl_set:Nn \l_stex_get_symbol_type_tl {#6}
4126         \tl_set:Nn \l_stex_get_symbol_return_tl {#7}
4127         \tl_set:Nn \l_stex_get_symbol_invoke_cs {#8}
4128     }}
4129 }
4130
4131 \cs_new_protected:Nn \__stex_syms_get_symbol_from_modules:nn {
4132     %\seq_set_split:Nnn \l_tmpa_seq ? {#1}
4133     %\seq_pop_right:NN \l_tmpa_seq \l__stex_syms_name_str
4134     %\str_set:Ne \l__stex_syms_path_str {\seq_use:Nn \l_tmpa_seq ?}
4135     \stex_debug:nn{symbols}{Getting~#2~in~#1...}
4136     \seq_map_inline:Nn \l_stex_all_modules_seq {
4137         %\stex_debug:nn{symbols}{comparing~\stex_module_uri_as_qm:n{##1}~and~#1}
4138         \exp_args:Ne \stex_str_if_ends_with:nnT{\stex_module_uri_as_qm:n{##1}}{#1}{
4139             \prop_map_inline:cn{\_stex_symbol_macro:n {##1} }{
4140                 \__stex_syms_sym_from_str_i:nnnn{##1}{#2} ####2
4141             }

```

```

4142     }
4143
4144     %\str_if_empty:NTF \l__stex_syms_path_str {
4145     % \exp_args:NNe \exp_args:Nno \stex_str_if_ends_with:nnT {\stex_module_uri_name:n{##1}}
4146     %}{
4147     % \exp_args:NNNe \exp_args:No \str_if_eq:nnT\l__stex_syms_name_str{
4148     % \stex_module_uri_name:n {##1}
4149     % }
4150     %}{
4151     % \str_if_empty:NTF \l__stex_syms_path_str {
4152     % \prop_map_inline:cn{\stex_symbol_macro:n {##1} }{
4153     % \__stex_syms_sym_from_str_i:nnnn{##1}{#2} ####2
4154     % }
4155     % }{
4156     % \stex_debug:nn{symbols}{Comparing~\l__stex_syms_path_str;~with~\tl_to_str:n{##1}}
4157     % \exp_args:NNe \exp_args:Nno \stex_str_if_ends_with:nnT{\stex_module_uri_path:n {##1}}
4158     % \prop_map_inline:cn{\stex_symbol_macro:n {##1} }{
4159     % \__stex_syms_sym_from_str_i:nnnn{##1}{#2} ####2
4160     % }
4161     % }
4162     % }
4163     %}
4164 }
4165 }
4166
4167 \prg_new_conditional:Nnn \__stex_syms_uri_match:n {T,F,TF} {
4168 \str_if_eq:nnT
4169 }
4170
4171
4172 \cs_new_protected:Nn \__stex_syms_get_from_one_string:n {
4173 \stex_debug:nn{symbols}{Getting~#1~anywhere...}
4174 \stex_iterate_symbols:n{
4175 %\stex_debug:nn{symbols}{>#1==##2~|~#1==##3<...}
4176 \bool_lazy_any:nnT{
4177 {\str_if_eq_p:nn{##1}{##2}}
4178 {\str_if_eq_p:nn{##1}{##3}}
4179 {\stex_str_if_ends_with_p:nn{##3}{/#1}}
4180 }{
4181 \stex_iterate_break:n{
4182 \tl_set:Nn \l_stex_get_symbol_uri { \stex_new_symbol_uri:nn {##1} {##3}}
4183 \int_set:Nn \l_stex_get_symbol_arity_int {##4}
4184 \tl_set:Nn \l_stex_get_symbol_args_tl {##5}
4185 \tl_set:Nn \l_stex_get_symbol_def_tl {##6}
4186 \tl_set:Nn \l_stex_get_symbol_type_tl {##7}
4187 \tl_set:Nn \l_stex_get_symbol_return_tl {##8}
4188 \tl_set:Nn \l_stex_get_symbol_invoke_cs {##9}
4189 }
4190 }
4191 }
4192 }

```

(End of definition for `\stex_get_symbol:n` and `\stex_get_symbol:nF`. These functions are documented on page 140.)

sfunction

32.3 Variables

```

4193 <@@=stex_vars>

\l_stex_variables_prop

4194 \prop_new:N \l_stex_variables_prop

(End of definition for \l_stex_variables_prop. This variable is documented on page 140.)

\vardef

4195 \bool_new:N \l__stex_vars_bind_bool
4196 \cs_new_protected:Nn \_stex_variable:nnnnnnnN {}
4197
4198 \stex_new_stylable_cmd:nnnn {vardef} { m 0{} m} {
4199   \stex_keys_set:nn{vardef}{#2}
4200   \str_set:Ne \l_stex_macroname_str { #1 }
4201   \str_if_empty:NT \l_stex_key_name_str {
4202     \str_set:Ne \l_stex_key_name_str { #1 }
4203   }
4204
4205   \stex_symdecl_do:
4206   \stex_if_check_terms:T \_stex_check_terms:
4207   \__stex_vars_add:
4208   \__stex_vars_macro:
4209   \stex_if_do_html:T \__stex_vars_html:
4210
4211   \int_set:Nn \l_stex_get_symbol_arity_int {\l_stex_get_symbol_arity_int}
4212   \stex_debug:nn{vardef}{Doing~\l_stex_key_name_str}
4213   \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4214   \stex_notation_parse:n{#3}
4215   \stex_if_check_terms:T{ \_stex_notation_check: }
4216   \_stex_var_notation_macro:
4217   \stex_if_do_html:T {
4218     \def\comp{\_varcomp}
4219     \_stex_notation_do_html:n \l_stex_key_name_str
4220   }
4221   \group_begin:
4222   \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4223   \tl_clear:N \thisstyle
4224   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4225   \def\thisnotation{
4226     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{ }{
4227       \_stex_notation_make_args:
4228     }
4229   }
4230   \stex_style_apply:
4231   \group_end:\ignorespaces
4232 }{}

```

(End of definition for \vardef. This function is documented on page 140.)

sfunction

Adds a new variable to the current scope:

```

4233 \cs_new_protected:Nn \__stex_vars_add: {
4234   \exp_args:NNNo \exp_args:NNne
4235   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {

```



```

4236     {\l_stex_macroname_str}
4237     {\l_stex_key_name_str}
4238     {\int_use:N \l_stex_get_symbol_arity_int}
4239     {\l_stex_get_symbol_args_tl}
4240     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4241     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4242     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4243     \stex_invoke_symbol:
4244   }
4245 }

```

Defines the macro for a variable:

```

4246 \cs_new_protected:Nn \__stex_vars_macro: {
4247   \tl_set:ce{\l_stex_macroname_str}{
4248     \stex_invoke_variable:nnnnnnN
4249     {\l_stex_key_name_str}
4250     {\int_use:N \l_stex_get_symbol_arity_int}
4251     {\l_stex_get_symbol_args_tl}
4252     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4253     {\exp_args:No \exp_not:n \l_stex_key_type_tl}
4254     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4255     \stex_invoke_symbol:
4256   }
4257 }

```

Insert the HTML for a variable declaration:

```

4258
4259 \cs_new_protected:Nn \__stex_vars_html: {
4260   \stex_if_do_html:T {
4261     \hbox\bgroup\exp_args:Ne \stex_annotate_invisible:nn {
4262       data-ftml-vardef = {\l_stex_key_name_str},
4263       data-ftml-args = {\l_stex_key_args_str}
4264       \str_if_empty:NF \l_stex_macroname_str {,
4265         data-ftml-macroname={\l_stex_macroname_str}
4266       }
4267       \str_if_empty:NF \l_stex_key_assoc_str {,
4268         data-ftml-assoctype={\l_stex_key_assoc_str}
4269       }
4270       \str_if_empty:NF \l_stex_key_role_str {,
4271         data-ftml-role={\l_stex_key_role_str}
4272       }
4273       \str_if_empty:NF \l_stex_key_reorder_str {,
4274         data-ftml-reorderargs={\l_stex_key_reorder_str}
4275       }
4276       \bool_if:NT \l__stex_vars_bind_bool {,
4277         data-ftml-bind={}
4278       }
4279     }{
4280       \stex_annotate_force_break:n{
4281         \bool_set_true:N \stex_in_invisible_html_bool
4282         \tl_if_empty:NF \l_stex_key_type_tl {
4283           \stex_annotate:nn{data-ftml-type={}}{${\l_stex_key_type_tl$}
4284         }
4285         \tl_if_empty:NF \l_stex_key_def_tl {
4286           \stex_annotate:nn{data-ftml-definiens={}}{${\l_stex_key_def_tl$}

```

```

4287     }
4288     \tl_if_empty:NF \l_stex_key_return_tl{
4289       \exp_args:Nno \use:n{
4290         \cs_generate_from_arg_count:NNnn \l__stex_vars_cs
4291         \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4292         \tl_set:Ne \l__stex_vars_args_tl {\_stex_map_args:N \_stex_return_args:nn}
4293         $\stex_annotate:nn{data-ftml-returntype={}}{
4294           \exp_after:wN \l__stex_vars_cs \l__stex_vars_args_tl!}$
4295       }
4296     \tl_if_empty:NF \l_stex_key_argtypes_clist {
4297       \stex_annotate:nn{data-ftml-argtypes={}}{
4298         \_stex_annotate_force_break:n{
4299           \clist_map_inline:Nn \l_stex_key_argtypes_clist {
4300             $\stex_annotate:nn{data-ftml-type={}}{##1}$
4301           }
4302         }
4303       }
4304     }
4305   }
4306 } \egroup
4307 }
4308 }

```

\stex_get_var:n

```

4309 \cs_new_protected:Nn \stex_get_var:n {
4310   \str_clear:N \l_stex_get_variable_str
4311   \__stex_vars_get_var:n{#1}
4312   \str_if_empty:NT \l_stex_get_variable_str {
4313     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4314   }
4315 }
4316
4317 \cs_new_protected:Nn \__stex_vars_get_var:n {
4318   \prop_map_inline:Nn \l_stex_variables_prop {
4319     \__stex_vars_check_var:nnnnnnN {#1} ##2
4320   }
4321 }
4322
4323 \cs_new_protected:Nn \__stex_vars_check_var:nnnnnnN {
4324   \str_if_eq:nnTF{#1}{#2}{
4325     \prop_map_break:n{\_stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}
4326   }{
4327     \str_if_eq:nnT{#1}{#3}{
4328       \prop_map_break:n{\_stex_vars_set_vars:nnnnnnN {#3}{#4}{#5}{#6}{#7}{#8}{#9}
4329     }
4330   }
4331 }
4332
4333 \cs_new_protected:Nn \__stex_vars_set_vars:nnnnnnN {
4334   \stex_debug:nn{symbols}{Variable~#1~found}
4335   \cs_set:Npn \_stex_variable:nnnnnnN ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 {}
4336   \str_set:Nn \l_stex_get_variable_str {#1}
4337   \int_set:Nn \l_stex_get_symbol_arity_int {#2}
4338   \tl_set:Nn \l_stex_get_symbol_args_tl {#3}

```

```

4339 \tl_set:Nn \l_stex_get_symbol_def_tl {#4}
4340 \tl_set:Nn \l_stex_get_symbol_type_tl {#5}
4341 \tl_set:Nn \l_stex_get_symbol_return_tl {#6}
4342 \tl_set:Nn \l_stex_get_symbol_invoke_cs {#7}
4343 }

```

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

sfunction

32.3.1 Variable Sequences

```

4344 <@@=stex_seqs>

\varseq

4345 \stex_new_stylable_cmd:nnnn {varseq}{m 0{ m m} {
4346   \stex_keys_set:nn{symdef}{#2}
4347   \str_set:Ne \l_stex_macroname_str { #1 }
4348   \str_if_empty:NT \l_stex_key_name_str {
4349     \str_set:Ne \l_stex_key_name_str { #1 }
4350   }
4351   \str_if_empty:NT \l_stex_key_args_str {
4352     \str_set:Nn \l_stex_key_args_str {1}
4353   }
4354   \stex_symdecl_do:
4355
4356   \tl_set_eq:NN \l_stex_get_symbol_return_tl \l_stex_key_return_tl
4357   \clist_set:Nn \l__stex_seqs_range_clist {#3}
4358   \tl_if_empty:NTF \l_stex_key_op_tl {
4359     \stex_notation_parse:n{#4}
4360     \tl_clear:N \l_stex_key_op_tl
4361   }{
4362     \stex_notation_parse:n{#4}
4363   }
4364   \stex_if_do_html:T \__stex_seqs_html:
4365   \stex_if_check_terms:T \_stex_check_terms:
4366   \__stex_seqs_add:
4367   \__stex_seqs_macro:
4368   \stex_if_check_terms:T \_stex_notation_check:
4369   \_stex_var_notation_macro:
4370   \group_begin:
4371   \tl_set_eq:NN \thisvarname \l_stex_key_name_str
4372   \tl_clear:N \thisstyle
4373   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
4374   \def\thisnotation{
4375     \l_stex_notation_macrocode_cs{ }!
4376   }
4377   \stex_style_apply:
4378   \group_end:\ignorespaces
4379 }{ }

4380
4381 \cs_new_protected:Nn \__stex_seqs_add: {
4382   \exp_args:NNNo \exp_args:NNne
4383   \prop_put:Nnn \l_stex_variables_prop \l_stex_key_name_str {
4384     {\l_stex_macroname_str}

```

```

4385     {\l_stex_key_name_str}
4386     {\int_use:N \l_stex_get_symbol_arity_int}
4387     {\l_stex_get_symbol_args_tl}
4388     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4389     {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
4390     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4391     \stex_invoke_sequence:
4392   }
4393 }
4394
4395 \cs_new_protected:Nn \__stex_seqs_macro: {
4396   \tl_set:ce{\l_stex_macroname_str}{
4397     \stex_invoke_variable:nnnnnnN
4398     {\l_stex_key_name_str}
4399     {\int_use:N \l_stex_get_symbol_arity_int}
4400     {\l_stex_get_symbol_args_tl}
4401     {\exp_args:No \exp_not:n \l_stex_key_def_tl}
4402     {\exp_args:No \exp_not:n \l__stex_seqs_range_clist}
4403     {\exp_args:No \exp_not:n \l_stex_key_return_tl}
4404     \stex_invoke_sequence:
4405   }
4406 }
4407
4408 \cs_new_protected:Nn \__stex_seqs_html: {
4409   \exp_args:Ne \stex_annotate_invisible:nn {
4410     data-ftml-varseq = {\l_stex_key_name_str},
4411     data-ftml-args = {\l_stex_key_args_str}
4412     \str_if_empty:NF \l_stex_macroname_str {,
4413       data-ftml-macroname={\l_stex_macroname_str}
4414     }
4415     \str_if_empty:NF \l_stex_key_assoc_str {,
4416       data-ftml-assoctype={\l_stex_key_assoc_str}
4417     }
4418     \str_if_empty:NF \l_stex_key_role_str {,
4419       data-ftml-role={\l_stex_key_role_str}
4420     }
4421     \str_if_empty:NF \l_stex_key_reorder_str {,
4422       data-ftml-reorderargs={\l_stex_key_reorder_str}
4423     }
4424   }{\hbox\bgroup
4425     \stex_annotate_force_break:n{
4426       \tl_if_empty:NF \l_stex_key_type_tl {
4427         \stex_annotate:nn{data-ftml-type={}}{\l_stex_key_type_tl$}
4428       }
4429       \tl_if_empty:NF \l_stex_key_def_tl {
4430         \stex_annotate:nn{data-ftml-definiens={}}{\l_stex_key_def_tl$}
4431       }
4432       \tl_if_empty:NF \l_stex_key_return_tl{
4433         \exp_args:Nno \use:n{
4434           \cs_generate_from_arg_count:NNnn \l__stex_seqs_cs
4435           \cs_set:Npn \l_stex_get_symbol_arity_int} \l_stex_key_return_tl
4436           \tl_set:Nx \l__stex_seqs_args_tl {\stex_map_args:N \stex_return_args:nn}
4437           \stex_annotate:nn{data-ftml-returntype={}}{
4438             $\exp_after:wN \l__stex_seqs_cs \l__stex_seqs_args_tl!$}

```

```

4439     }
4440     \tl_if_empty:NF \l_stex_key_argtypes_clist {
4441       \stex_annotate:nn{data-ftml-argtypes={}}{
4442         \stex_annotate_force_break:n{
4443           \clist_map_inline:Nn \l_stex_key_argtypes_clist {
4444             \stex_annotate:nn{data-ftml-type={}}{###1$}
4445           }
4446         }
4447       }
4448     }
4449   }
4450   \egroup}
4451 }

```

(End of definition for \varseq. This function is documented on page 141.)

sfunction

32.4 Expressions

```

4452 <@@=stex_expr>

```

\symuse

```

4453 \cs_new_protected:Npn \symuse #1 {
4454   \stex_get_symbol:n{#1}
4455   \exp_args:Nno \use:n {\_stex_invoke_symbol:NeooooN
4456     \l_stex_get_symbol_uri
4457     {\int_use:N \l_stex_get_symbol_arity_int}
4458     \l_stex_get_symbol_args_tl
4459     \l_stex_get_symbol_def_tl
4460     \l_stex_get_symbol_type_tl
4461     \l_stex_get_symbol_return_tl}
4462   \l_stex_get_symbol_invoke_cs
4463 }

```

(End of definition for \symuse. This function is documented on page 141.)

sfunction

_stex_next_symbol:n

```

4464 \tl_new:N \l__stex_expr_reset_tl
4465
4466 \cs_new_protected:Nn \_stex_next_symbol:n {
4467   \tl_set:Ne \l_stex_every_symbol_tl {
4468     \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
4469       \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
4470         \exp_args:No \exp_not:n \l_stex_every_symbol_tl
4471       }
4472       \tl_gset:Nn \exp_not:N \l__stex_expr_reset_tl {
4473         \exp_args:No \exp_not:n \l__stex_expr_reset_tl
4474       }
4475     }
4476     \tl_set:Nn \exp_not:N \l_stex_every_symbol_tl {
4477       \exp_args:No \exp_not:n \l_stex_every_symbol_tl
4478     }
4479     \exp_not:n{ \aftergroup \l__stex_expr_reset_tl }

```

```

4480 \exp_not:N \l_stex_every_symbol_tl
4481 \exp_not:n{ #1 }
4482 }
4483 }
4484 \cs_generate_variant:Nn \stex_next_symbol:n {e}

```

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

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).

```

4485 \cs_new_protected:Nn \stex_invoke_symbol:nnnnnnN {
4486   \bool_if:NTF \l_stex_allow_semantic_bool{
4487     \group_begin:
4488     \tl_set:Nn \l_stex_current_symbol_uri {#1}
4489     \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_current_symbol_uri
4490     \tl_set:Nn \l_stex_current_display_tl {
4491       \stex_symbol_uri_name:N \l_stex_current_symbol_uri
4492     }
4493     \__stex_expr_invoke:nnnnN{#2}{#3}{#5}{#6}{#7}
4494   }{
4495     \msg_error:nnxx{stex}{error/notallowed}{
4496       \stex_use_symbol_uri:n{#1}
4497     }{
4498       \l_stex_current_full_tl
4499     }
4500   }
4501 }
4502
4503 \cs_new_protected:Npn \stex_invoke_symbol:NnnnnnnN {
4504   \exp_args:No \stex_invoke_symbol:nnnnnnN
4505 }
4506 \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:

```

4507 \bool_new:N \l_stex_allow_semantic_bool
4508 \bool_set_true:N \l_stex_allow_semantic_bool

```

Code to execute every time a semantic macro is invoked:

```

4509 \tl_set:Nn \l_stex_every_symbol_tl {
4510   \bool_set_false:N \l_stex_allow_semantic_bool
4511 }

```

The current top-level term; may be undefined:

```

4512 \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:

```
4513 \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

```
4514 \cs_new_protected:Nn \__stex_expr_invoke:nnnnN {
4515   \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
4516   \__stex_expr_setup:nnnnn{\_comp}{#1}{#2}{#4}{#3}
4517   \cs_set_eq:NN \stex_term_oms_or_omv:nn \stex_term_oms:nn
4518   \tl_put_right:Nn \l_stex_current_redo_tl{
4519     \cs_set_eq:NN \stex_term_oms_or_omv:nn \stex_term_oms:nn
4520   }
4521   #5
4522 }
```

general setup; takes: comp, arity, args, return, type

```
4523 \cs_new_protected:Nn \__stex_expr_setup:nnnnn {
4524   \tl_clear:N \l_stex_return_notation_tl
4525   \tl_set:Nn \l_stex_current_redo_tl {
4526     \let \this \stex_current_this:
4527     \def\comp{#1}
4528     \def\maincomp{\comp}
4529     \str_set:Nn \l_stex_current_arity_str{ #2 }
4530     \tl_set:Nn \l_stex_current_args_tl{ #3 }
4531     \tl_set:Nn \l_stex_current_return_tl{ #4 }
4532     \tl_set:Nn \l_stex_current_type_tl{ #5 }
4533     \tl_clear:N \l_stex_current_term_tl
4534   }
4535   \tl_put_right:No \l_stex_current_redo_tl \l_stex_every_symbol_tl
4536   \tl_put_left:Ne \l_stex_current_redo_tl {
4537     \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl}
4538   }
4539   \l_stex_current_redo_tl
4540 }
```

(End of definition for `\stex_invoke_symbol:NnnnnnN` and `\stex_invoke_symbol:nnnnnnN`. These functions are documented on page 141.)

sfunction

`\stex_invoke_symbol:` Math or text mode?

```
4541 \cs_new_protected:Nn \stex_invoke_symbol: {
4542   \stex_debug:nn{expressions}{Invoking~ \l_stex_current_full_tl}
4543   \mode_if_math:TF \__stex_expr_math: \__stex_expr_text:
4544 }
```

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

sfunction

`\stex_invoke_text_symbol:`

```
4545 \cs_new_protected:Nn \stex_invoke_text_symbol: {
4546   \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
4547   \stex_term_oms_or_omv:nn{\maincomp{\let\xspace\relax\l_stex_current_return_tl}}
4548   \group_end:\mode_if_math:F{\cs_if_exist:NT\xspace\xspace}
4549 }
```

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

sfunction

`\stex_invoke_sequence`:

```

4550 \cs_new_protected:Nn \stex_invoke_sequence: {
4551   \peek_charcode_remove:NTF ! {
4552     \peek_charcode:NTF [ \__stex_expr_seq_op:w { \__stex_expr_seq_op:w [] }
4553   }\__stex_expr_seq_first:
4554 }
4555
4556 \cs_new_protected:Npn \__stex_expr_seq_op:w [#1] {
4557   \stex_use_op_notation:nnTF \l_stex_current_full_tl{#1}{
4558     %\stex_debug:nn{vars}{Here: \meaning\l_stex_notation_cs}
4559     \stex_maybe_brackets:nn{\neginfprec}{
4560       \stex_term_oms_or_omv:nn{#1}
4561       {\l_stex_notation_cs}\group_end:
4562     }
4563   }{
4564     \__stex_expr_get_index_notation:n{#1}
4565     \peek_charcode:NTF [ \__stex_expr_range:w { \__stex_expr_range:w[] }
4566   }
4567 }
4568
4569 \cs_new_protected:Nn \__stex_expr_get_index_notation:n {
4570   \stex_use_notation:nnTF \l_stex_current_full_tl{#1}{}{
4571     \stex_do_default_notation:
4572     \cs_set_eq:NN \l_stex_notation_cs \l_stex_default_notation
4573   }
4574 }
4575
4576 \cs_new_protected:Npn \__stex_expr_range:w [#1] {
4577   \bool_set_true:N \l_stex_allow_semantic_bool
4578   \clist_clear:N \l__stex_expr_clist
4579   \clist_map_function:NN \l_stex_current_type_tl \__stex_expr_seq_arg:n
4580   \stex_annotate:nn{
4581     data-ftml-term=OMV,
4582     data-ftml-head={\l_stex_current_full_tl},
4583     data-ftml-notationid={}
4584   }{
4585     \l__stex_expr_clist
4586   }
4587   \group_end:
4588 }
4589
4590 \cs_new_protected:Nn \__stex_expr_seq_arg:n {
4591   \tl_if_eq:nnTF{#1}{\ellipses}{
4592     \clist_put_right:Nn \l__stex_expr_clist {
4593       \ellipses
4594     }
4595   }{
4596     \clist_put_right:Nn \l__stex_expr_clist {
4597       \exp_args:No \str_if_eq:nnTF \l_stex_current_arity_str {1}{
4598         \group_begin:
4599         \l_stex_notation_cs \group_end: {#1}

```



```

4600     }{
4601     \group_begin:
4602     \l_stex_notation_cs \group_end: #1
4603     }
4604   }
4605 }
4606 }
4607 }
4608
4609 \cs_new_protected:Nn \__stex_expr_seq_first: {
4610   \exp_args:Nne \use:nn{
4611     \cs_generate_from_arg_count:NNnn \l_stex_notation_cs \cs_set:Npn
4612     \l_stex_current_arity_str} {{
4613     \tl_set:Nn \exp_not:N \l__stex_expr_first_args_tl {
4614       \int_step_function:nN \l_stex_current_arity_str \__stex_expr_do_first_arg:n
4615     }
4616     \exp_not:N \__stex_expr_do_first_next:
4617   }}
4618   \l_stex_notation_cs
4619 }
4620
4621 \cs_new:Nn \__stex_expr_do_first_arg:n {\exp_not:n{## #1}}
4622
4623 \cs_new_protected:Nn \__stex_expr_do_first_next: {
4624   \peek_charcode_remove:NTF ! {
4625     \peek_charcode:NTF [ \__stex_expr_do_one:w {\__stex_expr_do_one:w []}
4626   }{
4627     \peek_charcode:NTF [ \__stex_expr_do_all:w {\__stex_expr_do_all:w []}
4628   }
4629 }
4630
4631 \cs_new_protected:Npn \__stex_expr_do_one:w [#1] {
4632   \__stex_expr_get_index_notation:n{#1}
4633   \exp_args:Nno\use:nn{\l_stex_notation_cs\group_end:}\l__stex_expr_first_args_tl
4634 }
4635
4636 \cs_new_protected:Npn \__stex_expr_do_all:w [#1] {
4637   \exp_args:Nno\use:nn{\__stex_expr_notation:w [#1]}\l__stex_expr_first_args_tl
4638 }

```

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

sfunction

`\stex_invoke_structure:`

```

4639 \cs_new_protected:Nn \stex_invoke_structure: {
4640   \tl_set:Nn \l__stex_expr_set_comp_tl {\__stex_expr_set_thiscomp:}
4641   \__stex_expr_struct_top:n {}
4642 }
4643
4644 \cs_new_protected:Nn \__stex_expr_set_thiscomp: {
4645   \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_thiscomp {
4646     \edef\maincomp {\_thiscomp{\comp}}
4647   }
4648 }

```

```

4649
4650 \cs_new_protected:Nn \__stex_expr_struct_top:n {
4651   \stex_debug:nn{structure}{
4652     invoking~structure~{\l_stex_current_type_tl}<\tl_to_str:n{#1}>
4653   }
4654   \peek_charcode:NTF [ {
4655     \__stex_expr_merge:nw{#1}
4656   }{
4657     \__stex_expr_struct_type:n {#1}
4658     \tl_set:Nn \l_stex_struct_this_tl {}
4659     \peek_charcode_remove:NTF ! {
4660       \peek_charcode:NTF [ {
4661         \__stex_expr_maybe_notation:w
4662       }{
4663         \__stex_expr_maybe_notation:w []
4664       }
4665     }{
4666       \__stex_expr_invoke_this:n
4667     }
4668   }
4669 }
4670
4671 \cs_new_protected:Npn \__stex_expr_merge:nw #1 [ #2 ] {
4672   \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{comp=}{
4673     \__stex_expr_set_customcomp: #2 \__stex_expr_end:
4674     \__stex_expr_struct_top:n{#1}
4675   }{
4676     \exp_args:Ne \stex_str_if_starts_with:nnTF {\tl_to_str:n{#2}}{this=}{
4677       \__stex_expr_set_thisnotation: #2 \__stex_expr_end:
4678       \__stex_expr_struct_top:n{#1}
4679     }{
4680       \tl_if_empty:nTF{#1}{
4681         \__stex_expr_struct_top:n{#2}
4682       }{
4683         \tl_if_empty:nTF{#2}{
4684           \__stex_expr_struct_top:n{#1}
4685         }{
4686           \__stex_expr_struct_top:n{#1,#2}
4687         }
4688       }
4689     }
4690   }
4691 }
4692
4693 \cs_new_protected:Npn \__stex_expr_set_thisnotation: this= #1 \__stex_expr_end: {
4694   \tl_set:Nn \l_stex_return_notation_tl { \comp{#1} }
4695   \tl_set:Nn \l__stex_expr_set_comp_tl {}
4696 }
4697
4698 \cs_new_protected:Npn \__stex_expr_set_customcomp: comp= #1 \__stex_expr_end: {
4699   \tl_set:Nn \l__stex_expr_set_comp_tl {
4700     \__stex_expr_set_custom_comp:n{#1}
4701   }
4702   \tl_set:Nn \l_stex_return_notation_tl { \comp{} }

```

4703 }

The structure type:

```
4704 \cs_new_protected:Nn \__stex_expr_struct_type:n {
4705   \__stex_expr_do_assign_list:n{#1}
4706   \clist_if_empty:NTF \l__stex_expr_fields_clist {
4707     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
4708       = 1 {
4709         \tl_set:Ne \l__stex_expr_current_type_tl {
4710           \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl }
4711           \exp_args:No \exp_not:n \l_stex_current_redo_tl
4712           \stex_term_oms_or_omv:nn{}{}
4713         }
4714       }{
4715         \exp_args:No \__stex_expr_make_type:n \l_stex_current_type_tl
4716       }
4717   }{
4718     \int_compare:nNnTF {\clist_count:N \l_stex_current_type_tl}
4719       = 1 {
4720         \__stex_expr_make_type:n {}
4721       }{
4722         \exp_args:No \__stex_expr_make_type:n \l_stex_current_type_tl
4723       }
4724   }
4725 }
4726
4727 \cs_new_protected:Nn \__stex_expr_do_assign_list:n {
4728   \clist_clear:N \l__stex_expr_fields_clist
4729   \tl_if_empty:nF {#1} {
4730     \keyval_parse:NNn\TODO\__stex_expr_do_assign:nn{#1}
4731   }
4732 }
4733
4734 \cs_new_protected:Nn \__stex_expr_do_assign:nn {
4735   \clist_put_right:Nn \l__stex_expr_fields_clist {{#1}{#2}}
4736 }
4737
4738 \cs_new_protected:Nn \__stex_expr_make_type:n {
4739   \tl_if_empty:nTF{#1}{
4740     \seq_clear:N \l_tmpa_seq
4741   }{
4742     \seq_set_split:Nnn \l_tmpa_seq ,{#1}
4743     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
4744     \seq_reverse:N \l_tmpa_seq
4745   }
4746   \tl_set:Ne \l__stex_expr_current_type_tl {
4747     \symuse{Metatheory?module~type~merge}{
4748       {
4749         \exp_args:No \exp_not:n \l_stex_current_redo_tl
4750         \stex_term_oms_or_omv:nn{}{}
4751       }
4752     \seq_map_function:NN \l_tmpa_seq \__stex_expr_make_mod:n
4753     \clist_if_empty:NF \l__stex_expr_fields_clist {
4754       ,\symuse{Metatheory?anonymous~record}{
4755         \exp_args:Ne \tl_tail:n{
```

```

4756         \clist_map_function:NN \l__stex_expr_fields_clist \__stex_expr_make_oml:n
4757     }
4758 }
4759 }
4760 }
4761 }
4762 }
4763
4764 \cs_new:Nn \__stex_expr_make_mod:n {
4765     ,\symuse{Metatheory?module~type}{
4766         \exp_args:Ne \stex_annotate:nn{data-ftml-term=OMMOD,data-ftml-head={\stex_use_module_ur
4767     }
4768 }
4769
4770
4771 \cs_new:Nn \__stex_expr_make_oml:n {
4772     \__stex_expr_make_oml:nn #1
4773 }
4774 \cs_new:Nn \__stex_expr_make_oml:nn {
4775     ,\stex_annotate:nn{
4776         data-ftml-term=OML,
4777         data-ftml-head={#1}
4778     }{
4779         \stex_annotate_force_break:n{
4780             \stex_annotate:nn{data-ftml-definiens={}}{\exp_not:n{#2!}}
4781         }
4782     }
4783 }

```

Insert the structure type as a term:

```

4784 \cs_new:Nn \__stex_expr_current_type: {
4785     %\exp_args:No \exp_not:n \l_stex_current_redo_tl
4786     \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
4787         \exp_args:No\exp_not:n\l__stex_expr_current_type_tl
4788     }
4789     \stex_term_oms_or_omv:nn{}{}
4790 }

```

The structure type itself:

```

4791 \cs_new_protected:Npn \__stex_expr_maybe_notation:w [ #1 ] {
4792     \tl_set_eq:NN \l_stex_current_term_tl \l__stex_expr_current_type_tl
4793     \stex_use_notation:nnTF \l_stex_current_full_tl {#1}{
4794         \l_stex_notation_cs\group_end:
4795     }{
4796         \__stex_expr_make_prop:
4797         \__stex_expr_make_prop_assign:
4798         \__stex_expr_present_i:w [ #1 ]
4799     }
4800 }
4801
4802 \cs_new_protected:Nn \__stex_expr_present: {
4803     \peek_charcode:NTF [ {
4804         \__stex_expr_present_i:w
4805     }{
4806         \__stex_expr_present:nn{}{}

```

```

4807   }
4808 }
4809
4810 \cs_new_protected:Npn \__stex_expr_present_i:w [#1] {
4811   \int_compare:nNnTF{\clist_count:n{#1}} = 1 {
4812     \__stex_expr_present:nn{#1}
4813   }{
4814     \peek_charcode:NTF [ {
4815       \__stex_expr_present_ii:nw{#1}
4816     }{
4817       \__stex_expr_present:nn{#1}{}
4818     }
4819   }
4820 }
4821
4822 %First: clist, second:notation-id
4823 \cs_new_protected:Npn \__stex_expr_present_ii:nw #1 [#2] {
4824   \__stex_expr_present:nn{#1}{#2}
4825 }
4826
4827 \cs_new_protected:Nn \__stex_expr_present:nn {
4828   \clist_clear:N \l__stex_expr_clist
4829   \tl_if_empty:nTF{#1}{
4830     \cs_set:Npn \l__stex_expr_cs ##1 ##2 ##3 {
4831       \tl_if_empty:nF{##2}{
4832         \__stex_expr_present_entry:nn {##1}{##3}
4833       }
4834     }
4835   }{
4836     \cs_set:Npn \l__stex_expr_cs ##1 ##2 ##3 {
4837       \exp_args:Ne \clist_if_in:nnT{\tl_to_str:n{#1}}{##1}{
4838         \__stex_expr_present_entry:nn {##1}{##3}
4839       }
4840     }
4841   }
4842   \prop_map_inline:Nn \l__stex_expr_prop {
4843     \l__stex_expr_cs {##1} ##2
4844   }
4845   \stex_term_oms_or_omv:nn{}{
4846     \exp_args:Nno \use:n{
4847       \bool_set_true:N \l_stex_allow_semantic_bool
4848       \symuse{Metatheory?mathematical~structure}[#2]
4849     }{\l__stex_expr_clist}
4850   }\group_end:
4851 }
4852
4853 \cs_new_protected:Nn \__stex_expr_present_entry:nn {
4854   \seq_if_in:NnTF \l__stex_expr_assigned_seq {#1}{
4855     \clist_put_right:Nn \l__stex_expr_clist {#2!}
4856   }{
4857     \exp_args:Nne \clist_put_right:Nn \l__stex_expr_clist {
4858       \stex_next_symbol:n {
4859         \exp_args:No \exp_not:n \l__stex_expr_set_comp_tl
4860         \tl_set:Nn \exp_not:N \l_stex_struct_this_tl {

```

```

4861         \exp_args:No \exp_not:n \l_stex_struct_this_tl
4862     }
4863     \exp_not:n {
4864         \tl_set_eq:NN \this \l_stex_struct_this_tl
4865     }
4866     \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
4867         \exp_args:No \exp_not:n \l_stex_return_notation_tl
4868     }
4869 }
4870 \exp_not:n{#2!}
4871 }
4872 }
4873 }
4874
4875 %\cs_new_protected:Nn \__stex_expr_set_custom_comp:n {
4876 % \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
4877 % \cs_set_protected:Npx \_customthiscomp ##1 {
4878 % \group_begin:
4879 % \bool_set_true:N \l_stex_allow_semantic_bool
4880 % \exp_not:n{
4881 % \cs_set:Npn \l__stex_expr_comp_cs ##1 {
4882 % #1
4883 % }
4884 % \def\maincomp
4885 % }{\comp}
4886 % \exp_not:N\l__stex_expr_comp_cs{\comp{##1}}
4887 % \group_end:
4888 % }
4889 % \def\maincomp {\_customthiscomp}
4890 % }
4891 %}
4892
4893 \cs_new_protected:Nn \__stex_expr_set_custom_comp:n {
4894 \exp_args:Ne \tl_if_eq:NNF {\tl_head:N \maincomp} \_customthiscomp {
4895 \stex_debug:nn{structs}{Setting~custom~comp:~\tl_to_str:n{#1}}
4896 \exp_args:Nne \__stex_expr_set_custom_i:nn{#1}{\comp}
4897 \def\maincomp {\_customthiscomp}
4898 }
4899 }
4900
4901 \cs_new_protected:Nn \__stex_expr_set_custom_i:nn {
4902 \cs_set_protected:Npn \_customthiscomp ##1 {
4903 \group_begin:
4904 \bool_set_true:N \l_stex_allow_semantic_bool
4905 \cs_set:Npn \l__stex_expr_comp_cs ####1 { #1 }
4906 \def \maincomp {#2}
4907 \l__stex_expr_comp_cs{#2{##1}}
4908 \group_end:
4909 }
4910 }
4911
4912 this (of type structure):
4913 \cs_new_protected:Nn \__stex_expr_invoke_this:n {
4914 \peek_charcode_remove:NTF ! {
4915 \exp_args:Nne\use:nn{

```

```

4914     \group_end:\symuse{Metatheory?of~type}[invisible]{
4915       \tl_if_empty:nTF{#1}{\stex_annotate_force_break:n{}}{#1}
4916     }
4917   }{
4918     {
4919       \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl}
4920       \stex_expr_current_type:
4921     }
4922   }
4923 }{
4924   \stex_expr_invoke_maybe_field:nn{#1}
4925 }
4926 }
4927
4928 \cs_new_protected:Nn \stex_expr_invoke_maybe_field:nn {
4929   \stex_expr_make_prop:
4930   \stex_expr_set_this:n{#1}
4931   \tl_if_empty:nTF{#2}{
4932     \stex_expr_make_prop_assign:
4933     \stex_expr_present:
4934   }{
4935     \stex_expr_invoke_field:n{#2}
4936   }
4937 }
4938
4939 \cs_new_protected:Nn \stex_expr_set_this:n {
4940   \tl_if_empty:nTF{#1}{
4941     %\tl_put_right:Nn \l_stex_current_redo_tl {
4942     % \tl_clear:N \l_stex_struct_this_tl
4943     %}
4944   }{
4945     \tl_set:Ne \l_stex_struct_this_tl {{
4946       \bool_set_true:N \l_stex_allow_semantic_bool
4947       \bool_set_true:N \l_stex_in_this_bool
4948       \tl_set:Nn \exp_not:N \this {
4949         \exp_args:No \exp_not:n \this
4950       }
4951       \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl }
4952       \exp_not:n{#1}
4953     }}
4954     \tl_set_eq:NN \this \l_stex_struct_this_tl
4955     % \l_stex_return_notation_tl
4956   }
4957 }
4958
4959 \cs_new_protected:Nn \stex_expr_get_field_name:n {
4960   \str_set:Ne \l_stex_expr_field_name_str {
4961     \exp_args:Nne \use:n {\exp_after:wN \use_i:nn \use:n}
4962     {\prop_item:Nn \l_stex_expr_prop {#1}}
4963   }
4964   \str_if_empty:NT \l_stex_expr_field_name_str {
4965     \str_set:Nn \l_stex_expr_field_name_str {#1}
4966   }
4967 }

```

```

4968
4969 \cs_new_protected:Nn \__stex_expr_invoke_field:n {
4970   \prop_if_in:NnTF \l__stex_expr_prop {#1}{
4971     \__stex_expr_get_field_name:n{#1}
4972     \tl_clear:N \l__stex_expr_more_nextsymbol_tl
4973     %\exp_args:NNe \seq_if_in:NnF \l__stex_expr_assigned_seq {\tl_to_str:n{#1}}{
4974       \tl_set:Ne \l__stex_expr_more_nextsymbol_tl {
4975         \tl_set:Nn \exp_not:N \l_stex_struct_this_tl {
4976           \exp_args:No \exp_not:n \l_stex_struct_this_tl
4977         }
4978         \exp_not:n {
4979           \tl_set_eq:NN \this \l_stex_struct_this_tl
4980         }
4981         \tl_set:Nn \exp_not:N \l_stex_return_notation_tl {
4982           \exp_args:No \exp_not:n \l_stex_return_notation_tl
4983         }
4984         \exp_args:No \exp_not:n \l__stex_expr_set_comp_tl
4985       }
4986     %}
4987     \exp_args:NNe \use:nn \group_end: {
4988       \_stex_next_symbol:n {
4989         \exp_args:No \exp_not:n \l__stex_expr_redo_tl
4990         \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
4991           \symuse{Metatheory?record~field}{
4992             \symuse{Metatheory?of~type}{
4993               \exp_args:No\tl_if_empty:nTF \l_stex_struct_this_tl {\_stex_annotate_force_br
4994             }{
4995               \tl_set:Nn \exp_not:N \l_stex_current_full_tl { \l_stex_current_full_tl}
4996               \__stex_expr_current_type:
4997             }
4998           }{
4999             \stex_annotate:nn{data-ftml-term=OML,data-ftml-head={\l__stex_expr_field_name_s
5000           }
5001         }
5002         \exp_args:No \exp_not:n \l__stex_expr_more_nextsymbol_tl
5003       }
5004       \exp_not:N \use_ii:nn
5005       \prop_item:Nn \l__stex_expr_prop {#1}
5006     }
5007   }{
5008     \msg_error:nnn{stex}{error/unknownfield}{#1}
5009   }
5010 }
5011
5012 \cs_new_protected:Nn \__stex_expr_make_prop: {
5013   \prop_clear:N \l__stex_expr_prop
5014   \seq_clear:N \l__stex_expr_seq
5015   \seq_clear:N \l__stex_expr_assigned_seq
5016   \tl_clear:N \l__stex_expr_redo_tl
5017   \__stex_expr_prop_do_decls:
5018 }
5019
5020 \cs_new_protected:Nn \__stex_expr_prop_do_decls: {
5021   \group_begin:

```



```

5022 \stex_debug:nn{expressions}{constructing~record}
5023 \seq_clear:N \l_stex_all_module_seq
5024 \exp_args:Ne \stex_activate_module:n {\clist_item:Nn \l_stex_current_type_tl 1}
5025 \exp_args:No \stex_iterate_symbols:nn \l_stex_current_type_tl {
5026   % uri, macro name, name, arity, args, type, def, return, macro
5027   \tl_if_empty:nTF{##2}{
5028     \exp_args:Nne\__stex_expr_do_decl_nomacro:nnnnnnnn{##3}
5029   }{
5030     \exp_args:Nne\__stex_expr_do_decl:nnnnnnnn{##2}
5031   }
5032   {\stex_new_symbol_uri:nn{##1}{##3}}{##3}{##4}{##5}{##6}{##7}{##8}##9
5033 }
5034 \exp_after:wN \tl_set:Nn \exp_after:wN \l__stex_expr_after_tl \exp_after:wN {
5035 \exp_after:wN \tl_set:Nn \exp_after:wN \l__stex_expr_prop \exp_after:wN { \l__stex_ex
5036 }
5037 \__stex_expr_prop_do_notations:
5038 \stex_debug:nn{expressions}{record:~\meaning\l__stex_expr_after_tl}
5039 \tl_put_right:Ne \l__stex_expr_after_tl {\tl_set:Nn \exp_not:N \l__stex_expr_redo_tl {\l
5040 \exp_after:wN \group_end: \l__stex_expr_after_tl
5041 }
5042
5043 % id, uri, name, arity, args, type, def, return, macro
5044 \cs_new_protected:Nn \__stex_expr_do_decl_nomacro:nnnnnnnn {
5045   \prop_if_in:NnF \l__stex_expr_prop {#1} {
5046     \seq_put_left:Ne \l__stex_expr_seq {\tl_to_str:n{#2}}
5047     \prop_put:Nnn \l__stex_expr_prop {#1}{
5048       {}{
5049         \stex_invoke_symbol:nnnnnnN
5050         {#2}
5051         {#4}
5052         {#5}{#6}{#7}{#8}#9
5053       }
5054     }
5055   }
5056 }
5057
5058 \cs_new_protected:Nn \__stex_expr_do_decl:nnnnnnnn {
5059   \prop_if_in:NnF \l__stex_expr_prop {#1} {
5060     \seq_put_left:Ne \l__stex_expr_seq {\tl_to_str:n{#2}}
5061     \prop_put:Nnn \l__stex_expr_prop {#1}{
5062       {#3}{
5063         \stex_invoke_symbol:nnnnnnN
5064         {#2}
5065         {#4}
5066         {#5}{#6}{#7}{#8}#9
5067       }
5068     }
5069   }
5070 }
5071
5072 \cs_new_protected:Nn \__stex_expr_make_prop_assign: {
5073   \clist_if_empty:NF \l__stex_expr_fields_clist {
5074     \clist_map_inline:Nn \l__stex_expr_fields_clist {
5075       \__stex_expr_make_prop_assign:nn ##1

```

```

5076     }
5077   }
5078 }
5079
5080 \cs_new_protected:Nn \__stex_expr_make_prop_assign:nn {
5081   \prop_if_in:NnTF \l__stex_expr_prop {#1}{
5082     \exp_args:NNe \seq_put_right:Nn \l__stex_expr_assigned_seq {\tl_to_str:n{#1}}
5083     \exp_args:Nne \use:nn {\__stex_expr_make_prop_assign_replace:nnnn {#1}{#2}}
5084     {\prop_item:Nn \l__stex_expr_prop {#1}}
5085   }{
5086     \msg_error:nnn{stex}{error/unknownfieldass}{#1}
5087   }
5088 }
5089 \cs_new_protected:Nn \__stex_expr_make_prop_assign_replace:nnnn {
5090   \prop_put:Nnn \l__stex_expr_prop {#1}{#{3}{#2}}
5091   \tl_if_empty:nF{#3}{
5092     \tl_set:cn{#1}{ #2 }
5093     \tl_put_right:Nn \l__stex_expr_redo_tl {
5094       \tl_set:cn{#1}{ #2 }
5095     }
5096   }
5097 }
5098
5099 \cs_new_protected:Nn \__stex_expr_prop_do_notations: {
5100   \exp_args:No \stex_iterate_notations:nn \l_stex_current_type_tl {
5101     \exp_args:NNe \seq_if_in:NnTF \l__stex_expr_seq {\tl_to_str:n{##1}}{
5102       \tl_set:Nn \l_tmpa_tl {
5103         \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}##2_cs}{
5104           \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}##2_cs}{##4}
5105         }
5106         \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}?_cs}{
5107           \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}?_cs}{##4}
5108         }
5109       }
5110       \exp_args:NNo \tl_put_right:Nn \l__stex_expr_redo_tl \l_tmpa_tl
5111       \exp_args:NNo \tl_put_right:Nn \l__stex_expr_after_tl \l_tmpa_tl
5112       \tl_if_empty:nF{##5}{
5113         \tl_set:Nn \l_tmpa_tl {
5114           \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}##2_op_cs}{
5115             \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}##2_op_cs}{##5}
5116           }
5117           \cs_if_exist:cF{l_stex_notation_\stex_use_symbol_uri:n{##1}?_op_cs}{
5118             \tl_set:cn{l_stex_notation_\stex_use_symbol_uri:n{##1}?_op_cs}{##5}
5119           }
5120         }
5121         \exp_args:NNo \tl_put_right:Nn \l__stex_expr_redo_tl \l_tmpa_tl
5122         \exp_args:NNo \tl_put_right:Nn \l__stex_expr_after_tl \l_tmpa_tl
5123       }
5124     }
5125   }
5126 }

```

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

sfunction

`\stex_invoke_variable:nnnnnn`

```

5127 \cs_new_protected:Nn \stex_invoke_variable:nnnnnnN {
5128   \bool_if:NTF \l_stex_allow_semantic_bool{
5129     \group_begin:
5130     \tl_set:Nn \l_stex_current_full_tl { #1 }
5131     \tl_set:Nn \l_stex_current_display_tl { #1 }
5132     \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
5133     \__stex_expr_setup:nnnnn{\_varcomp}{#2}{#3}{#6}{#5}
5134     \cs_set_eq:NN \stex_term_oms_or_omv:nn \stex_term_omv:nn
5135     \tl_put_right:Nn \l_stex_current_redo_tl {
5136       \cs_set_eq:NN \stex_term_oms_or_omv:nn \stex_term_omv:nn
5137     }
5138     #7
5139   }{
5140     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_full_tl}
5141   }
5142 }

```

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

sfunction

`\svar`

```

5143 \NewDocumentCommand \svar {0{} m}{
5144   \group_begin:
5145   \tl_if_empty:nTF{#1}{
5146     \tl_set:Nn \l_stex_current_full_tl { #2 }
5147     \tl_set:Nn \l_stex_current_display_tl { #2 }
5148   }{
5149     \tl_set:Nn \l_stex_current_full_tl { #1 }
5150     \tl_set:Nn \l_stex_current_display_tl { #1 }
5151   }
5152   \bool_if:NTF \l_stex_allow_semantic_bool{
5153     \tl_clear:N \l_stex_current_term_tl
5154     \stex_term_omv:nn{}{\_varcomp{#2}}
5155   }{
5156     \msg_error:nnxx{stex}{error/notallowed}{Variable}{\l_stex_current_full_tl}
5157   }
5158   \group_end:
5159 }

```

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

sfunction

Now for the brunt of the work:

Math

Modifiers (! or *)?

```

5160 \cs_new_protected:Nn \__stex_expr_math: {
5161   \stex_debug:nn{expressions}{math-mode}
5162   \peek_charcode_remove:NTF ! {
5163     % operator
5164     \peek_charcode_remove:NTF * \__stex_expr_op_custom:n {
5165       % op notation
5166       \peek_charcode:NTF [ \__stex_expr_op_notation:w {

```

```

5167     \_stex_expr_op_notation:w []
5168   }
5169 }
5170 }{
5171   \peek_charcode_remove:NTF * \_stex_expr_custom:n {
5172     % normal
5173     \peek_charcode:NTF [ \_stex_expr_notation:w {
5174       \_stex_expr_notation:w []
5175     }
5176   }
5177 }
5178 }

```

Text

Operator?

```

5179 \cs_new_protected:Nn \_stex_expr_text: {
5180   \stex_debug:nn{expressions}{text~mode}
5181   \peek_charcode_remove:NTF ! \_stex_expr_op_custom:n \_stex_expr_custom:n
5182 }

```

Notation

Operator?

```

5183 \cs_new_protected:Npn \_stex_expr_notation:w [ #1 ] {
5184   \peek_charcode_remove:NTF ! {
5185     \_stex_expr_op_notation:w [#1]
5186   }{
5187     \_stex_expr_full_notation:n {#1}
5188   }
5189 }

```

(Possibly) complex notation taking arguments with optional particular identifier:

```

5190 \cs_new_protected:Nn \_stex_expr_full_notation:n {
5191   \stex_use_notation:nnTF \l_stex_current_full_tl {#1}{
5192     \stex_debug:nn{expressions}{using~notation~"#1"~for~\l_stex_current_full_tl}
5193     \tl_if_empty:NTF \l_stex_current_return_tl {
5194       \stex_debug:nn{expressions}{return~empty}
5195       \l_stex_notation_cs{\group_end:\stex_eat_exclamation_point:}
5196     }{
5197       \stex_debug:nn{expressions}{return?}
5198       \exp_after:wN \_stex_expr_maybe_return:n \exp_after:wN {
5199         \l_stex_notation_cs{
5200       }
5201     }
5202   }{
5203     \stex_debug:nn{expressions}{notation~"#1"~for~\l_stex_current_full_tl}~does~not~exist;
5204     \stex_do_default_notation:
5205     \tl_if_empty:NTF \l_stex_current_return_tl {
5206       \l_stex_default_notation{\group_end:\stex_eat_exclamation_point:}
5207     }{
5208       \exp_after:wN
5209       \_stex_expr_maybe_return:n
5210       \exp_after:wN
5211       {\l_stex_default_notation {}}

```

```

5212     }
5213   }
5214 }

Operator notation:
5215 \cs_new_protected:Npn \__stex_expr_op_notation:w [#1] {
5216   \stex_debug:nn{expressions}{op~notation~for~\l_stex_current_full_tl}
5217   \stex_use_op_notation:nnTF \l_stex_current_full_tl {#1}{
5218     \stex_maybe_brackets:nn{\neginfprec}{
5219       \stex_term_oms_or_omv:nn{#1}
5220       {\l_stex_notation_cs}
5221     }
5222   \group_end:
5223 }{
5224   \int_compare:nNnTF \l_stex_current_arity_str = 0 {
5225     \tl_clear:N \l_stex_current_return_tl
5226     \__stex_expr_notation:w [#1]
5227   }{
5228     \stex_do_default_notation_op:
5229     \stex_maybe_brackets:nn{\neginfprec}{
5230       \stex_term_oms_or_omv:nn{#1}
5231       {\l_stex_default_notation}
5232     }
5233   \group_end:
5234 }
5235 }
5236 }

```

Custom Notations

Operator:

```

5237 \cs_new_protected:Nn \__stex_expr_op_custom:n {
5238   \stex_debug:nn{expressions}{custom-op}
5239   \bool_set_true:N \l_stex_allow_semantic_bool
5240   \stex_term_oms_or_omv:nn{}{\maincomp{#1}}
5241   \group_end:
5242 }

```

Complex:

```

5243 \int_new:N \l__stex_expr_arg_counter_int
5244
5245 \cs_new_protected:Nn \__stex_expr_custom:n {
5246   \stex_debug:nn{custom}{custom~notation~for~\l_stex_current_full_tl}
5247   \stex_pseudogroup:nn{
5248     \bool_set_true:N \l_stex_allow_semantic_bool
5249     \prop_gclear:N \l__stex_expr_customs_prop
5250     \seq_gclear:N \l__stex_expr_customs_seq
5251     \int_gzero:N \l__stex_expr_arg_counter_int
5252     \tl_if_empty:NF \l_stex_current_args_tl {
5253       \exp_after:wN \__stex_expr_add_prop_arg:nnw \l_stex_current_args_tl \stex_args_end:
5254       \cs_set_eq:NN \arg \__stex_expr_arg:n
5255     }
5256     \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
5257     \cs_set_eq:NN \__stex_expr_do_ab_next:nn \stex_term_oma:nn
5258     \stex_map_args:N \__stex_expr_check_b:nn

```

```

5259 \__stex_expr_do_ab_next:nn{ }{#1}
5260 }{
5261 \prop_if_exist:NT \l__stex_expr_customs_prop {
5262 \prop_gset_from_keyval:Nn \exp_not:N \l__stex_expr_customs_prop {
5263 \prop_to_keyval:N \l__stex_expr_customs_prop
5264 }
5265 }
5266 \int_gset:Nn \l__stex_expr_arg_counter_int { \int_use:N \l__stex_expr_arg_counter_int}
5267 \seq_if_exist:NT \l__stex_expr_customs_seq {
5268 \seq_gset_split:Nnn \exp_not:N \l__stex_expr_customs_seq , {
5269 \seq_use:Nn \l__stex_expr_customs_seq ,
5270 }
5271 }
5272 }
5273 % TODO check that all arguments are present
5274 \group_end:
5275 }
5276
5277 \cs_new_protected:Npn \__stex_expr_add_prop_arg:nnw #1 #2 #3\__stex_args_end: {
5278 \prop_gput:Nnn \l__stex_expr_customs_prop {#1} {}
5279 \seq_gput_right:Nn \l__stex_expr_customs_seq {#2}
5280 \tl_if_empty:nF{#3}{\__stex_expr_add_prop_arg:nnw #3 \__stex_args_end:}
5281 }
5282
5283 \cs_new:Nn \__stex_expr_check_b:nn {
5284 \str_case:nn #2 {
5285 b {\cs_set_eq:NN \__stex_expr_do_ab_next:nn \stex_term_omb:nn}
5286 B {\cs_set_eq:NN \__stex_expr_do_ab_next:nn \stex_term_omb:nn}
5287 }
5288 }

```

Arguments:

```

5289 \NewDocumentCommand \__stex_expr_arg:n {s O{} m} {
5290 \IfBooleanTF #1 {
5291 \stex_annotate_invisible:n{
5292 \__stex_expr_arg_inner:nn{#2}{#3}
5293 }
5294 }{
5295 \__stex_expr_arg_inner:nn{#2}{#3}
5296 }
5297 }
5298
5299 \cs_new_protected:Nn \__stex_expr_arg_inner:nn {
5300 \tl_if_empty:nTF{#1}{
5301 \int_gincr:N \l__stex_expr_arg_counter_int
5302 \exp_args:Ne \__stex_expr_check:nTF{ \int_use:N \l__stex_expr_arg_counter_int }{
5303 \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl
5304 }{
5305 \__stex_expr_arg_inner:nn{ }
5306 }{ #2 }
5307 }{
5308 \__stex_expr_check:nTF {#1}{
5309 \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5310 }{
5311 \exp_args:No \str_case:nnTF \l_tmpb_tl {

```

```

5312     {a}{
5313         \exp_args:NNne \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
5314             \l_tmpa_tl X
5315         }
5316         \tl_set:Nx \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5317     }
5318     {B}{
5319         \exp_args:NNne \prop_gput:Nnn \l__stex_expr_customs_prop {#1}{
5320             \l_tmpa_tl X
5321         }
5322         \tl_set:Ne \l_tmpa_tl { #1 \int_eval:n {\tl_count:N \l_tmpa_tl + 1} }
5323     }
5324 }{
5325     \__stex_expr_arg_do:oon \l_tmpa_tl \l_tmpb_tl { #2 }
5326 }{
5327     \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_full_tl}
5328 }
5329 }
5330 }
5331 }
5332
5333 \prg_new_conditional:Nnn \__stex_expr_check:n {TF} {
5334     \exp_args:NNe \prop_get:NnNTF \l__stex_expr_customs_prop {#1} \l_tmpa_tl {
5335         \tl_set:Ne \l_tmpb_tl {\seq_item:Nn \l__stex_expr_customs_seq {#1} }
5336         \tl_if_empty:NTF \l_tmpa_tl {
5337             \exp_args:NNe \prop_gput:Nnn \l__stex_expr_customs_prop
5338                 { #1 }{X}
5339             \exp_args:No \str_case:nnF \l_tmpb_tl {
5340                 {a}{
5341                     \tl_set:Ne \l_tmpa_tl{ #1 1 }
5342                 }
5343                 {B}{
5344                     \tl_set:Ne \l_tmpa_tl{ #1 1 }
5345                 }
5346             }{
5347                 \tl_set:Ne \l_tmpa_tl{ #1 }
5348             }
5349             \prg_return_true:
5350         }{
5351             \prg_return_false:
5352         }
5353     }{
5354         \msg_error:nnxx{stex}{error/invalidarg}{#1}{\l_stex_current_full_tl}
5355         \prg_return_false:
5356     }
5357 }
5358
5359 % #1 argnum #2 argmode #3 code
5360 \cs_new_protected:Nn \__stex_expr_arg_do:nnn {
5361     \stex_debug:nn{custom}{Doing~argument~#1~of~mode~#2:~\tl_to_str:n{#3}}
5362     \group_begin:
5363         \bool_set_true:N \l_stex_allow_semantic_bool
5364         \stex_term_arg:nnn {#2}{#1}{#3}
5365     \group_end:

```

```

5366 }
5367 \cs_generate_variant:Nn \__stex_expr_arg_do:nnn {oon}

```

Return code

```

5368 \cs_new_protected:Nn \__stex_expr_maybe_return:n {
5369   \tl_set:Nn \l__stex_expr_return_this_tl {#1}
5370   \tl_clear:N \l__stex_expr_return_args_tl
5371   \exp_args:Nne \use:n {
5372     \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs:
5373     \cs_set:Npn \l_stex_current_arity_str } {
5374       \int_step_function:nN \l_stex_current_arity_str \__stex_expr_return_arg:n
5375       \__stex_expr_return_next:
5376     }
5377   \__stex_expr_ret_cs:
5378 }
5379
5380 \cs_new:Nn \__stex_expr_return_arg:n {
5381   \tl_put_right:Nn \exp_not:N \l__stex_expr_return_args_tl {{## #1}}
5382 }
5383
5384 \cs_new_protected:Nn \__stex_expr_return_next: {
5385   \peek_charcode_remove:NTF ! {
5386     \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl \group_end:
5387   }\__stex_expr_return:
5388 }
5389
5390 \cs_new_protected:Nn \__stex_expr_return: {
5391   \tl_set:Ne \l__stex_expr_return_this_tl {
5392     \tl_if_empty:NTF \l_stex_return_notation_tl {
5393       \exp_after:wN \exp_after:wN \exp_after:wN
5394       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
5395         \exp_after:wN \l__stex_expr_return_this_tl \l__stex_expr_return_args_tl
5396       }
5397     }{
5398       \l_stex_return_notation_tl
5399     }
5400   }
5401   \stex_debug:nn{return}{Notation:~\meaning\l__stex_expr_return_this_tl}
5402
5403   \exp_after:wN
5404   \tl_put_left:Nn \exp_after:wN \l__stex_expr_return_this_tl \exp_after:wN {
5405     \exp_after:wN \group_begin: \l_stex_current_redo_tl
5406   }
5407   \exp_args:Nne \use:n {
5408     \cs_generate_from_arg_count:NNnn \__stex_expr_ret_cs:
5409     \cs_set:Npn \l_stex_current_arity_str } {
5410       \exp_args:No \exp_not:n \l_stex_current_return_tl
5411     }
5412   \stex_debug:nn{return}{
5413     \meaning\__stex_expr_ret_cs:^^J
5414     \meaning\l__stex_expr_return_this_tl^^J
5415     \exp_args:No \exp_not:n \l__stex_expr_return_args_tl^^J
5416   }

```



```

5417 \exp_args:Nne \use:nn {
5418   \exp_after:wN \group_end: \__stex_expr_ret_cs:
5419 }{
5420   \exp_args:No \exp_not:n \l__stex_expr_return_args_tl
5421   {
5422     \exp_args:No \exp_not:n \l__stex_expr_return_this_tl
5423     \group_end:
5424   }
5425 }
5426 }

```

32.4.1 Term Annotations

`\stex_eat_exclamation_point:`

```

5427 \cs_new_protected:Nn \stex_eat_exclamation_point: {
5428   \peek_charcode_remove:NT ! \stex_eat_exclamation_point:
5429 }

```

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

sfunction

We occasionally allow for semantic macros where they otherwise shouldn't be, if we are in “invisible” parts:

```

5430 \bool_new:N \stex_in_invisible_html_bool

```

`\stex_term_oms:nn`

```

5431 \stex_if_html_backend:TF {
5432   \cs_new_protected:Nn \stex_term_oms:nn {
5433     \bool_if:NTF\l_stex_in_this_bool{#2}{
5434       \__stex_expr_check_comp:
5435       \tl_if_empty:NTF \l_stex_current_term_tl {
5436         \__stex_expr_annotate:nnn{OMID}{#1}{#2}
5437       }{
5438         \__stex_expr_do_headterm:nn{#1}{#2}
5439       }
5440     }
5441   }
5442 }{
5443   \cs_new_protected:Nn \stex_term_oms:nn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_check_comp:}
5444 }
5445
5446 \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 142.)

sfunction

`\stex_term_omv:nn`

```

5447 \stex_if_html_backend:TF {
5448   \cs_new_protected:Nn \stex_term_omv:nn {
5449     \bool_if:NTF\l_stex_in_this_bool{#2}{
5450       \__stex_expr_check_comp:
5451       \tl_if_empty:NTF \l_stex_current_term_tl {
5452         \__stex_expr_annotate:nnn{OMV}{#1}{#2}
5453       }{

```

```

5454     \_stex_expr_do_headterm:nn{#1}{#2}
5455   }
5456 }
5457 }
5458 }{
5459   \cs_new_protected:Nn \_stex_term_omv:nn {\bool_if:NF\l_stex_in_this_bool\_stex_expr_che
5460 }

```

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

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:

```

5461 \stex_if_html_backend:TF {
5462   \cs_new_protected:Nn \_stex_expr_do_headterm:nn {
5463     \bool_if:NTF \stex_in_invisible_html_bool {
5464       {\bool_set_true:N \l_stex_allow_semantic_bool
5465        \ensuremath{\l_stex_current_term_tl}
5466       }
5467     }{
5468       \_stex_expr_annotate:nnn{complex}{#1}{
5469         \stex_annotate_invisible:nn{data-ftml-headterm={}}{
5470           {\bool_set_true:N \l_stex_allow_semantic_bool
5471            \ensuremath{\l_stex_current_term_tl}
5472           }
5473         }
5474         #2
5475       }
5476     }
5477   }
5478 }{
5479   \cs_new_protected:Nn \_stex_expr_do_headterm:nn { #2 }
5480 }

```

`_stex_term_oma:nn`

```

5481 \stex_if_html_backend:TF {
5482   \cs_new_protected:Nn \_stex_term_oma:nn {
5483     \bool_if:NTF \l_stex_in_this_bool{#2}{
5484       \_stex_expr_check_comp:
5485       \_stex_expr_annotate:nnn{OMA}{#1}{
5486         \tl_if_empty:NF \l_stex_current_term_tl {
5487           \stex_annotate_invisible:nn{data-ftml-headterm={}}{
5488             \bool_set_true:N \l_stex_allow_semantic_bool
5489             \l_stex_current_term_tl
5490           }}
5491         }
5492         #2
5493       }
5494     }
5495   }
5496 }{
5497   \cs_new_protected:Nn \_stex_term_oma:nn {\bool_if:NF\l_stex_in_this_bool\_stex_expr_che
5498 }

```

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

sfunction

`_stex_term_omb:nn`

```

5499 \stex_if_html_backend:TF {
5500   \cs_new_protected:Nn \_stex_term_omb:nn {
5501     \bool_if:NTF\l_stex_in_this_bool{#2}{
5502       \__stex_expr_check_comp:
5503       \__stex_expr_annotate:nnn{OMBIND}{#1}{
5504         \tl_if_empty:NF \l_stex_current_term_tl {
5505           \stex_annotate_invisible:nn{data-ftml-headterm={}}{
5506             \bool_set_true:N \l_stex_allow_semantic_bool
5507             \l_stex_current_term_tl
5508           }}
5509         }
5510       #2
5511     }
5512   }
5513 }
5514 }{
5515   \cs_new_protected:Nn \_stex_term_omb:nn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_chec
5516 }

```

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

sfunction

Does the actual annotating: **TODO: This shouldn't use `\l_stex_current_symbol_`**
`uri`!

```

5517 % kind, notation id, body
5518 \stex_if_html_backend:TF {
5519   \cs_new_protected:Nn \__stex_expr_annotate:nnn {
5520     \exp_args:Ne \stex_annotate:nn{
5521       data-ftml-term=#1,
5522       data-ftml-head={\l_stex_current_full_tl},
5523       data-ftml-notationid={#2},
5524     }{
5525       \_stex_annotate_force_break:n{#3}
5526     }
5527   }
5528 }{
5529   \cs_new_protected:Nn \__stex_expr_annotate:nnn { #3 }
5530 }
5531

```

32.4.2 Symbol Arguments

`_stex_term_arg:nnnnn` #1 : argument number
 #2 : argument mode
 #3 : precedence
 #4 : argument name (WiP)
 #5 : code / body

```

5532 \cs_new_protected:Nn \_stex_term_arg:nnnnn {
5533   \group_begin:
5534   \tl_clear:N \l_stex_current_symbol_uri
5535   \tl_clear:N \l_stex_current_full_tl
5536   \tl_clear:N \l_stex_current_display_tl
5537   \tl_clear:N \l_stex_current_term_tl

```

```

5538 \int_set:Nn \l_stex_notation_downprec { #3 }
5539 \bool_set_true:N \l_stex_allow_semantic_bool
5540 \_stex_term_arg:nnn {#2}{#1}{#5}
5541 \group_end:
5542 }

```

(End of definition for `_stex_term_arg:nnnnn`. This function is documented on page 143.)

sfunction

`_stex_term_arg:nnn`

```

5543 \cs_new_protected:Nn \__stex_expr_check_comp: {
5544   \bool_if:NT \g_stex_in_comp_bool {
5545     \msg_error:nn{stex}{error/argincomp}
5546   }
5547 }
5548 \stex_if_html_backend:TF {
5549   \cs_new_protected:Nn \_stex_term_arg:nnn {
5550     \stex_if_do_html:TF{
5551       \bool_if:NTF\l_stex_in_this_bool{#3}{
5552         \_stex_annotate_force_break:n{
5553           \stex_annotate:nn{
5554             data-ftml-arg={#2}, data-ftml-argmode={#1}
5555           }{
5556             \_stex_annotate_force_break:n{
5557               \__stex_expr_check_comp:
5558               #3
5559             }
5560           }
5561         }{ #3 }
5562       }{ #3 }
5563     }{
5564   }
5565   \cs_new_protected:Nn \_stex_term_arg:nnn {\bool_if:NF\l_stex_in_this_bool\__stex_expr_che
5566 }

```

(End of definition for `_stex_term_arg:nnn`. This function is documented on page 143.)

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:

```

5567 \cs_new:Nn \__stex_expr_do_aB_clist: {
5568   \_stex_annotate_force_break:n{
5569     \seq_use:Nn \l_stex_aB_args_seq {
5570       \mathpunct{\comp{,}}
5571     }
5572   }
5573 }
5574
5575 \cs_set_eq:NN \_stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5576
5577 \cs_new_protected:Nn \_stex_is_sequentialized:n {
5578   \group_begin: #1 \group_end:
5579 }

```

Setup:

```

5580 \int_new:N \l__stex_expr_count_int
5581 \cs_new_protected:Nn \_stex_term_arg_aB:nnnnn {
5582   \stex_debug:nn{sequences}{Processing~argument:~ \tl_to_str:n{#5}}
5583   \tl_if_empty:nTF{#5}{
5584     \_stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{
5585     }{
5586       \seq_clear:N \l_stex_aB_args_seq
5587       \int_zero:N \l__stex_expr_count_int
5588       \clist_map_inline:nn{#5}{
5589         \__stex_expr_aB_arg:nnnnn{##1}{#1}{#2}{#3}{#4}
5590       }
5591       \_stex_term_do_aB_clist:
5592     }
5593   }

```

We “pattern match” the sequence argument - is it a comma-separated list? A sequence variable macro? `\argmap?` `\seqmap?`

```

5594 % 1: code 2: argnum 3: argmode 4: precedence 5: argname
5595 \cs_new_protected:Npn \__stex_expr_aB_arg:nnnnn #1 #2 #3 {
5596   \int_incr:N \l__stex_expr_count_int
5597   \stex_debug:nn{expressions}{matching~argument~ #2 #3:~ \tl_to_str:n{#1}}
5598   \__stex_expr_is_varseq:nTF{#1}{
5599     \stex_debug:nn{expressions}{=sequence-variable}
5600     \exp_after:wN \exp_after:wN \exp_after:wN
5601     \__stex_expr_assoc_seq:nnnnnnn
5602     \exp_after:wN
5603     \__stex_expr_gobble:nnnnnnnn #1 \__stex_expr_end:
5604   }{
5605     \__stex_expr_is_seqmap:nTF{#1}{
5606       \stex_debug:nn{expressions}{=seqmap}
5607       \exp_args:NNe \use:nn \__stex_expr_do_seqmap:nnnnnn { \tl_tail:n{#1} }
5608     }{
5609       \stex_debug:nn{expressions}{=simple}
5610       \__stex_expr_aB_simple_arg:nnnnn{#1}
5611     }
5612   }
5613   {#2}{#3}
5614 }
5615
5616 \cs_new:Npn \__stex_expr_gobble:nnnnnnnn #1 #2 #3 #4 #5 #6 #7 #8 #9 \__stex_expr_end: {
5617   {#2} #3 {#6}
5618 }

```

Is it a varseq?

```

5619 \prg_new_conditional:Nnn \__stex_expr_is_varseq:n {TF} {
5620   \int_compare:nNnTF {\tl_count:n{#1}} = 1 {
5621     \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_head:n{#1}}
5622     \_stex_invoke_variable:nnnnnnN {
5623       \exp_args:Ne \cs_if_eq:NNTF {\exp_args:No \tl_item:nn{#1}{8}}
5624       \stex_invoke_sequence:
5625       \prg_return_true:\prg_return_false:
5626     }\prg_return_false:
5627   }\prg_return_false:
5628 }

```

Is it a seqmap?

```

5629 \prg_new_conditional:Nnn \__stex_expr_is_seqmap:n {TF} {
5630   \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
5631     \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\seqmap}
5632     \prg_return_true:\prg_return_false:
5633   }\prg_return_false:
5634 }

```

Is it an argsep?

```

5635 \prg_new_conditional:Nnn \__stex_expr_is_argsep:n {TF} {
5636   \int_compare:nNnTF {\tl_count:n{#1}} = 3 {
5637     \exp_args:Ne \tl_if_eq:nnTF {\tl_head:n{#1}} {\argsep}
5638     \prg_return_true:\prg_return_false:
5639   }\prg_return_false:
5640 }

```

Simple argument:

```

5641 \cs_new_protected:Nn \__stex_expr_aB_simple_arg:nnnnn{
5642   \seq_put_right:Ne \l_stex_aB_args_seq {
5643     \stex_term_arg:nnnnn{#2}\int_use:N\l__stex_expr_count_int}{#3}{#4}{#5}{
5644       \exp_not:n{
5645         \tl_set_eq:NN \stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5646         #1
5647       }
5648     }
5649   }
5650 }

```

Sequence variable:

```

5651 % 1: name 2: arity 3: clist 4: argnum 5: argmode 6: precedence 7: argname
5652 \cs_new_protected:Nn \__stex_expr_assoc_seq:nnnnnnn {
5653   \group_begin:
5654     \seq_clear:N \l_stex_aB_args_seq
5655     \tl_set:Nn \l_stex_current_full_tl{#1}
5656     \tl_set:Nn \l_stex_current_display_tl{#1}
5657     \__stex_expr_assoc_make_seq:nnn{#1}{#3}{#2}
5658   \exp_args:NNe \use:nn \group_end: {
5659     \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5660       \stex_is_sequentialized:n{
5661         \stex_term_arg:nnnnn{#4}\int_use:N\l__stex_expr_count_int}{#5}{#6}{#7}{
5662           \bool_set_true:N \l_stex_allow_semantic_bool
5663           \tl_if_empty:NF \l_stex_current_term_tl {
5664             \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5665               \exp_args:No \exp_not:n \l_stex_current_term_tl
5666             }
5667           }
5668         \stex_annotate:nn{
5669           data-ftml-term=OMV,
5670           data-ftml-head={#1},
5671           data-ftml-notationid={}
5672         }{
5673           \stex_annotate_force_break:n{
5674             \stex_term_do_aB_clist:
5675           }
5676         }

```

```

5677     }
5678   }
5679 }
5680 }
5681 }
5682
5683 % #1: name, #2: clist, #3:arity
5684 \cs_new_protected:Nn \__stex_expr_assoc_make_seq:nnn {
5685   \stex_use_notation:nnTF {#1}{-}{
5686     \cs_set_eq:NN \l__stex_expr_cs \l_stex_notation_cs
5687   }{
5688     \stex_do_default_notation:
5689     \cs_set_eq:NN \l__stex_expr_cs \l_stex_default_notation
5690   }
5691   \clist_map_inline:nn{#2}{
5692     \tl_if_eq:nnTF{##1}{\ellipses}{
5693       \seq_put_right:Nn \l_stex_aB_args_seq { ##1 }
5694     }{
5695       \int_compare:nNnTF {#3} = 1 {
5696         \tl_set:Nn \l__stex_expr_iarg_tl { {##1} }
5697       }{
5698         \tl_set:Nn \l__stex_expr_iarg_tl { ##1 }
5699       }
5700       \seq_put_right:Ne \l_stex_aB_args_seq {
5701         \group_begin:
5702         \exp_not:n {
5703           \tl_set_eq:NN \stex_term_do_aB_clist: \__stex_expr_do_aB_clist:
5704           \def\comp{\_varcomp}
5705           \tl_set:Nn \l_stex_current_full_tl{#1}
5706         }
5707         \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
5708         \exp_after:wN \exp_after:wN \exp_after:wN {
5709           \exp_after:wN \l__stex_expr_cs \exp_after:wN \group_end: \l__stex_expr_iarg_tl
5710         }
5711       }
5712     }
5713   }
5714 }

```

seqmap:

```

5715 % 1: fun 2: clist 3: argnum 4: argmode 5: precedence 6: argname
5716 \cs_new_protected:Nn \__stex_expr_do_seqmap:nnnnnn {
5717   \group_begin:
5718   \cs_set:Npn \l_tmpa_cs ##1 {#1}
5719   \seq_clear:N \l_stex_aB_args_seq
5720   \clist_map_inline:nn{#2}{
5721     \__stex_expr_is_varseq:nTF{##1}{
5722       \exp_after:wN
5723       \__stex_expr_varseq_in_map:nnnnnnnn ##1
5724     }{
5725       \seq_put_right:Nn \l_stex_aB_args_seq {##1}
5726     }
5727   }
5728   \seq_clear:N \l__stex_expr_old_seq
5729   \seq_map_inline:Nn \l_stex_aB_args_seq {

```

```

5730 \tl_if_eq:nnTF{##1}{\ellipses}{
5731 \seq_put_right:Nn \l__stex_expr_old_seq {##1}
5732 }
5733 % TODO \stex_is_sequentialized:n
5734 {
5735 \exp_args:NNo \seq_put_right:Nn \l__stex_expr_old_seq {
5736 \l_tmpa_cs {##1}
5737 }
5738 }
5739 }
5740 \seq_set_eq:NN \l_stex_aB_args_seq \l__stex_expr_old_seq
5741 \exp_args:NNe \use:nn \group_end: {
5742 \seq_put_right:Nn \exp_not:N \l_stex_aB_args_seq {
5743 \stex_is_sequentialized:n{
5744 \stex_term_arg:nnnnn{#3\int_use:N\l__stex_expr_count_int}{#4}{#5}{#6}{
5745 \bool_set_true:N \l_stex_allow_semantic_bool
5746 \tl_set:Nn \exp_not:N \l_stex_current_full_tl
5747 {\l_stex_current_full_tl}
5748 \tl_set:Nn \exp_not:N \l_stex_current_term_tl {
5749 \symuse{Metatheory?sequence~map}
5750 {\exp_args:No \exp_not:n \l_tmpa_tl}
5751 {\stex_term_do_aB_clist:}
5752 }
5753 \__stex_expr_do_headterm:nn{}{
5754 \stex_term_do_aB_clist:
5755 }
5756 }
5757 }
5758 }
5759 }
5760 }
5761
5762 \cs_new_protected:Nn \__stex_expr_varseq_in_map:nnnnnnnn {
5763 \__stex_expr_assoc_make_seq:nnn{#2}{#6}{#3}
5764 }

```

(End of definition for `\stex_term_arg:aB:nnnnn`. This function is documented on page 143.)

sfunction

32.4.3 Notation components

```

5765 <@@=stex_comps>

\comp
\compemph@uri
\compemph
5766 \cs_set_protected:Npn \comp {}
5767
5768 \cs_new_protected:Npn \compemph@uri #1 #2 {
5769 \compemph{ #1 }
5770 }
5771
5772 \cs_new_protected:Npn \compemph #1 { #1 }
5773
5774 \cs_new_protected:Npn \_comp {
5775 \_do_comp:nnNn {comp}{ }\compemph@uri

```



```

5776 }
5777
5778 \cs_new_protected:Npn \_thiscomp #1 #2 {
5779   {\tl_set:cn{this}{\{}}#1{#2}\c_math_subscript_token{
5780     \group_begin:
5781     \bool_set_true:N \l_stex_allow_semantic_bool
5782     \bool_set_true:N \l_stex_in_this_bool
5783     \l_stex_struct_this_tl
5784     \group_end:
5785   }
5786 }
5787 }
5788
5789 \def\maincomp{\comp}

```

(End of definition for \comp, \compemph@uri, and \compemph. These functions are documented on page 143.)

sfunction

\varemp@uri
\varemp

```

5790 \cs_new_protected:Npn \varemp@uri #1 #2 {
5791   \varemp{#1}
5792 }
5793
5794 \cs_new_protected:Npn \varemp #1 { #1 }
5795
5796 \cs_new_protected:Npn \_varcomp {
5797   \_do_comp:nnNn {comp}{\}\varemp@uri
5798 }

```

(End of definition for \varemp@uri and \varemp. These functions are documented on page 143.)

sfunction

\defemph@uri
\defemph

```

5799 \cs_new_protected:Npn \defemph@uri #1 #2 {
5800   \defemph{#1}
5801 }
5802
5803 \cs_new_protected:Npn \defemph #1 {
5804   \ifmmode\else\expandafter\textbf\fi{#1}
5805 }
5806
5807 \cs_new_protected:Npn \_defcomp {
5808   \_do_comp:nnNn {defcomp}{\}\defemph@uri
5809 }

```

(End of definition for \defemph@uri and \defemph. These functions are documented on page 143.)

sfunction

\symrefemph@uri
\symrefemph

```

5810 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
5811   \symrefemph{#1}
5812 }
5813
5814 \cs_new_protected:Npn \symrefemph #1 { \emph{#1} }

```

(End of definition for `\symrefemph@uri` and `\symrefemph`. These functions are documented on page 143.)

sfunction

The following commands do the actual attributes:

`_do_comp:nnNn`

```

5815 \bool_new:N \g_stex_in_comp_bool
5816
5817 \cs_new_protected:Nn \_do_comp:nnNn {
5818   \stex_pseudogroup_with:nn{\comp}{
5819     \def\comp##1{##1}
5820     \bool_set_true:N \g_stex_in_comp_bool
5821     \stex_if_html_backend:TF {
5822       \stex_annotate:nn { data-ftml-#1 = {#2}}{ #4 }
5823     }{
5824       \tl_if_empty:NTF \l_stex_current_full_tl {
5825         #4
5826       }{
5827         #3 { #4 } \l_stex_current_full_tl
5828
5829       }
5830     }
5831     \bool_set_false:N \g_stex_in_comp_bool
5832   }
5833 }
```

(End of definition for `_do_comp:nnNn`. This function is documented on page 143.)

sfunction

32.4.4 Symbol References

Keys:

```

5834 \stex_keys_define:nnnn{symname}{
5835   \tl_clear:N \l_stex_key_pre_tl
5836   \tl_clear:N \l_stex_key_post_tl
5837   %\tl_clear:N \l_stex_key_root_tl
5838 }{
5839   pre .tl_set:N = \l_stex_key_pre_tl ,
5840   post .tl_set:N = \l_stex_key_post_tl ,
5841   %root .code:n = {}%.tl_set:N = \l_stex_key_root_tl
5842 }{}
5843
5844 \stex_keys_define:nnnn{symref}{
5845   %\tl_clear:N \l_stex_key_root_tl
5846 }{
5847   root .code:n = {}%.tl_set:N = \l_stex_key_root_tl
5848 }{}

```

`\symref`

`\sr`

```

5849 \NewDocumentCommand \symref { 0{ } m m } {
5850   \group_begin:
5851   \stex_keys_set:nn{symname}{#1}
5852   \stex_get_symbol:n{#2}

```

```

5853   \_stex_comps_do_ref:nNn{#3}\symrefemph@uri\_stex_term_oms:nn
5854 }
5855 \let\sr\symref

```

(End of definition for \symref and \sr. These functions are documented on page 144.)

sfunction

\symname

\sn

\sns

\Symname

\Sn\Sns

```

5856 \cs_new:Nn \_stex_comps_split_slash:n {
5857   \_stex_comps_split_slash_i:nw #1//
5858 }
5859
5860 \cs_new:Npn \_stex_comps_split_slash_i:nw #1/#2/ {
5861   \tl_if_empty:nTF{#2}{#1}{
5862     \_stex_comps_split_slash_i:nw #2/
5863   }
5864 }
5865
5866 \NewDocumentCommand \symname { 0{} m} {
5867   \group_begin:
5868   \stex_keys_set:nn{symname}{#1}
5869   \stex_get_symbol:n{#2}
5870   \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }
5871   \tl_set:Ne \l_stex_current_display_tl {
5872     \exp_args:Ne \_stex_comps_split_slash:n {\stex_symbol_uri_name:N \l_stex_get_symbol_uri
5873   }
5874   \_stex_comps_do_ref:nNn{
5875     \l_stex_key_pre_tl \l_stex_current_display_tl \l_stex_key_post_tl
5876   }\symrefemph@uri\_stex_term_oms:nn
5877 }
5878 \let\sn\symname
5879 \protected\def\sns{\symname[post=s]}
5880
5881 \cs_new:Nn \_stex_comps_uppercase:n { \uppercase{#1}}
5882
5883 \NewDocumentCommand \Symname { 0{} m} {
5884   \group_begin:
5885   \stex_keys_set:nn{symname}{#1}
5886   \stex_get_symbol:n{#2}
5887   \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }
5888   \tl_set:Ne \l_stex_current_display_tl {
5889     \exp_args:Ne \_stex_comps_split_slash:n {\stex_symbol_uri_name:N \l_stex_get_symbol_uri
5890   }
5891   \_stex_comps_do_ref:nNn{
5892     \l_stex_key_pre_tl \exp_args:NNe \use:nn \_stex_comps_uppercase:n \l_stex_current_disp
5893   }\symrefemph@uri\_stex_term_oms:nn
5894 }
5895 \let\Sn\Symname
5896 \protected\def\Sns{\Symname[post=s]}

```

(End of definition for \symname and others. These functions are documented on page 144.)

sfunction

\varref

\varname

\Varname

```

5897 \NewDocumentCommand \varref { 0{} m m } {
5898   \group_begin:
5899   \stex_keys_set:nn{symname}{#1}
5900   \stex_get_var:n{#2}
5901   \__stex_comps_do_ref:nNn{#3}\vareph@uri{
5902     \tl_set:Nn \l_stex_current_full_tl { \l_stex_get_variable_str }
5903     \tl_set:Nn \l_stex_current_display_tl {\l_stex_get_variable_str}
5904     \def\comp{\_varcomp}
5905     \_stex_term_omv:nn
5906   }
5907 }
5908
5909 \NewDocumentCommand \varname { 0{} m } {
5910   \group_begin:
5911   \stex_keys_set:nn{symname}{#1}
5912   \stex_get_var:n{#2}
5913   \__stex_comps_do_ref:nNn{
5914     \l_stex_key_pre_tl\l_stex_get_variable_str\l_stex_key_post_tl
5915   }\vareph@uri{
5916     \tl_set:Nn \l_stex_current_full_tl { \l_stex_get_variable_str }
5917     \tl_set:Nn \l_stex_current_display_tl {\l_stex_get_variable_str}
5918     \def\comp{\_varcomp}
5919     \_stex_term_omv:nn
5920   }
5921 }
5922
5923 \NewDocumentCommand \Varname { 0{} m } {
5924   \group_begin:
5925   \stex_keys_set:nn{symname}{#1}
5926   \stex_get_var:n{#2}
5927   \__stex_comps_do_ref:nNn{
5928     \l_stex_key_pre_tl\exp_after:wN\__stex_comps_uppercase:n\l_stex_get_variable_str\l_stex
5929   }\vareph@uri{
5930     \tl_set:Nn \l_stex_current_full_tl { \l_stex_get_variable_str }
5931     \tl_set:Nn \l_stex_current_display_tl {\l_stex_get_variable_str}
5932     \def\comp{\_varcomp}
5933     \_stex_term_omv:nn
5934   }
5935 }

```

(End of definition for `\varref`, `\varname`, and `\Varname`. These functions are documented on page 144.)

sfunction

```

\definiendum
  \define
  \Defname
\defnotation
5936 \stex_keys_define:nnnn{defname}{-}{-}
5937   gf      .code:n      = {},
5938   root    .code:n      = {}
5939 }{symname}
5940
5941 \stex_keys_define:nnnn{definiendum}{-}{-}
5942   gf      .code:n      = {},
5943   root    .code:n      = {}
5944 }{-}
5945

```

```

5946 \NewDocumentCommand \definiendum { O{} m m } {
5947   \stex_keys_set:nn{definiendum}{ #1 }
5948   \__stex_comps_do_defref:nn{#2}{#3}
5949 }
5950 \stex_deactivate_macro:Nn \definiendum {definition~environments}
5951
5952 \NewDocumentCommand \definame { O{} m } {
5953   \stex_keys_set:nn{definame}{#1}
5954   \__stex_comps_do_defref:nn{#2}{
5955     \l_stex_key_pre_tl\l_stex_current_display_tl\l_stex_key_post_tl
5956   }
5957 }
5958 \protected\def\definames{\definame[post=s]}
5959 \stex_deactivate_macro:Nn \definame {definition~environments}
5960
5961
5962 \NewDocumentCommand \Definame { O{} m } {
5963   \stex_keys_set:nn{definame}{#1}
5964   \__stex_comps_do_defref:nn{#2}{
5965     \l_stex_key_pre_tl \exp_args:NNe \use:nn \__stex_comps_uppercase:n \l_stex_current_disp
5966   }
5967 }
5968 \protected\def\Definames{\Definame[post=s]}
5969 \stex_deactivate_macro:Nn \Definame {definition~environments}
5970
5971
5972 \NewDocumentCommand \defnotation{ m } {
5973   \_stex_next_symbol:n { \def\comp{\_defcomp}}#1
5974 }
5975 \stex_deactivate_macro:Nn \defnotation {definition~environments}

```

(End of definition for \definiendum and others. These functions are documented on page 144.)

sfunction

Does the actual referencing:

```

5976 \cs_new_protected:Nn \__stex_comps_do_ref:nNn {
5977   \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
5978   \bool_if:NTF \l_stex_allow_semantic_bool{
5979     \tl_set_eq:NN \l_stex_current_symbol_uri \l_stex_get_symbol_uri
5980     \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_current_symbol_uri
5981       %\str_set:Ne \l_stex_current_symbol_name_str { \stex_symbol_uri_name:N \l_stex_current_
5982       %\str_if_in:NnT \l_stex_current_symbol_name_str / {
5983       % \str_set:Ne \l_stex_current_symbol_name_str {
5984       % \exp_after:wN \__stex_comps_slash:w \l_stex_current_symbol_name_str
5985       % /\_stex_args_end:
5986       % }
5987       %}
5988       \tl_clear:N \l_stex_current_term_tl
5989       \def\comp{\_comp}
5990       \let\compemph@uri#2
5991       #3{\_comp{#1}}
5992   }{
5993     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_full_tl}
5994   }
5995   \group_end:

```

```

5996 }
\definame et al:
5997 \cs_new_protected:Nn \__stex_comps_do_defref:nn {
5998   \stex_if_html_backend:T{\relax\ifvmode\indent\fi}
5999   \group_begin:
6000   \stex_get_symbol:n{#1}
6001   \bool_if:NTF \l_stex_allow_semantic_bool{
6002     \tl_set:Nn \l_stex_current_full_tl { \stex_use_symbol_uri:N \l_stex_get_symbol_uri }
6003     \tl_set:Nn \l_stex_current_display_tl {
6004       \stex_symbol_uri_name:N \l_stex_get_symbol_uri
6005     }
6006     %\str_if_in:NnT \l_stex_get_svariable_str / {
6007     % \str_set:Ne \l_stex_get_variable_str {
6008     %   \exp_after:wN \__stex_comps_slash:w \l_stex_get_variable_str
6009     %   /\_stex_args_end:
6010     % }
6011     %}
6012     \exp_args:Ne \stex_ref_new_sym_target:n \l_stex_current_full_tl
6013     \do_comp:nnNn {definiendum}{\l_stex_current_full_tl}\defemph@uri{#2}
6014   }{
6015     \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_full_tl}
6016   }
6017   \group_end:
6018 }
6019
6020 \cs_new:Npn \__stex_comps_slash:w #1/#2/#3\_stex_args_end: {
6021   \tl_if_empty:nTF{#3}{
6022     #2
6023   }{
6024     \__stex_comps_slash:w #2 / #3 \_stex_args_end:
6025   }
6026 }

```

32.5 Checking

```

\stex_if_check_terms_p:
\stex_if_check_terms:TF
6027 <@@=stex_check>
6028 \stex_if_html_backend:TF {
6029   \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
6030     \prg_return_false:
6031   }
6032 }{
6033   \stex_get_env:Nn\__stex_check_env_str{STEX_CHECKTERMS}
6034   \str_if_empty:NF\__stex_check_env_str{
6035     \exp_args:No \str_if_eq:nnF \__stex_check_env_str{false}{
6036       \bool_set_true:N \c_stex_check_terms_bool
6037     }
6038   }
6039   \bool_if:NTF \c_stex_check_terms_bool {
6040     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
6041       \prg_return_true:
6042     }

```

```

6043   }{
6044     \prg_new_conditional:Nnn \stex_if_check_terms: {p, T, F, TF} {
6045       \prg_return_false:
6046     }
6047   }
6048 }

```

(End of definition for `\stex_if_check_terms:TF`. This function is documented on page 144.)

sfunction

`\stex_check_term:nn`

```

6049 \stex_if_check_terms:TF{
6050   \cs_new_protected:Nn \stex_check_term:nn {
6051     \marginpar {\tiny Checking~#1:~
6052       \group_begin:
6053         $#2$
6054       \group_end:
6055     }
6056   }
6057 }{
6058   \cs_new_protected:Nn \stex_check_term:nn {}
6059 }

```

(End of definition for `\stex_check_term:nn`. This function is documented on page 144.)

sfunction

`_stex_check_terms:`

```

6060 \stex_if_check_terms:TF{
6061   \cs_new_protected:Nn \_stex_check_terms: {
6062     \stex_debug:nn{check_terms}{Checking~type...}
6063     \tl_if_empty:NF \l_stex_key_type_tl {
6064       \stex_check_term:nn{type}\l_stex_key_type_tl
6065     }
6066     \stex_debug:nn{check_terms}{Checking~definiens...}
6067     \tl_if_empty:NF \l_stex_key_def_tl {
6068       \stex_check_term:nn{definiens}\l_stex_key_def_tl
6069     }
6070     \stex_debug:nn{check_terms}{Checking~return...}
6071     \tl_if_empty:NF \l_stex_key_return_tl {
6072       \stex_check_term:nn{return}\l_stex_key_return_tl!}
6073     }
6074     \stex_debug:nn{check_terms}{Checking~argument~types...}
6075     \group_begin:\l_stex_key_argtypes_clist\group_end:
6076     \tl_if_empty:NF \l_stex_key_argtypes_clist {
6077       \stex_check_term:nn{argument~types}\l_stex_key_argtypes_clist}
6078     }
6079   }
6080 }{
6081   \cs_new_protected:Nn \_stex_check_terms: {}
6082 }
6083

```

(End of definition for `_stex_check_terms:.` This function is documented on page 144.)

sfunction

Chapter 33

Notations

```
6084 <@@=stex_notations>

      keys:
6085 \stex_keys_define:nnnn{notation}{
6086   \str_clear:N \l_stex_key_variant_str
6087   \str_clear:N \l_stex_key_prec_str
6088   \str_clear:N \l_stex_key_op_tl
6089   \str_clear:N \l_stex_key_intent_str
6090   \clist_clear:N \l_stex_key_intent_args_clist
6091 }{
6092   variant      .str_set_x:N = \l_stex_key_variant_str ,
6093   prec         .str_set_x:N = \l_stex_key_prec_str ,
6094   op           .tl_set:N     = \l_stex_key_op_tl ,
6095   intent       .str_set:N     = \l_stex_key_intent_str ,
6096   argnames     .clist_set:N   = \l_stex_key_intent_args_clist ,
6097   unknown      .code:n       = {
6098     \str_if_empty:NTF \l_keys_key_str {
6099       \str_set:Nc \l_stex_key_variant_str {\l_keys_key_tl}
6100     }{
6101       \str_set_eq:NN \l_stex_key_variant_str \l_keys_key_str
6102     }
6103   }
6104 }{style}
6105
6106 \stex_keys_define:nnnn{symdef}{ }{ }{decl,notation}
6107
6108 \stex_keys_define:nnnn{vardef}{
6109   \bool_set_false:N \l__stex_notations_bind_bool
6110 }{
6111   bind .bool_set:N = \l__stex_notations_bind_bool
6112 }{symdef}
6113
\notation

6114 \stex_new_stylable_cmd:nnnn {notation} { s m O{ } m } {
6115   \stex_keys_set:nn{notation}{#3}
6116   \stex_get_symbol:n{#2}
6117   \stex_notation_parse:n{#4}
6118   \stex_if_check_terms:T{ \_stex_notation_check: }
```



```

6119 \_stex_notation_add:
6120 \stex_if_do_html:T {
6121   \def\comp{\_comp}
6122   \_stex_notation_do_html:n{\stex_use_symbol_uri:N \l_stex_get_symbol_uri}
6123 }
6124 \IfBooleanTF#1{
6125   \_stex_notation_set_default:n{
6126     \l_stex_get_symbol_uri
6127   }
6128 }{}
6129 \stex_if_smsmode:F{
6130   \group_begin:
6131   \__stex_notations_styledefs:
6132   \stex_style_apply:
6133   \group_end:
6134 }
6135 \stex_smsmode_do:
6136 }{}
6137
6138 \stex_deactivate_macro:Nn \notation {module~environments}
6139 \stex_every_module:n {\stex_reactivate_macro:N \notation}
6140 \stex_sms_allow_escape:N \notation
6141
6142 \cs_new_protected:Nn \__stex_notations_styledefs: {
6143   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
6144   \str_set:Nn \thisdeclname {\stex_symbol_uri_name:N \l_stex_get_symbol_uri}
6145   \tl_set:Ne \thisdecluri {\stex_use_symbol_uri:N \l_stex_get_symbol_uri}
6146   \def\thisnotation{
6147     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{ }{
6148       \_stex_notation_make_args:
6149     }
6150   }
6151 }

```

(End of definition for \notation. This function is documented on page 145.)

sfunction

\varnotation

```

6152 \stex_new_stylable_cmd:nnnn {\varnotation} { s m O{} m} {
6153   \stex_keys_set:nn{notation}{#3}
6154   \stex_get_var:n{#2}
6155   \str_set_eq:NN \l_stex_key_name_str \l_stex_get_variable_str
6156   \stex_notation_parse:n{#4}
6157   \stex_if_check_terms:T{ \_stex_notation_check: }
6158   \_stex_var_notation_macro:
6159   \IfBooleanTF#1{
6160     \_stex_notation_set_default:n{\l_stex_get_variable_str}
6161   }{}
6162   \group_begin:
6163   \tl_set_eq:NN \thisvarname \l_stex_get_variable_str
6164   \tl_clear:N \thisstyle
6165   \str_set_eq:NN\thisnotationvariant\l_stex_key_variant_str
6166   \def\thisnotation{
6167     \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{ }{

```

```

6168     \_stex_notation_make_args:
6169   }
6170 }
6171 \stex_style_apply:
6172 \group_end:
6173 }{}

```

(End of definition for \varnotation. This function is documented on page 145.)

sfunction

\stex_notation_parse:n

```

6174 \cs_new_protected:Nn \stex_notation_parse:n {
6175   \tl_if_empty:NF \l_stex_key_op_tl {
6176     \tl_set:Ne \l_stex_key_op_tl { \exp_not:N\maincomp {
6177       \exp_args:No \exp_not:n \l_stex_key_op_tl
6178     } }
6179   }
6180   \seq_clear:N \l__stex_notations_precs_seq
6181   \tl_clear:N \l_stex_notation_args_tl
6182   \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0 {
6183     \__stex_notations_const_precs:
6184     \tl_if_empty:NT \l_stex_key_op_tl {
6185       \tl_set:Nn \l_stex_key_op_tl { \maincomp{#1} }
6186     }
6187   }{
6188     \__stex_notations_fun_precs:
6189     \__stex_notations_do_missing_args:n{#1}
6190     \tl_if_empty:NT \l_stex_key_op_tl {
6191       \hbox_set:Nn \l_tmpa_box {
6192         \tl_clear:N \l_stex_current_full_tl
6193         \tl_clear:N \l_stex_current_display_tl
6194         \cs_set:Npn \l_tmpa_cs ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9 { #1 }
6195         \cs_set:Npn \maincomp ##1 {
6196           \tl_gset:Nn \l_stex_key_op_tl { \maincomp{##1} }
6197           ##1
6198         }
6199         \cs_set:Npn \argsep ##1 ##2 {##1 ##2}
6200         \cs_set:Npn \argmap ##1 ##2 ##3 {##1 ##3}
6201         \cs_set:Npn \argarraymap ##1 ##2 ##3 ##4 {
6202           ##1 ##2
6203         }
6204         \stex_suppress_html:n{$\l_tmpa_cs abcdefghj$}
6205       }
6206     }
6207   }
6208   \exp_args:NNe
6209   \tl_set:Nn \l_stex_notation_macrocode_cs {
6210     \STEXInternalNotation
6211     { \l_stex_key_variant_str }
6212     { \l__stex_notations_opprec_tl }
6213     { \l_stex_key_intent_str }
6214     { \l_stex_notation_args_tl }
6215     {
6216       \int_compare:nNnTF \l_stex_get_symbol_arity_int = 0

```

```

6217         { \exp_not:n { \maincomp{ #1 } } }
6218         { \exp_not:n { #1 } \l__stex_notations_missing_tl }
6219     }
6220 }
6221 \stex_debug:nn{notation}{Notation:~\meaning\l_stex_notation_macrocode_cs}
6222 }

precedences:
6223 \cs_new_protected:Nn \__stex_notations_const_precs: {
6224   \str_if_empty:NTF \l_stex_key_prec_str {
6225     \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
6226   }{
6227     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
6228       \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
6229     }{
6230       \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
6231     }
6232   }
6233 }
6234
6235 \cs_new_protected:Nn \__stex_notations_fun_precs: {
6236   \str_if_empty:NTF \l_stex_key_prec_str {
6237     \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
6238   }{
6239     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
6240       \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
6241     }{
6242       \tl_set_eq:NN \l__stex_notations_opprec_tl \l_stex_key_prec_str
6243     }
6244   }
6245   \str_if_empty:NTF \l_stex_key_prec_str {
6246     \tl_set:Nn \l__stex_notations_opprec_tl { 0 }
6247     \int_step_inline:nn \l_stex_get_symbol_arity_int {
6248       \seq_put_right:Nn \l__stex_notations_precs_seq {0}
6249     }
6250   }{
6251     \str_if_eq:onTF \l_stex_key_prec_str {nobrackets}{
6252       \stex_debug:nn{notation}{No~brackets}
6253       \tl_set:No \l__stex_notations_opprec_tl { \neginfprec }
6254       \int_step_inline:nn \l_stex_get_symbol_arity_int {
6255         \exp_args:NNo \seq_put_right:Nn \l__stex_notations_precs_seq \infprec
6256       }
6257     } \__stex_notations_parse_precs:
6258   }
6259   \__stex_notations_do_argnames:
6260 }
6261
6262 \cs_new_protected:Nn \__stex_notations_parse_precs: {
6263   \stex_debug:nn{notation}{parsing~precedence~\l_stex_key_prec_str}
6264   \seq_set_split:NnV \l__stex_notations_seq ; \l_stex_key_prec_str
6265   \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
6266     \tl_set_eq:NN \l__stex_notations_opprec_tl \l__stex_notations_str
6267     \seq_pop_left:NNT \l__stex_notations_seq \l__stex_notations_str {
6268       \exp_args:NNo \seq_set_split:NnV \l__stex_notations_seq
6269         {\tl_to_str:n{x}} \l__stex_notations_str

```

```

6270     }
6271   }{
6272     \tl_set:No \l__stex_notations_opprec_tl { 0 }
6273   }
6274   \int_step_inline:nn \l_stex_get_symbol_arity_int {
6275     \seq_pop_left:NNTF \l__stex_notations_seq \l__stex_notations_str {
6276       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_str
6277     }{
6278       \seq_put_right:No \l__stex_notations_precs_seq \l__stex_notations_opprec_tl
6279     }
6280   }
6281 }

```

Experimental; named arguments:

```

6282 \cs_new_protected:Nn \__stex_notations_do_argnames: {
6283   \tl_clear:N \l_stex_notation_args_tl
6284   \stex_map_args:N \__stex_notations_do_argname:nn
6285 }
6286
6287 \cs_new_protected:Nn \__stex_notations_do_argname:nn {
6288   \clist_if_empty:NNTF \l_stex_key_intent_args_clist {
6289     \tl_put_right:Ne \l_stex_notation_args_tl {
6290       #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}{
6291         \str_if_empty:NF \l_stex_key_intent_str {#1}
6292       }
6293     }
6294   }{
6295     \tl_put_right:Ne \l_stex_notation_args_tl {
6296       #1#2{\seq_item:Nn \l__stex_notations_precs_seq #1}
6297       {\c_dollar_str\clist_item:Nn \l_stex_key_intent_args_clist 1}
6298     }
6299     \clist_pop:NN \l_stex_key_intent_args_clist \l_tmpa_tl
6300   }
6301 }

```

inserts hidden arguments that don't occur in the body of the notation:

```

6302 \cs_new:Nn \__stex_notations_do_missing_args:n {
6303   \str_set:Nn \l__stex_notations_missing_str {#1}
6304   \exp_args:NNe \seq_set_split:NnV \l_tmpa_seq {\c_hash_str\c_hash_str\c_hash_str\c_hash_str}
6305   \str_clear:N \l__stex_notations_missing_str
6306   \seq_map_inline:Nn \l_tmpa_seq {
6307     \tl_put_right:Nn \l__stex_notations_missing_str { ##1 }
6308   }
6309   \tl_clear:N \l__stex_notations_missing_tl
6310   \stex_map_args:N \__stex_notations_add_missing_args:nn
6311 }
6312
6313 \cs_new:Nn \__stex_notations_add_missing_args:nn {
6314   % TODO this skips arguments if e.g. ##1 occurs rather than #1!!
6315   \exp_args:NNe \str_if_in:NnF \l__stex_notations_missing_str {\c_hash_str\c_hash_str#1}{
6316     \tl_put_right:Nn \l__stex_notations_missing_tl{\STEXinvisible{## #1}}
6317   }
6318 }

```

(End of definition for `\stex_notation_parse:n`. This function is documented on page 145.)

sfunction

_stex_notation_add:

```

6319 \cs_new_protected:Nn \_stex_notation_add: {
6320   \stex_add_notation:ooeoo
6321   {\l_stex_get_symbol_uri}
6322   \l_stex_key_variant_str
6323   {\int_use:N \l_stex_get_symbol_arity_int}
6324   \l_stex_notation_macrocode_cs
6325   \l_stex_key_op_tl
6326 }

```

(End of definition for _stex_notation_add:.. This function is documented on page 145.)

sfunction

\stex_add_notation:nnnnn

\stex_add_notation:ooeoo

```

6327 \cs_new:Nn \__stex_notations_macro:nn {
6328   l_stex_notation_ #1?#2 _cs
6329 }
6330 \cs_new:Nn \__stex_notations_op_macro:nn {
6331   l_stex_notation_ #1?#2 _op_cs
6332 }
6333
6334 \cs_new_protected:Nn \stex_add_notation:nnnnn {
6335   \exp_args:Nne \stex_debug:nn{notations}{Adding~notation:^^J
6336     #1~\c_hash_str#2~#3^^J
6337     \tl_to_str:n{#4}^^J\tl_to_str:n{#5}^^J
6338     to~\stex_use_module_uri:N \l_stex_current_module_uri
6339   }
6340   \prop_gput:cen{ \_stex_notations_macro:N \l_stex_current_module_uri }
6341   {\stex_use_symbol_uri:n{#1}?#2}{\{#1\}{#2\}{#3\}{#4\}{#5\}}
6342   \stex_do_up_to_module:n {
6343     \__stex_notations_set_macro:nnnnn{#1}{#2}{#3}{#4}{#5}
6344     %\__stex_notations_activate_not:nn{#1}{#2}
6345   }
6346 }
6347 \cs_generate_variant:Nn \stex_add_notation:nnnnn {ooeoo}
6348
6349 \cs_new_protected:Nn \_stex_activate_notations: {
6350   \stex_debug:nn{activating}{All~notations:~\cs_meaning:c{ \_stex_notations_macro:N \l_stex
6351   \prop_map_inline:cn{ \_stex_notations_macro:N \l_stex_current_module_uri }{
6352     \__stex_notations_set_macro:nnnnn ##2
6353   }
6354 }
6355
6356 \cs_new_protected:Nn \__stex_notations_activate_not:nn {
6357   \bool_set_false:N \l_tmpa_bool
6358   \stex_debug:nn{notations}{activating~\stex_use_symbol_uri:n{#1}?#2}
6359   \prop_map_inline:cn{ \_stex_notations_macro:N \l_stex_current_module_uri }{
6360     %\stex_debug:nn{notations}{##1?}
6361     \exp_args:Ne \str_if_eq:nnT{\stex_use_symbol_uri:n{#1}?#2}{##1}{
6362       \prop_map_break:n{
6363         %\stex_debug:nn{notations}{Found:~\tl_to_str:n{##2}}
6364         \bool_set_true:N \l_tmpa_bool

```

```

6365         \_stex_notations_set_macro:nnnnn ##2
6366     }
6367 }
6368 }
6369 \bool_if:NF \l_tmpa_bool {
6370     \errmessage{Woop}
6371 }
6372 }
6373
6374 \cs_new_protected:Nn \_stex_notations_set_macro:nnnnn {
6375     \tl_set:cn {\_stex_notations_macro:nn{\stex_use_symbol_uri:n{#1}}{#2}}{#4}
6376     \cs_if_exist:cF{\_stex_notations_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{
6377         \tl_set:cn {\_stex_notations_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{#4}
6378     }
6379     \tl_if_empty:nF{#5}{
6380         \tl_set:cn{\_stex_notations_op_macro:nn{\stex_use_symbol_uri:n{#1}}{#2}}{#5}
6381         \cs_if_exist:cF{\_stex_notations_op_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{
6382             \tl_set:cn{\_stex_notations_op_macro:nn{\stex_use_symbol_uri:n{#1}}{}}{#5}
6383         }
6384     }
6385 }

```

(End of definition for `\stex_add_notation:nnnnn`. This function is documented on page 145.)

sfunction

_stex_map_args:N
_stex_map_notation_args:N

```

6386 \cs_new:Nn \_stex_map_args:N {
6387     \tl_if_empty:NF \l_stex_get_symbol_args_tl {
6388         \exp_after:wN \_stex_notations_map_args_i:w \exp_after:wN
6389         #1 \l_stex_get_symbol_args_tl \_stex_notations_args_end:
6390     }
6391 }
6392 \cs_new:Npn \_stex_notations_map_args_i:w #1 #2 #3 #4 \_stex_notations_args_end: {
6393     #1 {#2} {#3}
6394     \tl_if_empty:NF{#4}{
6395         \_stex_notations_map_args_i:w #1 #4 \_stex_notations_args_end:
6396     }
6397 }
6398
6399 \cs_new:Nn \_stex_map_notation_args:N {
6400     \tl_if_empty:NF \l_stex_notation_args_tl {
6401         \exp_after:wN \_stex_notations_map_args_ii:w \exp_after:wN
6402         #1 \l_stex_notation_args_tl \_stex_notations_args_end:
6403     }
6404 }
6405 \cs_new:Npn \_stex_notations_map_args_ii:w #1 #2 #3 #4 #5 #6 \_stex_notations_args_end: {
6406     #1 {#2} {#3} {#4} {#5}
6407     \tl_if_empty:NF{#6}{
6408         \_stex_notations_map_args_ii:w #1 #6 \_stex_notations_args_end:
6409     }
6410 }

```

(End of definition for `_stex_map_args:N` and `_stex_map_notation_args:N`. These functions are documented on page 145.)

sfunction

`_stex_var_notation_macro:`

```

6411 \cs_new_protected:Nn \_stex_var_notation_macro: {
6412   \tl_set_eq:cN {\_stex_notations_macro:nn{\l_stex_key_name_str}{\l_stex_key_variant_str}}
6413   \cs_if_exist:cF {\_stex_notations_macro:nn{\l_stex_key_name_str}{}}{
6414     \tl_set_eq:cN{\_stex_notations_macro:nn{\l_stex_key_name_str}{}}
6415     \l_stex_notation_macrocode_cs
6416   }
6417   \tl_if_empty:NF \l_stex_key_op_tl {
6418     \tl_set_eq:cN {\_stex_notations_op_macro:nn{\l_stex_key_name_str}{\l_stex_key_variant_
6419     \cs_if_exist:cF{\_stex_notations_op_macro:nn{\l_stex_key_name_str}{}}{
6420       \cs_set_eq:cN{\_stex_notations_op_macro:nn{\l_stex_key_name_str}{}}
6421       \l_stex_key_op_tl
6422     }
6423   }
6424 }

```

(End of definition for `_stex_var_notation_macro:`. This function is documented on page 145.)

`sfunction`

`\setnotation`

`_stex_notation_set_default:n`

```

6425 \cs_new_protected:Nn \_stex_notation_set_default:n{
6426   \stex_if_in_module:TF{
6427     \stex_add_notation:ooeo{#1}{
6428       {\int_use:N \l_stex_get_symbol_arity_int}
6429       \l_stex_notation_macrocode_cs
6430       \l_stex_key_op_tl
6431     }{
6432       \cs_set_eq:cN {\_stex_notations_macro:nn {\stex_use_symbol_uri:N \l_stex_get_symbol_ur
6433       \l_stex_notation_macrocode_cs
6434       \tl_if_empty:NF \l_stex_key_op_tl {
6435         \cs_set_eq:cN{\_stex_notations_op_macro:nn{\stex_use_symbol_uri:N \l_stex_get_symbol
6436         \l_stex_key_op_tl
6437       }
6438     }
6439   }
6440
6441   \cs_new_protected:Npn \setnotation #1 #2 {
6442     \stex_get_symbol:n{#1}
6443     \cs_if_exist:cTF{\_stex_notations_macro:nn{\stex_use_symbol_uri:N \l_stex_get_symbol_uri
6444     \tl_set_eq:Nc \l_stex_notation_macrocode_cs {\_stex_notations_macro:nn{\stex_use_symbol
6445     \cs_if_exist:cTF{\_stex_notations_op_macro:nn{\stex_use_symbol_uri:N \l_stex_get_symbol
6446     \tl_set_eq:Nc \l_stex_key_op_tl {\_stex_notations_op_macro:nn{\stex_use_symbol_uri:N
6447     }{
6448       \tl_clear:N \l_stex_key_op_tl
6449     }
6450     \_stex_notation_set_default:n{
6451       \l_stex_get_symbol_uri
6452     }
6453   }{
6454     \msg_error:nnxx{stex}{error/unknownnotation}{#2}{
6455       \stex_use_symbol_uri:N \l_stex_get_symbol_uri
6456     }
6457   }
6458 }

```

(End of definition for `\setnotation` and `_stex_notation_set_default:n`. These functions are documented on page 145.)

sfunction

`\stex_iterate_notations:nn`

```

6459 \cs_new_protected:Nn \stex_iterate_notations:nn {
6460   \seq_clear:N \l__stex_notations_mods_seq
6461   \stex_pseudogroup_with:nn{\__stex_notations_not_cs:nnnnn}{
6462     \cs_set:Npn \__stex_notations_not_cs:nnnnn
6463       ##1 ##2 ##3 ##4 ##5 { #2 }
6464     \clist_map_function:nN {#1} \__stex_notations_it_not_i:n
6465   }
6466 }
6467
6468 \cs_new_protected:Nn \__stex_notations_it_not_i:n {
6469   \seq_if_in:NnF \l__stex_notations_mods_seq {#1} {
6470     \seq_put_left:Nn \l__stex_notations_mods_seq {#1}
6471     \prop_map_inline:cn{\_stex_notations_macro:n{#1}}{
6472       \__stex_notations_not_cs:nnnnn ##2
6473     }
6474     \prop_map_inline:cn{\_stex_morphisms_macro:n{#1}}{
6475       \__stex_notations_it_not_check:nnnn ##2
6476     }
6477   }
6478 }
6479 \cs_new_protected:Nn \__stex_notations_it_not_check:nnnn {
6480   \tl_if_empty:nT{#1}{
6481     \__stex_notations_it_not_i:n {#2}
6482   }
6483 }

```

(End of definition for `\stex_iterate_notations:nn`. This function is documented on page 146.)

sfunction

`\stex_use_notation:nnTF`

`\stex_use_op_notation:nnTF`

```

6484 \cs_new_protected:Npn \stex_use_notation:nnTF #1 #2 #3 #4 {
6485   \cs_if_exist:cTF{\_stex_notations_macro:nn{#1}{#2}}{
6486     \cs_set_eq:Nc \l_stex_notation_cs {\_stex_notations_macro:nn{#1}{#2}}
6487     #3
6488   }{#4}
6489 }
6490 \cs_new_protected:Npn \stex_use_op_notation:nnTF #1 #2 #3 #4 {
6491   \cs_if_exist:cTF{\_stex_notations_op_macro:nn{#1}{#2}}{
6492     \cs_set_eq:Nc \l_stex_notation_cs {\_stex_notations_op_macro:nn{#1}{#2}}
6493     #3
6494   }{#4}
6495 }

```

(End of definition for `\stex_use_notation:nnTF` and `\stex_use_op_notation:nnTF`. These functions are documented on page 146.)

sfunction

`_stex_notation_check:`

```

6496 \cs_new_protected:Nn \_stex_notation_check: {

```



```

6497 \stex_check_term:nn{notation}{
6498   \cs_set:Npn \comp ##1 {##1}
6499   \stex_debug:nn{check}{Checking~notation:~\cs_meaning:N \l_stex_notation_macrocode_cs ^^
6500   \exp_args:Nne \use:nn{\l_stex_notation_macrocode_cs}{}}{
6501     \stex_notation_make_args:
6502   }
6503 }
6504 }

```

(End of definition for `_stex_notation_check:`. This function is documented on page 146.)

sfunction

`_stex_notation_do_html:n`

```

6505 \cs_new_protected:Nn \_stex_notation_do_html:n {
6506   \stex_annotate_invisible:n{\_stex_annotate_force_break:n{\hbox{\stex_annotate:nn {
6507     data-ftml-notation={#1},
6508     data-ftml-notationfragment={\l_stex_key_variant_str},
6509     data-ftml-precedence={\l__stex_notations_opprec_tl},
6510     data-ftml-argprec={\seq_use:Nn \l__stex_notations_precs_seq ,}
6511   }}{
6512     \cs_set_protected:Npn \argsep ##1 ##2 {
6513       \stex_annotate:nn{data-ftml-argsep={}}{
6514         ##1 \tl_if_empty:nTF{##2}{\!\,}{##2}
6515       }
6516     }
6517     \cs_set_protected:Npn \argmap ##1 ##2 ##3 {
6518       \cs_set:Npn \__stex_notations_map_cs: #####1 { ##2 }
6519       \stex_annotate:nn{data-ftml-argmap={}}{
6520         \__stex_notations_map_cs:{##1} \stex_annotate:nn{data-ftml-argmap-sep={}}{##3}
6521       }
6522     }
6523     \cs_set_protected:Npn \maincomp {
6524       \do_comp:nnNn {maincomp}{}\compemph@uri
6525     }
6526     \stex_debug:nn{notation}{Doing~notation~HTML:\meaning\l_stex_get_symbol_args_tl}
6527     $
6528     \tl_clear:N \l_stex_current_full_tl
6529     \stex_annotate:nn{data-ftml-notationcomp={}}{
6530       \exp_args:Nne \use:nn {
6531         \l_stex_notation_macrocode_cs {}
6532       }{
6533         \stex_map_args:N \__stex_notations_make_arg_html:nn
6534       }
6535     }
6536     $
6537     \tl_if_empty:NF \l_stex_key_op_tl {
6538       $
6539       \tl_clear:N \l_stex_current_full_tl
6540       \stex_annotate:nn{data-ftml-notationopcomp={}}{
6541         \stex_term_oms:nn{\l_stex_key_variant_str}{\l_stex_key_op_tl}
6542       }
6543       $
6544     }
6545   }}

```

```

6546   }}
6547 }
6548
6549 \cs_new:Nn \__stex_notations_make_arg_html:nn {
6550   {\stex_annotate:nn{data-ftml-argnum=#1}{x}}
6551 }

```

(End of definition for `_stex_notation_do_html:n`. This function is documented on page 146.)

sfunction

`_stex_notation_make_args:`

```

6552 \cs_new:Nn \_stex_notation_make_args: {
6553   \_stex_map_notation_args:N \__stex_notations_make_arg:nnnn
6554 }
6555
6556 \cs_new:Nn \__stex_notations_make_arg:nnnn {
6557   \str_case:nnF #2 {
6558     a {{
6559       a\c_math_subscript_token{#1,1},
6560       a\c_math_subscript_token{#1,2}
6561     }}
6562     B {{
6563       B\c_math_subscript_token{#1,1},
6564       B\c_math_subscript_token{#1,2}
6565     }}
6566   }{{
6567     \_stex_term_arg:nnnn{#1}{#2}{#3}{#4}
6568     {{#2}\c_math_subscript_token{#1}}
6569   }}
6570 }

```

(End of definition for `_stex_notation_make_args:.` This function is documented on page 146.)

sfunction

`\stex_do_default_notation:`

`\stex_do_default_notation_op:`

```

6571 \cs_new_protected:Nn \stex_do_default_notation: {
6572   \stex_do_default_notation_op:
6573   \tl_if_empty:NTF \l_stex_current_args_tl {
6574     \tl_clear:N \l__stex_notations_args_tl
6575   }\__stex_notations_default_args:
6576   \tl_set:Ne \l_stex_default_notation {\STEXInternalNotation{}{0}{}}{\l__stex_notations_args
6577   \exp_args:No \exp_not:n \l_stex_default_notation
6578   }}
6579 }
6580
6581 \cs_new_protected:Nn \stex_do_default_notation_op: {
6582   \__stex_notations_make_name:
6583   \tl_set:Ne \l_stex_default_notation {\exp_not:N \maincomp{ \exp_not:N \mathrm {\l__stex_n
6584   }
6585
6586 \cs_new_protected:Nn \__stex_notations_default_args: {
6587   \__stex_notations_make_name:
6588   \tl_set_eq:NN \l_stex_get_symbol_args_tl \l_stex_current_args_tl
6589   \tl_set:Ne \l__stex_notations_args_tl {

```

```

6590   \stex_map_args:N \__stex_notations_augment_arg:nn
6591 }
6592 \tl_put_right:Nn \l_stex_default_notation {\comp{}}
6593 \seq_clear:N \l_tmpa_seq
6594 \int_step_inline:nn \l_stex_current_arity_str {
6595   \seq_put_right:Nn \l_tmpa_seq {#### #1}
6596 }
6597 \tl_put_right:Ne \l_stex_default_notation {
6598   \seq_use:Nn \l_tmpa_seq {\mathpunct{\comp{,}}}}
6599 }
6600 \tl_put_right:Nn \l_stex_default_notation {\comp{}}
6601 }
6602
6603 \cs_new:Nn \__stex_notations_augment_arg:nn {
6604   #1#2{0}{}}
6605 }
6606
6607 \cs_new_protected:Nn \__stex_notations_make_name: {
6608   \exp_args:NNNe \seq_set_split:Nnn \l_tmpa_seq / \l_stex_current_display_tl
6609   \seq_pop_right:NN \l_tmpa_seq \l__stex_notations_name_str
6610 }

```

(End of definition for `\stex_do_default_notation:` and `\stex_do_default_notation_op:`. These functions are documented on page 146.)

sfunction

`\STEXInternalNotation`

```

6611 \cs_new_protected:Npn \STEXInternalNotation #1 #2 #3 #4 #5 #6 {
6612   \__stex_notations_process:nnnnnn{#1}{#2}{#3}{#4}{#5}{
6613     \l__stex_notations_code_tl
6614     #6
6615   }
6616 }
6617
6618 \cs_new_protected:Npn \__stex_notations_process:nnnnnn #1 #2 #3 #4 {
6619   \tl_if_empty:nTF{#4}{
6620     \__stex_notations_simple:nnnnn{#1}{#2}{#3}
6621   }{
6622     \__stex_notations_complex:nnnnnn{#1}{#2}{#3}{#4}
6623   }
6624 }
6625
6626 \cs_new_protected:Nn \__stex_notations_simple:nnnnn {
6627   \stex_debug:nn{Notation~code}{\tl_to_str:n{#4}}
6628   \tl_set:Nn \l__stex_notations_code_tl {
6629     \cs_set:Npn \l__stex_notations_cs {
6630       \stex_maybe_brackets:nn{#2}{
6631         \stex_term_oms_or_omv:nn{#1}{#4}
6632       }
6633     }
6634     \l__stex_notations_cs
6635   }
6636   #5
6637 }

```

```

6638
6639 \cs_new_protected:Nn \__stex_notations_complex:nnnnnn {
6640   \stex_debug:nn{Notation~code}{\tl_to_str:n{#5}}
6641   \int_zero:N \l_tmpa_int
6642   \tl_set:Nn \l__stex_notations_pre_tl {\cs_set_eq:NN \stex_term_oma_or_omb:nn \stex_term_
6643   \tl_set:Nn \l__stex_notations_code_tl {
6644     \cs_generate_from_arg_count:NNnn \l__stex_notations_cs \cs_set:Npn \l_tmpa_int
6645     {
6646       \stex_maybe_brackets:nn{#2}{
6647         \stex_term_oma_or_omb:nn{#1}{
6648           \bool_set_false:N \l_stex_brackets_dones_bool
6649           #5
6650         }
6651       }
6652     }
6653     \l__stex_notations_cs
6654   }
6655   \tl_set:Nn \l__stex_notations_after_tl{
6656     \exp_args:NNo
6657     \tl_put_left:Nn \l__stex_notations_code_tl \l__stex_notations_pre_tl
6658     \tl_put_left:Ne \l__stex_notations_code_tl {
6659       \int_set:Nn \l_tmpa_int {\int_use:N \l_tmpa_int}
6660     }
6661     #6
6662   }
6663   \__stex_notations_parse_args:nnnnw #4 \__stex_notations_args_end:
6664 }
6665
6666 \cs_new_protected:Npn \__stex_notations_parse_args:nnnnw #1 #2 #3 #4 #5 \__stex_notations_a
6667   \tl_if_empty:nTF{#5}{
6668     \__stex_notations_add_last:nnnnn{#1}{#2}{#3}{#4}
6669   }{
6670     \__stex_notations_add_next:nnnnnn{#1}{#2}{#3}{#4}{#5}
6671   }
6672 }
6673
6674 \cs_new_protected:Nn \__stex_notations_add_next:nnnnnn {
6675   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#6}
6676   \__stex_notations_parse_args:nnnnw #5 \__stex_notations_args_end:
6677 }
6678
6679 \cs_new_protected:Nn \__stex_notations_add_last:nnnnn {
6680   \__stex_notations_add:nnnnn{#1}{#2}{#3}{#4}{#5}
6681   \stex_debug:nn{sequences}{Executing~notation:~\meaning\l__stex_notations_code_tl}
6682   \l__stex_notations_after_tl
6683 }
6684
6685 \cs_new_protected:Nn \__stex_notations_add:nnnnn {
6686   \int_incr:N \l_tmpa_int
6687   \str_case:nn{#2}{
6688     i {
6689       \tl_put_right:Nn \l__stex_notations_code_tl {
6690         {\stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
6691       }

```

```

6692 }
6693 b {
6694   \tl_set:Nn \l__stex_notations_pre_tl {
6695     \cs_set_eq:NN \stex_term_oma_or_omb:nn \stex_term_omb:nn
6696   }
6697   \tl_put_right:Nn \l__stex_notations_code_tl {
6698     {\stex_term_arg:nnnnn{#1}{#2}{#3}{#4}{#5}}
6699   }
6700 }
6701 a {
6702   \tl_put_right:Nn \l__stex_notations_code_tl {
6703     {\stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
6704   }
6705 }
6706 B {
6707   \tl_set:Nn \l__stex_notations_pre_tl {
6708     \cs_set_eq:NN \stex_term_oma_or_omb:nn \stex_term_omb:nn
6709   }
6710   \tl_put_right:Nn \l__stex_notations_code_tl {
6711     {\stex_term_arg_aB:nnnnn{#1}{#2}{#3}{#4}{#5}}
6712   }
6713 }
6714 }
6715 }

```

(End of definition for \STEXInternalNotation. This function is documented on page 146.)

sfunction

\argsep

```

6716 \cs_new_protected:Npn \argsep #1 #2 {
6717   \__stex_notations_check_aB_arg:Nn\argsep{#1}
6718   \stex_pseudogroup_with:nn{\stex_term_do_aB_clist:}{
6719     \tl_set:Nn \stex_term_do_aB_clist: {
6720       \seq_use:Nn \l_stex_aB_args_seq {#2}
6721     }
6722     #1
6723   }
6724 }
6725
6726 \cs_new_protected:Nn \__stex_notations_check_aB_arg:Nn {
6727   \exp_args:Ne \cs_if_eq:NNF {\tl_head:n{#2}}
6728   \stex_term_arg_aB:nnnnn {
6729     \msg_error:nnx{stex}{error/assocarg}{\tl_to_str:n{#1}}
6730   }
6731 }

```

(End of definition for \argsep. This function is documented on page 146.)

sfunction

\seqmap implemented in notations via “pattern matching” the argument

```

6732 \cs_new_protected:Npn \seqmap #1 #2 {
6733   \symuse{Metatheory?sequence~expression}{\seqmap{#1}{#2}}%\l_tmpa_tl {#2}
6734 }

```

(End of definition for \seqmap. This function is documented on page 146.)

sfunction

\argmap

```

6735 \cs_new_protected:Npn \argmap #1 #2 #3 {
6736   \__stex_notations_check_aB_arg:Nn\argmap{#1}
6737   \stex_pseudogroup_with:nn{
6738     \_stex_term_do_aB_clist:
6739     \__stex_notations_map_cs:
6740   }{
6741     \cs_set:Npn \__stex_notations_map_cs: ##1 { #2 }
6742     \tl_set:Nn \_stex_term_do_aB_clist: {
6743       \seq_clear:N \l_tmpa_seq
6744       \seq_map_inline:Nn \l_stex_aB_args_seq {
6745         \tl_if_eq:nnTF{##1}{\ellipses}{
6746           \seq_put_right:Nn \l_tmpa_seq \ellipses
6747         }{
6748           \seq_put_right:Nx \l_tmpa_seq {
6749             \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
6750           }
6751         }
6752       }
6753       \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
6754       \seq_use:Nn \l_stex_aB_args_seq {#3}
6755     }
6756     #1
6757   }
6758 }
```

(End of definition for \argmap. This function is documented on page 146.)

sfunction

\argarraymap

```

6759 \int_new:N \l__stex_notations_clist_count_int
6760 \cs_new_protected:Npn \argarraymap #1 #2 #3 #4 {
6761   \__stex_notations_check_aB_arg:Nn\argarraymap{#1}
6762   \stex_pseudogroup_with:nn{
6763     \_stex_term_do_aB_clist:
6764     \__stex_notations_map_cs:
6765   }{
6766     \cs_set:Npn \__stex_notations_map_cs: ##1 { #3 }
6767     \int_set:Nn \l__stex_notations_clist_count_int {\exp_args:No\clist_count:n{\tl_to_str:n{#4}}}
6768     \tl_set:Nn \_stex_term_do_aB_clist: {
6769       \tl_clear:N \l_tmpa_tl
6770       \int_zero:N \l_tmpa_int
6771       \seq_map_inline:Nn \l_stex_aB_args_seq {
6772         \int_incr:N \l_tmpa_int
6773         \int_compare:nNnT \l_tmpa_int > \l__stex_notations_clist_count_int {
6774           \int_set:Nn \l_tmpa_int 1
6775         }
6776         \tl_put_right:Ne \l_tmpa_tl {
6777           \exp_after:wN \exp_not:n \exp_after:wN { \__stex_notations_map_cs: {##1} }
6778           \clist_item:nn{#4}\l_tmpa_int
6779         }
6780       }
6781     }
```

```

6780     }
6781     \seq_set_eq:NN \l_stex_aB_args_seq \l_tmpa_seq
6782     \begin{array}{#2}
6783       \l_tmpa_tl
6784     \end{array}
6785   }
6786   #1
6787 }
6788 }

```

(End of definition for `\argarraymap`. This function is documented on page 146.)
sfunction

33.1 Automated Bracketing

Variables for current (downwards) precedence, set of parentheses etc.:

```

6789 \int_new:N \l_stex_notation_downprec
6790 \tl_set:Nn \l__stex_notations_left_bracket_str (
6791 \tl_set:Nn \l__stex_notations_right_bracket_str )
6792 \bool_new:N \l_stex_brackets_dones_bool

\infprec
\neginfprec
6793 \tl_const:Ne \infprec {\int_use:N \c_max_int}
6794 \tl_const:Ne \neginfprec {-\int_use:N \c_max_int}
6795
6796 \int_set:Nn \l_stex_notation_downprec \infprec

```

(End of definition for `\infprec` and `\neginfprec`. These variables are documented on page 147.)

`_stex_maybe_brackets:nn`

```

6797 \cs_new_protected:Nn \_stex_maybe_brackets:nn {
6798   \bool_if:NTF \l_stex_brackets_dones_bool {
6799     \bool_set_false:N \l_stex_brackets_dones_bool
6800     #2
6801   } {
6802     \stex_debug:nn{brackets}{#1>\int_eval:n \l_stex_notation_downprec?}
6803     \int_compare:nNnTF { #1 } > \l_stex_notation_downprec {
6804       %\bool_if:NTF \l_stex_inarray_bool { #2 }{
6805         \dobrackets {
6806           #2
6807         }
6808       %}
6809     }{
6810       #2
6811     }
6812   }
6813 }

```

(End of definition for `_stex_maybe_brackets:nn`. This function is documented on page 147.)
sfunction

`\dobrackets`

```
6814 \cs_new_protected:Npn \dobrackets #1 {  
6815   \group_begin:  
6816     \bool_set_true:N \l_stex_brackets_dones_bool  
6817     \mathopen{\tl_if_exist:NT\l_stex_current_symbol_uri\comp  
6818       \l__stex_notations_left_bracket_str  
6819     }  
6820     #1  
6821   \group_end:  
6822   \mathclose{\tl_if_exist:NT\l_stex_current_symbol_uri\comp  
6823     \l__stex_notations_right_bracket_str  
6824   }  
6825 }
```

(End of definition for \dobrackets. This function is documented on page 147.)

sfunction

`\withbrackets`

```
6826 \cs_new_protected:Npn \withbrackets #1 #2 #3 {  
6827   \stex_pseudogroup:nn{  
6828     \tl_set:Nn \l__stex_notations_left_bracket_str { #1 }  
6829     \tl_set:Nn \l__stex_notations_right_bracket_str { #2 }  
6830     #3  
6831   }{  
6832     \stex_pseudogroup_restore:N \l__stex_notations_left_bracket_str  
6833     \stex_pseudogroup_restore:N \l__stex_notations_right_bracket_str  
6834   }  
6835 }
```

(End of definition for \withbrackets. This function is documented on page 147.)

sfunction

`\dowithbrackets`

```
6836 \cs_new_protected:Npn \dowithbrackets #1 #2 #3 {  
6837   \withbrackets{#1}{#2}{\dobrackets{#3}}  
6838 }
```

(End of definition for \dowithbrackets. This function is documented on page 147.)

sfunction

Chapter 34

Structures

```
6839 <@@=stex_structures>

Keys:
6840 \stex_keys_define:nnnn{mathstructure}{
6841   \tl_clear:N \l_stex_current_this_tl
6842   \str_clear:N \l__stex_structures_name_str
6843 }{
6844   this .tl_set:N = \l_stex_current_this_tl ,
6845   unknown .code:n = {
6846     \str_if_empty:NTF \l_keys_key_str {
6847       \str_set:Ne \l__stex_structures_name_str {\l_keys_key_tl}
6848     }{
6849       \str_set_eq:NN \l__stex_structures_name_str \l_keys_key_str
6850     }
6851   }
6852 }{}
```

`mathstructure (env.)`

```
6853 \stex_new_stylable_env:nnnnnnn {mathstructure}{m 0{}}{
6854   \__stex_structures_begin:nn{#1}{#2}
6855   \stex_smsmode_do:
6856 }{
6857   \stex_structural_feature_module_end:
6858   \__stex_structures_do externals:
6859 }{}{}{}
6860
6861 \stex_sms_allow_env:n{mathstructure}
6862 \stex_deactivate_macro:Nn \mathstructure {module~environments}
6863 \stex_every_module:n {\stex_reactivate_macro:N \mathstructure}
```

Shared code for `mathstructure` and `extstructure`:

```
6864 \cs_new_protected:Nn \__stex_structures_begin:nn {
6865   \stex_keys_set:nn {mathstructure}{#2}
6866   \str_if_empty:NT \l__stex_structures_name_str {
6867     \str_set:Nn \l__stex_structures_name_str {#1}
6868   }
6869   \def\comp{\_comp}
6870
6871   \tl_set:Ne \l__stex_structures_struct_uri {
```

```

6872 \exp_args:No \stex_new_module_uri:n \l__stex_structures_name_str
6873 }
6874
6875 \exp_args:Nne \use:nn { \stex_add_symbol:nnnnnnN }
6876 { {#1}{\l__stex_structures_name_str}{0}{\defed}{\l__stex_structures_struct_uri}}
6877 {\stex_invoke_structure:
6878 \str_set:Ne \l_stex_macroname_str {#1}
6879
6880 \exp_args:No \stex_structural_feature_module:nn
6881 {\l__stex_structures_name_str}{structure}
6882 }
6883
6884 \cs_new_protected:Nn \__stex_structures_doexternals: {
6885 \tl_set:Nn \l__stex_structures_replace_this_tl {####1}
6886 }
6887

```

extstructure (env.)

```

6888 \stex_new_stylable_env:nnnnnn {extstructure}{m O{} m}{
6889 \seq_clear:N \l__stex_structures_imports_seq
6890 \clist_map_inline:nn{#3}{
6891 \stex_get_mathstructure:n{##1}
6892 \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6893 \stex_if_module_exists:nT{####1}{
6894 \seq_put_right:Nn \l__stex_structures_imports_seq{####1}
6895 }
6896 }
6897 }
6898 \__stex_structures_begin:nn{#1}{#2}
6899 \seq_map_inline:Nn\l__stex_structures_imports_seq{
6900 \stex_if_do_html:T {
6901 \hbox{\stex_annotate_invisible:nn
6902 {data-ftml-import={\stex_use_module_uri:n{##1}}}{}}
6903 }
6904 \stex_add_morphism:nonn
6905 {}{##1}{import}{}
6906 \stex_execute_in_module:e{
6907 \stex_activate_module:n {##1}
6908 }
6909 }
6910 \stex_smsmode_do:
6911 }{
6912 \stex_structural_feature_module_end:
6913 \__stex_structures_doexternals:
6914 }{}{}
6915
6916 \stex_sms_allow_env:n{extstructure}
6917 \stex_deactivate_macro:Nn \extstructure {module~environments}
6918 \stex_every_module:n {
6919 \stex_reactivate_macro:N \extstructure
6920 }
6921
6922 \stex_new_stylable_env:nnnnnn {extstructure*}{m}{
6923 \__stex_structures_new_extstruct_name:

```

```

6924 \seq_clear:N \l__stex_structures_imports_seq
6925 \stex_get_mathstructure:n{#1}
6926
6927 \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
6928   \stex_if_module_exists:nT{##1}{
6929     \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
6930   }
6931 }
6932
6933 \stex_module_uri_split_name:NNN \l__stex_structures_parent_uri \l__stex_structures_last_n
6934
6935 \stex_execute_in_module:e{
6936   \__stex_structures_extend_structure:nnnn{\l__stex_structures_parent_uri}{\l__stex_struct
6937 }
6938
6939 \exp_args:No \__stex_structures_begin:nn\l__stex_structures_exstruct_name_str{
6940
6941 \seq_map_inline:Nn \l__stex_structures_imports_seq {
6942   \stex_if_do_html:T {
6943     \stex_annotate_invisible:nn
6944     {data-ftml-import= {\stex_use_module_uri:n{##1}}}{ }
6945   }
6946   %\stex_add_morphism:nonn
6947   % {}{##1}{import}{ }
6948   %\stex_execute_in_module:e{
6949   % \stex_activate_module:n {##1}
6950   %}
6951   \stex_activate_module:n {##1}
6952 }
6953
6954 \stex_smsmode_do:
6955 }{
6956 \prop_map_inline:cn{\_stex_symbol_macro:N \l_stex_current_module_uri }{
6957   \__stex_structures_check_def:nnnnnnn ##2
6958 }
6959 \stex_structural_feature_module_end:
6960 %\exp_args:Ne \stex_do_up_to_module:n {
6961 % \stex_activate_module:n {\exp_args:No \stex_new_module_uri:n {\l__stex_structures_exst
6962 %}
6963 \__stex_structures_do externals:
6964 }{}{}{}
6965
6966 \stex_sms_allow_env:n{extstructure*}
6967 \exp_after:wN \stex_deactivate_macro:Nn
6968 \cs:w extstructure*\cs_end: {module~environments}
6969 \stex_every_module:n {
6970 \exp_after:wN \stex_reactivate_macro:N \cs:w extstructure*\cs_end:
6971 }
6972
6973 \cs_new_protected:Nn \__stex_structures_extend_structure:nnnn {
6974 \stex_debug:nn{ext}{Extending~\stex_use_module_uri:n {#1} / #2-by~\stex_use_module_uri:n
6975 \exp_args:Nne \tl_put_right:cn{\_stex_module_code_macro:n{#4}}{
6976 \stex_activate_module:n{#3}
6977 }

```

```

6978 \exp_args:Nne \prop_put:cnn{\_stex_morphisms_macro:n{#4}}{\stex_use_module_uri:n{#3}}{
6979   {}{#3}{extstruct}}{
6980 }
6981 \exp_args:Nne \use:nn {\_stex_structures_extend_ii:nnnn}{
6982   \exp_args:Nne \use:nn \_stex_structures_extend_i:nnnnnnnnNn {
6983     \prop_item:cn{\_stex_symbol_macro:n {#1} } { #2 }
6984     } { #3 }
6985   }{#1}{#2}
6986 }
6987
6988 \cs_new_protected:Nn \_stex_structures_extend_ii:nnnn {
6989   \prop_put:cnn{\_stex_symbol_macro:n {#3} } { #4 }{#2}
6990   \tl_set:ce{#1}{
6991     \stex_invoke_symbol:nnnnnnN {\stex_new_symbol_uri:nn {#3}{#4}}
6992     \use_none:nn #2
6993   }
6994 }
6995
6996 \cs_new:Nn \_stex_structures_extend_i:nnnnnnnnNn {
6997   {#1}{#{#1}{#2}{#3}{#4}{#5}{#6,#9}{#7}#8}
6998 }
6999
7000 \cs_new_protected:Nn \_stex_structures_check_def:nnnnnnnn {
7001   \tl_if_empty:nT{#5}{
7002     \msg_error:nnnn{stex}{error/needsdefinien}{#2}{extstructure*}
7003   }
7004 }
7005
7006 \cs_new_protected:Nn \_stex_structures_new_extstruct_name: {
7007   \stex_do_up_to_module:n {
7008     \str_set:Ne \l_stex_structures_extname_count {\int_eval:n{\l_stex_structures_extname_
7009   }
7010   \str_set:Ne \l_stex_structures_exstruct_name_str {EXTSTRUCT\_l_stex_structures_extname_
7011 }
7012
7013 \stex_every_module:n{ \str_set:Nn \l_stex_structures_extname_count 0}

```

\this

```

7014 \bool_new:N \l_stex_in_this_bool
7015
7016 \cs_new_protected:Npn \stex_current_this: {
7017   { \bool_set_true:N \l_stex_allow_semantic_bool
7018     \tl_if_empty:NTF \l_stex_current_this_tl {} }{
7019     \tl_set:Nn \l_stex_current_full_tl {
7020       \exp_args:Ne \stex_use_symbol_uri:n {
7021         \exp_args:No \stex_new_symbol_uri:n \l_stex_structures_name_str
7022       }
7023     }
7024     \str_set_eq:NN \l_stex_current_display_tl \l_stex_structures_name_str
7025     \bool_set_true:N \l_stex_in_this_bool
7026     \maincomp{\l_stex_current_this_tl}
7027     \bool_set_false:N \l_stex_in_this_bool
7028   }
7029 }

```

```

7030 }
7031
7032 \let \this \stex_current_this:

```

(End of definition for \this. This function is documented on page 148.)

```

sfunction

```

\stex_get_mathstructure:n

```

7033 \cs_new_protected:Nn \stex_get_mathstructure:n {
7034   \tl_clear:N \l_stex_get_structure_module_uri
7035   \stex_get_symbol:nF{#1}{
7036     \msg_error:nnn{stex}{error/unknownstructure}{#1}
7037   }
7038   \exp_args:No \tl_if_eq:NNTF \l_stex_get_symbol_invoke_cs \stex_invoke_structure: {
7039     \tl_set:Nc \l_stex_get_structure_module_uri { \exp_after:wN \__stex_structures_get:w \l
7040   }{
7041     \msg_error:nnn{stex}{error/unknownstructure}{#1}
7042   }
7043 }
7044
7045 \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 148.)

```

sfunction

```

\usestructure

```

7046 \cs_new_protected:Npn \usestructure #1 {
7047   \stex_get_mathstructure:n{ #1 }
7048   \stex_debug:nn{usestructure}{Using-structure-#1:-\stex_use_module_uri:N \l_stex_get_struct
7049   \seq_clear:N \l__stex_structures_imports_seq
7050   \clist_map_inline:Nn \l_stex_get_symbol_type_tl {
7051     \stex_if_module_exists:nT{##1}{
7052       \seq_put_right:Nn \l__stex_structures_imports_seq{##1}
7053     }
7054   }
7055   \seq_map_inline:Nn \l__stex_structures_imports_seq {
7056     \stex_if_do_html:T {
7057       \hbox{\stex_annotate_invisible:nn
7058         {data-ftml-usemodule=\stex_use_module_uri:n {##1}} {}}
7059     }
7060     \stex_activate_module:n {##1}
7061   }
7062 }

```

(End of definition for \usestructure. This function is documented on page 148.)

```

sfunction

```

Chapter 35

Statements

```
7063 <@@=stex_statements>
```

Keys:

```
7064 \stex_keys_define:nnnn{statement}{  
7065   \str_clear:N \l_stex_key_name_str  
7066   \str_clear:N \l_stex_key_macroname_str  
7067   \clist_clear:N \l_stex_key_for_clist  
7068   \str_clear:N \l_stex_key_args_str  
7069   \tl_clear:N \l_stex_key_type_tl  
7070   \tl_clear:N \l_stex_key_def_tl  
7071   \tl_clear:N \l_stex_key_return_tl  
7072   \clist_clear:N \l_stex_key_argtypes_clist  
7073 }{  
7074   name      .str_set:N = \l_stex_key_name_str ,  
7075   for       .clist_set:N = \l_stex_key_for_clist ,  
7076   macro     .str_set:N = \l_stex_key_macroname_str ,  
7077   % start   .str_set:N = \l_stex_key_title_str , % TODO remove  
7078   type      .tl_set:N = \l_stex_key_type_tl ,  
7079   judgment .code:n = {},  
7080   from      .code:n= {}, % TODO remove  
7081   to        .code:n={} % TODO remove  
7082 }{id,title,style,symargs}
```

`\stex_do_for_list:`

```
7083 \cs_new_protected:Npn \stex_do_for_list: {  
7084   \seq_clear:N \l_stex_fors_seq  
7085   \clist_map_inline:Nn \l_stex_key_for_clist {  
7086     \exp_args:Ne\stex_get_symbol:n{\tl_to_str:n{##1}}  
7087     \seq_put_right:No \l_stex_fors_seq  
7088     {\l_stex_get_symbol_uri}  
7089   }  
7090 }
```

(End of definition for \stex_do_for_list:. This function is documented on page 149.)

sfunction

`\stex_new_statement:nnn`

```
7091 \cs_new_protected:Nn \stex_new_statement:nnn {  
7092   \stex_new_stylable_env:nnnnnnn {#1}{0}}{
```

```

7093 \stex_keys_set:nn{statement}{##1}
7094 #3
7095
7096 \stex_if_smsmode:F {
7097   \exp_args:Nne \begin{stex_annotate_env}{
7098     \__stex_statements_html_keyvals:nn{#1}{false}
7099   }
7100   \tl_set_eq:NN \thistitle \l_stex_key_title_tl
7101   \str_set_eq:NN \thisname \l_stex_key_name_str
7102   \clist_set_eq:NN \thisfor \l_stex_key_for_str
7103   \stex_if_html_backend:TF {
7104     \noindent
7105     \stex_annotate:nn{data-ftml-title={}, style:display={none}}{\_stex_annotate_force_b
7106     \tl_if_empty:NF \l_stex_key_title_tl{~}
7107     \l_stex_key_title_tl
7108   }}
7109 }
7110 \stex_style_apply:
7111 }
7112 \_stex_do_id:
7113 \stex_smsmode_do:
7114 }{
7115   \stex_if_smsmode:F {
7116     \stex_if_html_backend:F \stex_style_apply:
7117     \end{stex_annotate_env}
7118   }
7119 }{}{}{s}
7120 \stex_sms_allow_env:n{s#1}
7121
7122 \tl_if_empty:NF{#2}{\__stex_statements_make_macro:nnn{#1}{#2}{#3}}
7123 }
7124
7125 \cs_new_protected:Nn \__stex_statements_make_macro:nnn {
7126   \exp_after:wN \NewDocumentCommand \cs:w inline#2\cs_end: { 0{} m}{
7127     \group_begin:
7128     \stex_keys_set:nn{statement}{##1}
7129     #3
7130     \_stex_do_id:
7131     \stex_if_smsmode:F{
7132       \exp_args:Ne \stex_annotate:nn{\__stex_statements_html_keyvals:nn{#1}{true}}{
7133         \_stex_annotate_force_break:n{##2}
7134       }
7135     }
7136     \group_end:
7137     %\stex_if_smsmode:TF \stex_smsmode_do: \stex_ignore_spaces_and_pars:
7138     \stex_smsmode_do:
7139   }
7140   \exp_after:wN \stex_sms_allow_escape:N\cs:w inline#2\cs_end:
7141 }
7142
7143 \cs_new:Nn \__stex_statements_html_keyvals:nn {
7144   data-ftml-#1={},
7145   data-ftml-inline={#2},
7146   \seq_if_empty:NF \l_stex_fors_seq {,

```

```

7147     data-ftml-fors={\seq_map_indexed_function:NN \l_stex_fors_seq \__stex_statements_comma
7148   }
7149   \str_if_empty:NF \l_stex_key_id_str {,
7150     data-ftml-id={\l_stex_key_id_str}%{\stex_uri_use:N \l_stex_current_doc_uri ? \l_stex_ke
7151   }
7152   \clist_if_empty:NF \l_stex_key_style_clist {,
7153     data-ftml-styles={\l_stex_key_style_clist}
7154   }
7155 }
7156
7157 \cs_new:Nn \__stex_statements_comma_sep_uri:nn {
7158   \int_compare:nNnF{#1}=1 {,}
7159   \stex_use_symbol_uri:n{#2}
7160 }

```

(End of definition for `\stex_new_statement:nnn`. This function is documented on page 149.)

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.

```

7161 \cs_new_protected:Nn \__stex_statements_setup:n {
7162   \str_if_empty:NF \l_stex_key_macroname_str {
7163     \str_if_empty:NT \l_stex_key_name_str {
7164       \str_set_eq:NN \l_stex_key_name_str \l_stex_key_macroname_str
7165     }
7166   }
7167   \stex_do_for_list:
7168   \tl_clear:N \l_stex_current_def_uri
7169
7170   \str_if_empty:NTF \l_stex_key_name_str \__stex_statements_setup_noname: {
7171     \__stex_statements_setup_named:n{#1}
7172   }
7173 }
7174
7175 \cs_new_protected:Nn \__stex_statements_setup_named:n {
7176   \__stex_statements_force_id:
7177   \seq_put_right:Ne \l_stex_fors_seq {
7178     \l_stex_current_module_uri {\l_stex_key_name_str}
7179   }
7180   \str_set_eq:NN \l_stex_macroname_str \l_stex_key_macroname_str
7181   \str_set:Nn \l_stex_key_role_str {#1}
7182   \stex_symdecl_do:
7183   \exp_args:Nne \use:nn {\stex_add_symbol:nnnnnnN}{
7184     {\l_stex_key_macroname_str}{\l_stex_key_name_str}
7185     {\int_use:N \l_stex_get_symbol_arity_int}
7186     {\l_stex_get_symbol_args_tl}
7187     {#1}{}}{\stex_invoke_symbol:
7188   }
7189   \stex_if_do_html:T \stex_symdecl_html:
7190   \tl_set:Ne \l_stex_current_def_uri {\exp_args:No \stex_new_symbol_uri:nn \l_stex_current_
7191   \stex_debug:nn{statement}{name:~\stex_use_symbol_uri:N \l_stex_current_def_uri}
7192 }
7193
7194 \cs_new_protected:Nn \__stex_statements_setup_noname: {

```



```

7195 \stex_debug:nn{statement}{no~name}
7196 \int_compare:nNnTF {\seq_count:N \l_stex_fors_seq} = 1 {
7197   \tl_set:Nc \l_stex_current_def_uri {\seq_item:Nn \l_stex_fors_seq 1}
7198   \stex_debug:nn{statement}{for:~\stex_use_symbol_uri:N \l_stex_current_def_uri}
7199 }{
7200   \stex_debug:nn{statement}{no~for}
7201 }
7202 }
7203
7204 \cs_new_protected:Nn \__stex_statements_force_id: {
7205   %\str_if_empty:NT \l_stex_key_id_str {
7206   %   \stex_ref_new_id:n{
7207   %     \str_set_eq:NN \l_stex_key_id_str \l__stex_refs_str
7208   %   }
7209 }

```

(End of definition for __stex_statements_setup:n.)

__stex_statements_setup_def: Sets up all the macros allowed in definition-like environments:

```

7210 \cs_new_protected:Nn \__stex_statements_setup_def: {
7211   \stex_if_smsmode:F{
7212     \seq_map_inline:Nn \l_stex_fors_seq {
7213       \exp_args:Ne \stex_ref_new_sym_target:n{\stex_use_symbol_uri:n{##1}}
7214     }
7215   }
7216   \stex_reactivate_macro:N \definiendum
7217   \stex_reactivate_macro:N \defnotation
7218   \stex_reactivate_macro:N \definame
7219   \stex_reactivate_macro:N \Definame
7220   \stex_reactivate_macro:N \varbind
7221   \stex_reactivate_macro:N \definiens
7222 }

```

(End of definition for __stex_statements_setup_def:.)

sdefinition (env.)

```

7223 \stex_new_statement:nnn{definition}{def}{
7224   \__stex_statements_force_id:
7225   \__stex_statements_setup:n{
7226     \__stex_statements_setup_def:
7227   }

```

sassertion (env.)

```

7228 \stex_new_statement:nnn{assertion}{ass}{
7229   \__stex_statements_setup:n{assertion}
7230   \stex_if_smsmode:F{
7231     \seq_map_inline:Nn \l_stex_fors_seq {
7232       \exp_args:Ne \stex_ref_new_sym_target:n{\stex_use_symbol_uri:n{##1}}
7233     }
7234   }
7235   \stex_reactivate_macro:N \varbind
7236   \stex_reactivate_macro:N \conclusion
7237   \stex_reactivate_macro:N \premise
7238   \stex_reactivate_macro:N \definiendum

```

```

7239 \stex_reactivate_macro:N \defnotation
7240 \stex_reactivate_macro:N \definame
7241 \stex_reactivate_macro:N \Definame
7242 }

```

sexample (*env.*)

```

7243 \stex_new_statement:nnn{example}{ex}{
7244   \stex_if_smsmode:F {\_stex_statements_setup:n{example}}
7245 }

```

sparagraph (*env.*)

```

7246 \stex_new_statement:nnn{paragraph}{}{
7247   \clist_if_in:NnTF \l_stex_key_style_clist {symdoc}{
7248     \_stex_statements_force_id:
7249     \_stex_statements_setup:n{
7250       \_stex_statements_setup_def:
7251     }{
7252       \_stex_statements_setup:n{
7253     }
7254 }

```

definiens

```

7255 \NewDocumentCommand \definiens { 0{} m }{
7256   \group_begin:
7257   \_stex_definiens_impl:nn{#1}{#2}
7258   \group_end:
7259   \stex_smsmode_do:
7260 }
7261
7262 \cs_new_protected:Nn \_stex_definiens_impl:nn {
7263   \tl_if_empty:nF {#1} {
7264     \stex_get_symbol:n { #1 }
7265     \tl_set_eq:NN \l_stex_current_def_uri \l_stex_get_symbol_uri
7266   }
7267   \tl_if_empty:NT \l_stex_current_def_uri {
7268     \msg_error:nn{stex}{error/definiensfor}
7269   }
7270   \stex_debug:nn{definiens}{Checking-\stex_use_symbol_uri:N \l_stex_current_def_uri }
7271
7272   \exp_args:No \_stex_add_definiens:nn \l_stex_current_def_uri {#2}
7273 }
7274
7275 \stex_deactivate_macro:Nn \definiens {definition~environments}
7276 \stex_sms_allow_escape:N \definiens

```

(End of definition for definiens. This function is documented on page 149.)

sfunction

_stex_add_definiens:nn

```

7277 \cs_new_protected:Nn \_stex_add_definiens:nn {
7278   \stex_if_do_html:TF {
7279     \stex_annotate:nn{data-ftml-definiens={\stex_use_symbol_uri:n{#1}}}{
7280       #2 %\_stex_annotate_force_break:n{ #2 }
7281     }

```

```

7282   }{ \stex_if_smsmode:F{#2} }
7283   \stex_if_in_module:T {
7284     \exp_args:Ne \str_if_eq:noT {\stex_symbol_uri_module:n{#1}}\l_stex_current_module_uri{
7285       \__stex_statements_add_definiens:n{#1}
7286     }
7287   }
7288 }
7289
7290 \cs_new_protected:Nn \__stex_statements_add_definiens:n {
7291   \stex_debug:nn{definiens}{Adding~definiens~to~\stex_use_symbol_uri:n{#1}}
7292   \exp_args:Nne \prop_gput:cne{\exp_args:Ne \stex_symbol_macro:n {\stex_symbol_uri_module:
7293     {\stex_symbol_uri_name:n {#1}} {
7294       \exp_args:Nne \use:nn { \__stex_statements_definiens_inner:nnnnnnN }{ \exp_args:Nne
7295     }
7296   }
7297
7298   \cs_new:Nn \__stex_statements_definiens_inner:nnnnnnN {
7299     {#1}{#2}{#3}{#4}{defed}{#6}{#7}{#8}
7300   }

```

(End of definition for `\stex_add_definiens:nn`. This function is documented on page 149.)

sfunction

`\varbind`

```

7301 \NewDocumentCommand \varbind {m} {
7302   \clist_map_inline:nn {#1} {
7303     \stex_get_var:n {##1}
7304     \stex_if_do_html:T {
7305       \stex_annotate_invisible:nn {data-ftml-bind=\l_stex_get_variable_str}{
7306     }
7307   }
7308 }
7309 \stex_deactivate_macro:Nn \varbind {definition~or~assertion~environments}

```

(End of definition for `\varbind`. This function is documented on page 149.)

sfunction

`\conclusion`

```

7310 \NewDocumentCommand \conclusion { 0{ } m} {
7311   \group_begin:
7312   \tl_if_empty:nF {#1} {
7313     \stex_get_symbol:n { #1 }
7314     \tl_set_eq:NN \l_stex_current_def_uri \l_stex_get_symbol_uri
7315   }
7316   \tl_if_empty:NT \l_stex_current_def_uri {
7317     \msg_error:nn{stex}{error/conclusionfor}
7318   }
7319   \stex_annotate:nn{ data-ftml-conclusion={\stex_use_symbol_uri:N \l_stex_current_def_uri}}
7320     #2 %\stex_annotate_force_break:n{ #2 }
7321   }
7322   \group_end:
7323 }
7324 \stex_deactivate_macro:Nn \conclusion {assertion~environments}

```

(End of definition for `\conclusion`. This function is documented on page 149.)
`sfunction`

`\premise`

```

7325 \NewDocumentCommand \premise {0{} m} {
7326   \tl_if_empty:nF {#1} {
7327     \stex_debug:nn{Here:}{Variable~#1}
7328     \exp_args:Nne\use:nn{\vardef}{\v#1}[name=#1]{#1}}
7329   }
7330   \stex_annotate:nn{data-ftml-premise={#1}}{#2}
7331 }
7332 \stex_deactivate_macro:Nn \premise {assertion~environments}

```

(End of definition for `\premise`. This function is documented on page 149.)
`sfunction`

Chapter 36

Proofs

```

7333 <@@=stex_proof>

Keys:
7334 \stex_keys_define:nnnn{ spf }{
7335   \tl_clear:N \l_stex_key_for_clist
7336   \tl_clear:N \l_stex_key_from_tl
7337   \tl_set_eq:NN \l_stex_key_proofend_tl \_stex_proof_proof_box_tl
7338   \tl_clear:N \l_stex_key_continues_tl
7339   \tl_clear:N \l_stex_key_term_tl
7340   \tl_clear:N \l_stex_key_functions_tl
7341   \tl_clear:N \l_stex_key_method_tl
7342   \bool_set_false:N \l_stex_key_hide_bool
7343 }{
7344   for      .clist_set:N = \l_stex_key_for_clist ,
7345   from      .tl_set:N   = \l_stex_key_from_tl ,
7346   proofend  .tl_set:N   = \l_stex_key_proofend_tl,
7347   continues .tl_set:N   = \l_stex_key_continues_tl,
7348   functions .tl_set:N   = \l_stex_key_functions_tl,
7349   term      .tl_set:N   = \l_stex_key_term_tl,
7350   method    .tl_set:N   = \l_stex_key_method_tl,
7351   hide      .bool_set:N = \l_stex_key_hide_bool
7352 }{id,style,title}
7353
7354 \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.

```

7355 \intarray_new:Nn\l_stex_proof_counter_intarray{50}
7356
7357 \cs_new_protected:Npn \_stex_proof_insert_number: {
7358   \int_set:Nn \l_tmpa_int {1}
7359   \bool_while_do:nn {
7360     \int_compare_p:nNn {

```

```

7361     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7362   } > 0
7363   ){
7364     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .
7365     \int_incr:N \l_tmpa_int
7366   }
7367 }
7368 \cs_new_protected:Nn \__stex_proof_number_as_string:N {
7369   \str_clear:N #1
7370   \int_set:Nn \l_tmpa_int {1}
7371   \bool_while_do:nn {
7372     \int_compare_p:nNn {
7373       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7374     } > 0
7375   ){
7376     \str_put_right:Nx #1 {\intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int .}
7377     \int_incr:N \l_tmpa_int
7378   }
7379 }
7380
7381 \cs_new_protected:Npn \__stex_proof_inc_counter: {
7382   \int_set:Nn \l_tmpa_int {1}
7383   \bool_while_do:nn {
7384     \int_compare_p:nNn {
7385       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7386     } > 0
7387   ){
7388     \int_incr:N \l_tmpa_int
7389   }
7390   \int_compare:nNnF \l_tmpa_int = 1 {
7391     \int_decr:N \l_tmpa_int
7392   }
7393   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int {
7394     \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int + 1
7395   }
7396 }
7397
7398 \cs_new_protected:Npn \__stex_proof_add_counter: {
7399   \int_set:Nn \l_tmpa_int {1}
7400   \bool_while_do:nn {
7401     \int_compare_p:nNn {
7402       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int
7403     } > 0
7404   ){
7405     \int_incr:N \l_tmpa_int
7406   }
7407   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 1 }
7408 }
7409
7410 \cs_new_protected:Npn \__stex_proof_remove_counter: {
7411   \int_set:Nn \l_tmpa_int {1}
7412   \bool_while_do:nn {
7413     \int_compare_p:nNn {
7414       \intarray_item:Nn \l__stex_proof_counter_intarray \l_tmpa_int

```

```

7415     } > 0
7416   }{
7417     \int_incr:N \l_tmpa_int
7418   }
7419   \int_decr:N \l_tmpa_int
7420   \intarray_gset:Nnn \l__stex_proof_counter_intarray \l_tmpa_int { 0 }
7421 }

sproof (env.)

7422 \bool_set_false:N \l__stex_proof_in_spfblock_bool
7423
7424 \cs_new_protected:Nn \__stex_proof_begin_proof:nn {\par
7425   \intarray_gzero:N \l__stex_proof_counter_intarray
7426   \intarray_gset:Nnn \l__stex_proof_counter_intarray 1 1
7427   \stex_keys_set:nn{spf}{#1}
7428   \stex_do_for_list:
7429   \stex_if_do_html:T {
7430     \__stex_proof_html_env:n{proof}
7431   }
7432   \seq_map_inline:Nn \l_stex_fors_seq {
7433     \stex_debug:nn{definiens}{Adding-definiens~to~##1}
7434     \stex_if_do_html:TF{
7435       \STEXinvisible{\hbox{\stex_add_definiens:nn {##1}{proven}}}}
7436     }{
7437       \stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7438     }
7439   }
7440   \stex_if_do_html:F\stex_style_apply:
7441   %\stex_do_id:
7442   \stex_reactivate_macro:N \subproof
7443   \stex_reactivate_macro:N \spfstep
7444   \stex_reactivate_macro:N \conclude
7445   \stex_reactivate_macro:N \assumption
7446   \stex_reactivate_macro:N \eqstep
7447   \stex_reactivate_macro:N \yield
7448   \stex_reactivate_macro:N \spfblock
7449   \stex_reactivate_macro:N \spfjust
7450   \noindent\stex_annotate:nn{data-ftml-title={}}{\stex_annotate_force_break:n{#2}}\par
7451   \stex_if_do_html:TF{
7452     \use:c{stex_annotate_env}{data-ftml-proofbody={}}
7453     \stex_annotate_force_break:n{}}
7454   }{
7455     \bool_if:NT \l_stex_key_hide_bool{\setbox0\vbox\bgroup}
7456   }
7457 }

7458
7459 \cs_new_protected:Nn \__stex_proof_html_env:n {
7460   \exp_args:Nne \use:c{stex_annotate_env}{
7461     data-ftml-#1={
7462       \seq_if_empty:NF \l_stex_fors_seq {
7463         \seq_map_function:NN \l_stex_fors_seq \stex_use_symbol_uri:N
7464       }
7465     }
7466     \bool_if:NT \l_stex_key_hide_bool {,

```

```

7467     data-ftml-proofhide=true
7468   }
7469 }
7470 \__stex_proof_html:
7471 }
7472
7473 \cs_new_protected:Nn \__stex_proof_html: {
7474   \stex_annotate_force_break:n{}
7475   \tl_set:N\l_tmpa_tl {\l_stex_key_term_tl\l_stex_key_method_tl}
7476   \tl_if_empty:NF\l_tmpa_tl{
7477     \stex_annotate_invisible:n{\hbox{
7478       \tl_if_empty:NF \l_stex_key_term_tl {
7479         $\stex_annotate:nn{data-ftml-proofterm={}}{\l_stex_key_term_tl}$
7480       }
7481       \tl_if_empty:NF \l_stex_key_method_tl {
7482         \stex_annotate:nn{data-ftml-proofmethod={}}{\l_stex_key_method_tl}
7483       }
7484     }}
7485   }
7486 }
7487
7488 \cs_new_protected:Nn \__stex_proof_start_comment: {
7489   \par
7490   \bool_if:NT \l__stex_proof_in_spfblock_bool {
7491     \group_begin:\stexcommentfont
7492   }
7493 }
7494 \cs_new_protected:Nn \__stex_proof_end_comment: {
7495   \par
7496   \bool_if:NT \l__stex_proof_in_spfblock_bool {
7497     \group_end:
7498   }
7499 }
7500
7501 \stex_new_stylable_env:nnnnnn{proof}{0}{m}{
7502   \__stex_proof_begin_proof:nn{#1}{#2}
7503   \bool_set_true:N\l__stex_proof_in_spfblock_bool
7504   %\__stex_proof_start_list:n{}
7505   \__stex_proof_start_comment:
7506 }{
7507   \stex_if_do_html:TF{
7508     %\__stex_proof_end_list:
7509     \use:c{endstex_annotate_env}\use:c{endstex_annotate_env}
7510   }\stex_style_apply:
7511 }{
7512   \emph{\sproofautorefname :}~
7513 }{
7514   \sproofend
7515 }{s}
7516
7517 \AddToHook{env/sproof/end}{
7518   \__stex_proof_end_comment:
7519   \stex_if_do_html:F{\bool_if:NT \l_stex_key_hide_bool\egroup}
7520 }

```



```

7521
7522
7523 \stex_new_stylable_env:nnnnnnn{proof*}{0{}}{
7524   \_stex_proof_begin_proof:nn{#1}{}
7525   \bool_set_false:N\l__stex_proof_in_spfblock_bool
7526 }{
7527   \stex_if_do_html:TF{\use:c{endstex_annotate_env}\use:c{endstex_annotate_env}}{\stex_style_
7528 }{
7529   \emph{Proof:}~
7530 }{
7531   \sproofend
7532 }{s}
7533
7534 \cs_new_protected:Nn \_stex_proof_start_list:n {
7535   %\par
7536   \group_begin:\list{}{
7537     \setlength\topsep{0pt}
7538     \setlength\parsep{0pt}
7539     \setlength\rightmargin{0pt}
7540   }\item[#1]
7541 }
7542
7543 \cs_new_protected:Nn \_stex_proof_end_list: {
7544   %\par
7545   \endlist\group_end:
7546 }

```

subproof (*env*.)

```

7547 \str_set_eq:NN \subproofautorefname \spfstepautorefname
7548
7549 \stex_new_stylable_env:nnnnnnn{subproof}{s 0{ } m}{
7550   \stex_keys_set:nn{spf}{#2}
7551   \_stex_do_for_list:
7552   \stex_if_do_html:T {
7553     \_stex_proof_html_env:n{subproof}
7554   }
7555   \seq_map_inline:Nn \l_stex_fors_seq {
7556     \stex_debug:nn{definiens}{Adding~definiens~to~##1}
7557     \stex_if_do_html:TF{
7558       \STEXinvisible{\hbox{\_stex_add_definiens:nn {##1}{proven}}}
7559     }{
7560       \_stex_add_definiens:nn {##1}{\STEXinvisible{proven}}
7561     }
7562   }
7563
7564   \IfBooleanTF #1 {
7565     \stex_if_do_html:F \stex_style_apply:
7566     \str_if_empty:NF \l_stex_key_id_str {
7567       \_stex_proof_number_as_string:N \@currentlabel
7568       %\str_set:Ne \@currentHref{subproof.\@currentlabel}
7569       %\_stex_do_id:
7570     }
7571     \bool_set_false:N \l__stex_proof_in_spfblock_bool
7572     \stex_annotate:nn{data-ftml-title={}}{#3}

```

```

7573   }{
7574     \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7575       %\str_if_empty:NF \l_stex_key_id_str {
7576         %\__stex_proof_number_as_string:N \@currentlabel
7577         %\str_set:Ne \@currentHref{subproof.\@currentlabel}
7578         %\__stex_do_id:
7579       %}
7580       \use:c{stex_annotate_env}{data-ftml-title={}} \__stex_proof_start_list:n\__stex_proof
7581       \__stex_proof_add_counter:
7582       \stex_if_do_html:F \stex_style_apply:
7583     }{
7584       \noindent \stex_annotate:nn{data-ftml-title={}}{#3} \par
7585       \stex_if_do_html:F \stex_style_apply:
7586       %\__stex_do_id:
7587     }
7588   }
7589   \stex_if_do_html:TF{
7590     \use:c{stex_annotate_env}{data-ftml-proofbody={}}
7591     \__stex_annotate_force_break:n{}
7592   }{\bool_if:NT \l_stex_key_hide_bool{\setbox0\vbox\bgroup}}
7593   \__stex_proof_start_comment:
7594 }{
7595   \bool_if:NT\l__stex_proof_in_spfblock_bool {
7596     \__stex_proof_remove_counter:
7597     \__stex_proof_inc_counter:
7598   }
7599
7600   \stex_if_do_html:TF{
7601     \use:c{endstex_annotate_env} %proofbody
7602     \bool_if:NT\l__stex_proof_in_spfblock_bool \__stex_proof_end_list:
7603     \use:c{endstex_annotate_env} % subproof
7604   }{
7605     \stex_style_apply:
7606     \bool_if:NT \l__stex_proof_in_spfblock_bool {\__stex_proof_inc_counter:\__stex_proof_en
7607   }
7608 }{}{}{}
7609
7610 \AddToHook{env/subproof/before}{
7611   \__stex_proof_end_comment:
7612 }
7613
7614 \AddToHook{env/subproof/after}{
7615   \__stex_proof_start_comment:
7616 }
7617
7618 \AddToHook{env/subproof/end}{
7619   \__stex_proof_end_comment:
7620   \stex_if_do_html:F{\bool_if:NT \l_stex_key_hide_bool\egroup}
7621 }
7622
7623 \stex_deactivate_macro:Nn \subproof {sproof~environments}
7624

```

spfsketchenv (env.) TODO:

```

7625 \newenvironment{spfsketchenv}{}{}

\spfsketch

7626 \stex_new_stylable_cmd:nnnn{spfsketch}{0{ } m}{\par
7627   \begin{spfsketchenv}
7628   \stex_keys_set:nn{spf}{#1}
7629   \_stex_do_for_list:
7630   \_stex_do_id:
7631   \par
7632   \exp_args:Nne \use:c{stex_annotate_env}{
7633     data-ftml-proofs sketch={
7634       \seq_if_empty:NF \l_stex_fors_seq {
7635         \seq_map_function:NN \l_stex_fors_seq \stex_use_symbol_uri:N
7636       }
7637     }
7638   }
7639   \stex_style_apply:
7640   #2\par
7641   \use:c{endstex_annotate_env}
7642 \end{spfsketchenv}
7643 }{
7644   \noindent\emph{\spfsketchenvautorefname :}~
7645 }

```

(End of definition for \spfsketch. This function is documented on page 150.)

sfunction

Keys for proof step macros:

```

7646 \stex_keys_define:nnnn { spfsteps } {
7647   \clist_clear:N \l_stex_key_for_clist
7648   \str_clear:N \l_stex_key_name_str
7649   \tl_clear:N \l_stex_key_method_tl
7650   \tl_clear:N \l_stex_key_term_tl
7651 }{
7652   for      .clist_set:N = \l_stex_key_for_clist ,
7653   method   .tl_set:N    = \l_stex_key_method_tl,
7654   term      .tl_set:N    = \l_stex_key_term_tl,
7655   name      .str_set_x:N = \l_stex_key_name_str
7656   % todo: style=inline
7657 }{id,style,title}

```

Common setup for all the proof step macros:

```

7658 \cs_new_protected:Nn \_stex_proof_make_step_macro:Nnnnn {
7659   \NewDocumentCommand #1 {s 0{ } +m} {
7660     \_stex_proof_end_comment:
7661     \stex_keys_set:nn{spfsteps}{##2}
7662     \str_if_empty:NF \l_stex_key_name_str {
7663       \exp_args:Nne\use:nn{\vardef}{{v\l_stex_key_name_str}[name=\l_stex_key_name_str]{\l_s
7664     }
7665
7666     \spfstepenv
7667     \str_if_empty:NF \l_stex_key_id_str {
7668       %\_stex_proof_number_as_string:N \@currentlabel
7669       %\str_set:Ne \@currentHref{spfstep.\@currentlabel}
7670       %\_stex_do_id:

```

```

7671 }
7672
7673 \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7674   \IfBooleanTF ##1 {
7675     \__stex_proof_step_html:nn{#2}{##3}
7676   }{
7677     \__stex_proof_step_html:nn{#2}{\__stex_proof_start_list:n{#3} ##3 \__stex_proof_end
7678       #5
7679   }
7680   \endspfststepenv
7681   \__stex_proof_start_comment:
7682 }{
7683   \__stex_proof_step_html:nn{#2}{##3}
7684   \endspfststepenv
7685 }
7686 }
7687 \stex_deactivate_macro:Nn #1 {sproof~environments}
7688 }
7689
7690 \cs_new_protected:Nn \__stex_proof_step_html:nn {
7691   \stex_if_do_html:TF{
7692     %\group_begin:
7693     \exp_args:Nne \use:c{stex_annotate_env}{
7694       data-ftml-spf#1={
7695         \seq_if_empty:NF \l_stex_fors_seq {
7696           \seq_map_function:NN \l_stex_fors_seq \stex_use_symbol_uri:N
7697         }
7698       }
7699       \str_if_empty:NF \l_stex_key_name_str {,
7700         data-ftml-stepname={\l_stex_key_name_str}
7701       }
7702     }
7703     \__stex_proof_html:
7704     #2
7705     \par
7706     \use:c{endstex_annotate_env}
7707     %\group_end:
7708   }{ #2 }
7709 }

```

spfststepenv (*env.*) TODO:

```

7710 \newenvironment{spfststepenv}{
7711   \str_set_eq:NN \spfststepenvautorefname \spfststepautorefname
7712 }{}

```

spfblock (*env.*)

```

7713 \NewDocumentEnvironment{spfblock}{}{
7714   \bool_set_false:N \l__stex_proof_in_spfblock_bool
7715 }{
7716   \aftergroup\__stex_proof_start_comment:
7717 }
7718
7719 \stex_deactivate_macro:Nn \spfblock {sproof~environments}
7720 \AddToHook{env/spfblock/before}{

```

```

7721   \__stex_proof_end_comment:
7722 }

\spfstep
7723 \__stex_proof_make_step_macro:Nnnnn \spfstep {step} \__stex_proof_insert_number: {} \__stex
(End of definition for \spfstep. This function is documented on page 150.)
sfunction

\assumption
7724 \__stex_proof_make_step_macro:Nnnnn \assumption {assumption} \__stex_proof_insert_number: {}
(End of definition for \assumption. This function is documented on page 150.)
sfunction

\conclude
7725 \__stex_proof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
(End of definition for \conclude. This function is documented on page 150.)
sfunction

\eqstep
7726 \NewDocumentCommand \eqstep {s m}{
7727   \__stex_proof_end_comment:
7728   \spfstepenv
7729
7730   \bool_if:NTF \l__stex_proof_in_spfblock_bool {
7731     \IfBooleanTF #1 {
7732       \__stex_proof_step_html:nn{eqstep}{\$= #2$}
7733     }{
7734       \__stex_proof_step_html:nn{eqstep}{\__stex_proof_start_list:n{\$= $} $#2$ \__stex_proof
7735     }
7736     \endspfstepenv
7737     \__stex_proof_start_comment:
7738   }{
7739     \__stex_proof_step_html:nn{eqstep}{\$= #2$}
7740     \endspfstepenv
7741   }
7742 }
7743 \stex_deactivate_macro:Nn \eqstep {sproof~environments}

(End of definition for \eqstep. This function is documented on page 150.)
sfunction

\yield
7744 \NewDocumentCommand \yield {+m}{
7745   \stex_annotate:nn{data-ftml-proofterm={}}{ #1 }
7746 }
7747 \stex_deactivate_macro:Nn \yield {sproof~environments}

(End of definition for \yield. This function is documented on page 150.)
sfunction

```

`\spfjust`

```
7748 \newcommand\spfjust[1]{
7749   \stex_annotate:nn{spfjust={}}{ #1 }
7750 }
7751 \stex_deactivate_macro:Nn \spfjust {sproof-environments}
```

(End of definition for \spfjust. This function is documented on page 150.)
sfunction

`\sproofend`

```
7752 \tl_set:Nn \__stex_proof_proof_box_tl {
7753   \ltx@ifpackageloaded{amssymb}{\square$}{
7754     \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
7755   }
7756 }
7757
7758 \tl_set:Nn \sproofend {
7759   \tl_if_empty:NF \l_stex_key_proofend_tl {
7760     \hfil\null\nobreak\hfill\l_stex_key_proofend_tl\par\smallskip
7761   }
7762 }
```

(End of definition for \sproofend. This function is documented on page 150.)
sfunction

`\stexcommentfont`

```
7763 \cs_new_protected:Npn \stexcommentfont {
7764   \small\itshape
7765 }
```

(End of definition for \stexcommentfont. This function is documented on page 151.)
sfunction

Metatheory

335

```

7802 \symdecl*{record~type}
7803
7804 \symdecl{mathstruct}[name=mathematical~structure,args=a] % TODO
7805 \notation{mathstruct}[angle,prec=nobrackets]
7806   {\mathopen{\comp\langle} #1 \mathclose{\comp\rangle}}
7807 \notation{mathstruct}[parens,prec=nobrackets]
7808   {\mathopen{\comp{}} #1 \mathclose{\comp{}}}
7809
7810 % sequences
7811 \symdef{ellipses}[ldots]{\ldots}
7812 \symdef{sequence-expression}[comma,args=a]{#1}
7813 \symdef{sequence-type}[args=1]{#1^{\comp\ast}}
7814 \symdef{sequence-map}[args=ia]{
7815   \comp{\mathrm{map}}\mathopen{\comp{}}#1\mathpunct{\comp{,}}
7816   #2\mathclose{\comp{}}
7817 }
7818
7819 \symdecl{bind}[args=Bi,assoc=pre]
7820 \notation{bind}[defun,prec=nobrackets,op=(\cdot)\;\to\;\cdot]
7821   {\mathopen{\comp{}} #1 \mathclose{\comp{}}\mathbin{\comp{\to}}} #2}
7822 \notation{bind}[forall]{\comp\forall #1.\;#2}
7823 \notation{bind}[Pi]{\mathop{\comp\prod}\c_math_subscript_token{#1}#2}
7824
7825 \symdef{implicit-bind}[args=Bi,assoc=pre]{\mathopen{\comp\{}} #1 \mathclose{\comp{\}}\c_math_
7826
7827 \symdecl*{integer-literal}
7828 \notation{integer-literal}{\mathbb Z}
7829
7830 \symdecl*{ordinal}
7831 \notation{ordinal}{\mathtt{Ord}}
7832
7833 % propositions
7834 \symdef{prop}[name=proposition]{\mathtt{Prop}}
7835 \symdef{judgment-holds}[args=i,role=judgment]{\comp\vdash\;#1}
7836
7837 % any object
7838 \symdef{object}{\mathtt{Obj}}
7839
7840 % TODO DELETE
7841 \symdef{aseqdots}[args=a,prec=nobrackets]
7842   {#1\comp{,\ldots}}%{##1\comp,##2}
7843 \symdef{aseqfromto}[args=ai,prec=nobrackets]
7844   {#1\comp{,\ldots,}#2}%{##1\comp,##2}
7845 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]
7846   {#1\comp{,\ldots,}#2\comp{,\ldots,}#3}%{##1\comp,##2}

Closing:
7847 \global \let \l_stex_metatheory_uri \l_stex_current_module_uri
7848 \global \let \c_stex_default_metatheory \l_stex_metatheory_uri
7849 \stex_close_module:
7850 \group_end:

```


Chapter 38

Others

```
7851 <@@=stex_others>
7852
7853 \cs_new_protected:Npn \MSC #1 {}
7854
7855 \_stex_persist_read_now:
7856 \cs_new_protected:Nn \_stex_others_newlabel:n {
7857   \tl_gset:cn{\_stex_sref_aux_sym_ \tl_to_str:n{#1}}{X}
7858   \exp_args:Ne\_stex_others_old_newlabel:{\tl_to_str:n{#1}}
7859 }
7860 \AtBeginDocument{
7861   \iow_now:Nn \@auxout {
7862     \ExplSyntaxOn
7863     \let\_stex_others_old_newlabel:\newlabel
7864     \let\newlabel\_stex_others_newlabel:n
7865     \ExplSyntaxOff
7866   }
7867 }
```

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.

```
7868 \*cls)
7869 @@=notesslides)
7870 \ProvidesExplClass{notesslides}{2025/11/11}{4.0.0}{notesslides Class}
7871 \RequirePackage{l3keys2e}
7872
7873 \str_const:Nn \c__notesslides_class_str {article}
7874
7875 \keys_define:nn{notesslides / cls}{
7876   class .str_set_x:N = \c__notesslides_class_str,
7877   notes .bool_set:N = \c__notesslides_notes_bool ,
7878   slides .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7879   %docopt .str_set_x:N = \c__notesslides_docopt_str,
7880   unknown .code:n      = {
7881     \PassOptionsToClass{\CurrentOption}{beamer}
7882     \PassOptionsToClass{\CurrentOption}{\c__notesslides_class_str}
7883     \PassOptionsToPackage{\CurrentOption}{notesslides}
7884     \PassOptionsToPackage{\CurrentOption}{stex}
7885   }
7886 }
7887 \ProcessKeysOptions{ notesslides / cls }
7888
7889 \RequirePackage{stex}
7890 \stex_if_html_backend:T {
7891   \bool_set_true:N \c__notesslides_notes_bool
7892 }
7893
7894 \bool_if:NTF \c__notesslides_notes_bool {
7895   \PassOptionsToPackage{notes=true}{notesslides}
7896   \message{notesslides.cls:~Formatting~document~in~notes~mode}
```

```

7897 }{
7898   \PassOptionsToPackage{notes=false}{notesslides}
7899   \message{notesslides.cls:~Formatting~document~in~slides~mode}
7900 }
7901
7902 \bool_if:NTF \c_notesslides_notes_bool {
7903   \LoadClass{\c_notesslides_class_str}
7904 }{
7905   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7906   %\newcounter{Item}
7907   %\newcounter{paragraph}
7908   %\newcounter{subparagraph}
7909   %\newcounter{Hfootnote}
7910 }
7911 \RequirePackage{notesslides}
7912 </cls>

```

now we do the same for the notesslides package.

```

7913 *package>
7914 \ProvidesExplPackage{notesslides}{2025/11/11}{4.0.0}{notesslides Package}
7915 \RequirePackage{l3keys2e}
7916
7917 \keys_define:nn{notesslides / pkg}{
7918   notes .bool_set:N = \c_notesslides_notes_bool ,
7919   slides .code:n = { \bool_set_false:N \c_notesslides_notes_bool },
7920   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
7921   topsect .str_set_x:N = \c_notesslides_topsect_str,
7922   unknown .code:n = {
7923     \PassOptionsToPackage{\CurrentOption}{stex}
7924     \PassOptionsToPackage{\CurrentOption}{tikzinput}
7925   }
7926 }
7927 \ProcessKeysOptions{ notesslides / pkg }
7928
7929 \RequirePackage{stex}
7930 \stex_if_html_backend:T {
7931   \bool_set_true:N \c_notesslides_notes_bool
7932 }
7933
7934 \cs_set:Npn \sectiontitleemph #1 {
7935   \textbf{\Large #1}
7936 }
7937
7938 \newif\ifnotes
7939 \bool_if:NTF \c_notesslides_notes_bool {
7940   \notesttrue
7941   \PassOptionsToPackage{dvipsnames,svgnames}{xcolor}
7942   \RequirePackage[noamsthm,hyperref]{beamerarticle}
7943   \RequirePackage{mdframed}
7944   \str_if_empty:NTF \c_notesslides_topsect_str{
7945     %\setsectionlevel{section}
7946   } {
7947     \exp_args:No \setsectionlevel \c_notesslides_topsect_str
7948   }

```

```

7949 }{
7950   \notesfalse
7951
7952   \cs_new_protected:Nn \__notesslides_do_sectocframes: {
7953     \cs_set_protected:Nn \__notesslides_do_label:n {
7954       \str_case:nnF{##1}{
7955         {part} {
7956           \tl_set:Nx\l__notesslides_num{\thepart}
7957           \tl_set:cx{@ @ label}{
7958             \cs_if_exist:NTF\parttitlename{\exp_not:N\parttitlename}{\exp_not:N\partname}{
7959             }
7960           {chapter} {
7961             \tl_set:Nx\l__notesslides_num{\thechapter}
7962             \tl_set:cx{@ @ label}{
7963               \cs_if_exist:NTF\chaptertitlename{\exp_not:N\chaptertitlename}{\exp_not:N\chapter
7964             }
7965           {section} {
7966             \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection
7967             \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7968           }
7969         {subsection} {
7970           \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection
7971           \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7972         }
7973       {subsubsection} {
7974         \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection
7975         \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7976       }
7977     {paragraph} {
7978       \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection
7979       \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7980     }
7981   }{
7982     \tl_set:Nx\l__notesslides_num{\cs_if_exist:NT\thechapter{\thechapter.}\thesection.\
7983     \tl_set:cx{@ @ label}{\l__notesslides_num\quad}
7984   }
7985 }
7986 \cs_set_protected:Nn \_sfragment_do_level:nn {
7987   \tl_if_exist:cT{c##1}{\stepcounter{##1}}
7988   \addcontentsline{toc}{##1}{\protect\numberline{\use:c{the##1}}##2}
7989   \__notesslides_do_label:n{##1}
7990   \pdfbookmark[\int_use:N \l_stex_current_section_level_int]{\l__notesslides_num\ ##2}{
7991   \begin{frame}[noframenumbering]
7992     \vfill\centering
7993     \sectiontitleemph{
7994       \use:c{@ @ label} ##2
7995     }
7996   \end{frame}
7997   \int_incr:N \l_stex_current_section_level_int
7998   \str_set:Nn \l_stex_current_section_level_str{##1}
7999 }
8000 }
8001
8002 \cs_set_protected:Nn \_stex_titlefragment: {

```

```

8003     \begin{frame}[noframenumbering]\maketitle\end{frame}
8004     \let\maketitle\relax
8005 }
8006
8007 \AtBeginDocument{
8008     \str_if_empty:NTF \c_notesslides_topsect_str {
8009         \setsectionlevel{section}
8010     } {
8011         \exp_args:No \setsectionlevel \c_notesslides_topsect_str
8012         \exp_args:No \str_if_eq:nnTF \c_notesslides_topsect_str {chapter} {
8013             \__notesslides_define_chapter:
8014         }{
8015             \exp_args:No \str_if_eq:nnT \c_notesslides_topsect_str {part} {
8016                 \__notesslides_define_chapter:
8017                 \__notesslides_define_part:
8018             }
8019         }
8020     }
8021 }
8022
8023 \bool_if:NT \c_notesslides_sectocframes_bool {
8024     \__notesslides_do_sectocframes:
8025 }
8026 }
8027
8028 \cs_new_protected:Nn \__notesslides_define_chapter: {
8029     \cs_if_exist:NF \chaptername {
8030         \cs_set_protected:Npn \chaptername {Chapter}
8031     }
8032     \cs_if_exist:NF \chapter {
8033         \cs_set_protected:Npn \chapter {INVALID}
8034     }
8035     \cs_if_exist:NF \c@chapter {
8036         \newcounter{chapter}\counterwithin*{section}{chapter}
8037     }
8038 }
8039
8040 \cs_new_protected:Nn \__notesslides_define_part: {
8041     \cs_if_exist:NF \partname {
8042         \cs_set_protected:Npn \partname {Part}
8043     }
8044     \cs_if_exist:NF \part {
8045         \cs_set_protected:Npn \part {INVALID}
8046     }
8047     \cs_if_exist:NF \c@part {
8048         \newcounter{part}\counterwithin*{chapter}{part}
8049     }
8050 }

```

\prematuarestop We initialize \afterprematuarestop, and provide \prematuarestop@endsfragment which looks up \sfragment@level and recursively ends enough {sfragment}s.

```

8051 \def \c__notesslides_document_str{document}
8052 \newcommand\afterprematuarestop{}
8053 \def\prematuarestop@endsfragment{

```

```

8054 \unless\ifx\@currenvir\c__notesslides_document_str
8055 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
8056 \expandafter\prematurestop@endsfragment
8057 \fi
8058 }
8059 \providecommand\prematurestop{
8060 \stex_if_html_backend:F{
8061 \message{Stopping~sTeX~processing~prematurely}
8062 \prematurestop@endsfragment
8063 \afterprematurestop
8064 \end{document}}
8065 }
8066 }

```

(End of definition for \prematurestop. This function is documented on page 103.)

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.

```

8067 \bool_if:NT \c_notesslides_notes_bool {
8068 \renewcommand\usetheme[2][\usepackage[#1]{beamertheme#2}]
8069 }
8070 \NewDocumentCommand \libusetheme {O{} m} {
8071 \libusepackage[#1]{beamertheme#2}
8072 }

```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```

8073 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
8074 \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.

```

8075 \ifnotes
8076
8077 \newlength{\slideframewidth}
8078 \setlength{\slideframewidth}{1.5pt}

```

frame (env.) We first define the keys.

```

8079 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
8080 \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
8081 \bool_set_true:N #1
8082 }{
8083 \bool_set_false:N #1
8084 }
8085 }
8086
8087 \stex_keys_define:nnnn{notesslides / frame}{
8088 \str_clear:N \l__notesslides_frame_label_str
8089 \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
8090 \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
8091 \bool_set_true:N \l__notesslides_frame_fragile_bool
8092 \bool_set_true:N \l__notesslides_frame_shrink_bool

```

```

8093 \bool_set_true:N \l__notesslides_frame_squeeze_bool
8094 \bool_set_true:N \l__notesslides_frame_t_bool
8095 }{
8096   label .str_set_x:N = \l__notesslides_frame_label_str,
8097   allowframebreaks .code:n = {
8098     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
8099   },
8100   allowdisplaybreaks .code:n = {
8101     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
8102   },
8103   fragile .code:n = {
8104     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
8105   },
8106   shrink .code:n = {
8107     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
8108   },
8109   squeeze .code:n = {
8110     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
8111   },
8112   t .code:n = {
8113     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
8114   },
8115   unknown .code:n = {}
8116 }{}

```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```

8117 \cs_new_protected:Nn \__notesslides_setup_itemize: {
8118   \def\itemize@level{outer}
8119   \def\itemize@outer{outer}
8120   \def\itemize@inner{inner}
8121   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
8122   \renewenvironment{itemize}{
8123     \ifx\itemize@level\itemize@outer
8124       \def\itemize@label{$\rhd$}
8125     \fi
8126     \ifx\itemize@level\itemize@inner
8127       \def\itemize@label{$\scriptstyle\rhd$}
8128     \fi
8129     \begin{list}
8130     {\itemize@label}
8131     {\setlength{\labelsep}{.3em}
8132      \setlength{\labelwidth}{.5em}
8133      \setlength{\leftmargin}{1.5em}
8134     }
8135     \edef\itemize@level{\itemize@inner}
8136   }{
8137     \end{list}
8138   }
8139 }

```

We create the box with the `mdframed` environment from the `equinymous` package.

```

8140 \stex_if_html_backend:TF {
8141   %\stex_css_literal:n{
8142   % .stex-frame {

```

```

8143 % background-color:#fff;
8144 % margin:5px ~ 0;
8145 % border:2px ~ solid ~ #000;
8146 % border-radius:5px;
8147 % padding:5px;
8148 % padding-right:10px;
8149 % width: var(--rustex-curr-width);
8150 % display:block;
8151 % }
8152 %}

8153
8154 \cs_new_protected:Nn \__notesslides_frame_box_begin: {
8155   \vbox\bgroup
8156   \begin{stex_annotate_env}{data-ftml-slide={}}%{class:stex-frame={}}
8157   \mdf@patchamsthm\notesslidesfont
8158 }
8159 \cs_new_protected:Nn \__notesslides_frame_box_end: {
8160   %^A \notesslides@slidelabel
8161   \medskip\par\hbox to \textwidth{\tiny\notesslidesfooter}
8162   \end{stex_annotate_env}\egroup
8163 }
8164 }{
8165   \cs_new_protected:Nn \__notesslides_frame_box_begin: {
8166     \begin{mdframed}[
8167       linewidth=\slideframewidth,
8168       skipabove=1ex,
8169       skipbelow=1ex,
8170       userdefinedwidth=\slidewidth,
8171       align=center,
8172       %usetwoside=false
8173     ]\notesslidesfont
8174   }
8175   \cs_new_protected:Nn \__notesslides_frame_box_end: {
8176     \medskip\par\noindent\tiny\notesslidesfooter%^A\notesslides@slidelabel
8177     \end{mdframed}
8178   }
8179 }

```

We define the environment, read them, and construct the slide number and label.

```

8180 \renewenvironment{frame}[1][]{
8181   \stex_keys_set:nn{notesslides / frame}{#1}
8182   \stepcounter{framenum}
8183   \renewcommand\newpage{\addtocounter{framenum}{1}}
8184   \def\@currentlabel{\theframenum}
8185   \str_if_empty:NF \l__notesslides_frame_label_str {
8186     \label{\l__notesslides_frame_label_str}
8187   }
8188   \__notesslides_setup_itemize:
8189   \__notesslides_frame_box_begin:
8190 }{
8191   \__notesslides_frame_box_end:
8192 }

```

Now, we need to redefine the frametitle (we are still in course notes mode).


```

8239 % \rustex_if:T {\par
8240 % \rustex_direct_HTML:n{</td><td>}
8241 % }
8242 % }
8243 % }
8244 \end{lrbox}\usebox\columnbox
8245 }
8246 \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.

```

8247 \bool_if:NTF \c_notesslides_notes_bool {
8248 \renewenvironment{note}{\ignorespaces}{}
8249 }{
8250 \renewenvironment{note}{\setbox \l_tmpa_box\vbox\bgroup}{\egroup}
8251 }

```

For other environments we introduce variants prefixed with `n`, which are excluded in `slides` mode.

```

8252 \cs_new_protected:Nn \__notesslides_notes_env:nnnn {
8253 \bool_if:NTF \c_notesslides_notes_bool {
8254 \newenvironment{#1}#2{#3}{#4}
8255 }{
8256 \newenvironment{#1}#2{
8257 \cs_set:Npn \__notesslides_eat: #####1 \end #####2 {
8258 \str_if_eq:nnTF{#1}{#####2}{
8259 \end{#1}
8260 }{
8261 \__notesslides_eat:
8262 }
8263 }
8264 \__notesslides_eat:
8265 %\setbox\l_tmpa_box\vbox\bgroup#3
8266 }{
8267 %#4\egroup
8268 }
8269 }
8270 }
8271
8272 \__notesslides_notes_env:nnnn{nparagraph}{[1] []}{\begin{sparagraph}[#1]}\end{sparagraph}}
8273 \__notesslides_notes_env:nnnn{nfragment}{[2] []}{\begin{sfragment}[#1]{#2}}\end{sfragment}}
8274 \__notesslides_notes_env:nnnn{ndefinition}{[1] []}{\begin{sdefinition}[#1]}\end{sdefinition}}
8275 \__notesslides_notes_env:nnnn{nassertion}{[1] []}{\begin{sassertion}[#1]}\end{sassertion}}
8276 \__notesslides_notes_env:nnnn{nproof}{[2] []}{\begin{sproof}[#1]{#2}}\end{sproof}}
8277 \__notesslides_notes_env:nnnn{nexample}{[1] []}{\begin{sexample}[#1]}\end{sexample}}
8278
8279 \RequirePackage{graphicx}
8280
8281 \NewDocumentCommand\frameimage{s O{} m}{
8282 \IfBooleanTF #1 {

```

```

8283 \begin{frame}[plain]
8284 }{
8285 \begin{frame}
8286 }
8287 \bool_if:NTF \c_notesslides_notes_bool {
8288 \slidewidth=\dimexpr\slidewidth-(2\slideframewidth)\relax
8289 }{
8290 \slidewidth=\textwidth\relax
8291 }
8292 \def\Gin@ewidth{}\setkeys{Gin}{#2}
8293 \tl_if_empty:NTF \Gin@ewidth {
8294 \mhgraphics[width=\slidewidth,#2]{#3}
8295 }{
8296 \mhgraphics[#2]{#3}
8297 }
8298 \end{frame}
8299 }

```

hacking inputref:

`\inputref*`

```

8300 \cs_set_eq:NN\__notesslides_inputref:\inputref
8301 \cs_set_protected:Npn\inputref{\@ifstar\ninputref\__notesslides_inputref:}
8302 \bool_if:NTF \c_notesslides_notes_bool {
8303 \newcommand\ninputref[2][]{
8304 \__notesslides_inputref:[#1]{#2}
8305 }
8306 }{
8307 \newcommand\ninputref[2][]{
8308 }

```

(End of definition for \inputref. This function is documented on page 102.)*

39.1.4 Styling Across Notes/Slides

```

8309 \def\notesslides_title_emph#1{
8310 {\Large\bf\sf#1}
8311 \vskip0.1\baselineskip
8312 \leaders\vrule width \textwidth
8313 \vskip0.4pt%
8314 \nointerlineskip
8315 }
8316
8317 \def\notesslides_footer{}
8318
8319 \let\notesslides_font\sfamily

```

39.1.5 Beamer Compatibility

All of this should be removed and made part of a template

```

8320
8321 \stex_if_do_html:TF{
8322 \NewDocumentEnvironment{slideshow}{}{
8323 \begin{stex_annotate_env}{data-ftml-slideshow={}}
8324 \cs_set_protected:Npn \nextslide ##1 {

```

```

8325     \begin{stex_annotate_env}{data-ftml-slideshow-slide={}} ##1
8326     \end{stex_annotate_env}
8327   }
8328   \cs_set_protected:Npn \lastslide ##1 {
8329     \begin{stex_annotate_env}{data-ftml-slideshow-slide={}} ##1
8330     \end{stex_annotate_env}
8331   }
8332 }{
8333   \end{stex_annotate_env}
8334 }
8335 }{
8336   \int_new:N \l__notesslides_slideshow_counter_int
8337   \NewDocumentEnvironment{slideshow}{}{
8338     \int_zero:N \l__notesslides_slideshow_counter_int
8339     \cs_set_protected:Npn \nextslide ##1 {
8340       \int_incr:N \l__notesslides_slideshow_counter_int
8341       \only<\int_use:N \l__notesslides_slideshow_counter_int>{##1}
8342     }
8343     \cs_set_protected:Npn \lastslide ##1 {
8344       \int_incr:N \l__notesslides_slideshow_counter_int
8345       \only<\int_use:N \l__notesslides_slideshow_counter_int ->{##1}
8346     }
8347   }{
8348
8349   }
8350 }
8351
8352 \bool_if:NT \c_notesslides_notes_bool {
8353   \def\author{\@dblarg\ns@author}
8354   \long\def\ns@author[#1]#2{%
8355     \tl_if_empty:nTF{#1}{
8356       \def\beamer@shortauthor{#2}
8357     }{
8358       \def\beamer@shortauthor{#1}
8359     }
8360     \def\@author{#2}
8361   }
8362   \def\title{\@dblarg\ns@title}
8363   \long\def\ns@title[#1]#2{%
8364     \tl_if_empty:nTF{#1}{
8365       \def\beamer@shorttitle{#2}
8366     }{
8367       \def\beamer@shorttitle{#1}
8368     }
8369     \def\@title{#2}
8370     \stexdoctitle{#2}
8371   }
8372   \def\insertshortauthor{
8373     \hbox\bgroup\def\\{\}\cs_if_exist:NT\beamer@shortauthor\beamer@shortauthor\egroup
8374   }
8375   \def\insertshorttitle{
8376     \hbox\bgroup\def\\{\}\cs_if_exist:NT\beamer@shorttitle\beamer@shorttitle\egroup
8377   }
8378   \stex_if_html_backend:TF{

```

```

8379 \def\insertframenumbers{\stex_annotate:nn{data-ftml-slide-number={}}{}}
8380 }{
8381 \def\insertframenumbers{@arabic\c@framenumbers}
8382 }
8383 \def\insertshortdate{\today}
8384 }

```

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.

```

8385 \gdef\printexcursions{
8386 \newcommand\excursionref[2]{% label, text
8387 \bool_if:NT \c_notesslides_notes_bool {
8388 \begin{sparagraph}[title=Excursion]
8389 #2 \sref[fallback=the appendix]{#1}.
8390 \end{sparagraph}
8391 }
8392 }
8393 \newcommand\activate@excursion[2][]{
8394 \tl_gput_right:Nn\printexcursions{\inputref{#1}{#2}}
8395 }
8396 \newcommand\excursion[4][]{% repos, label, path, text
8397 \bool_if:NT \c_notesslides_notes_bool {
8398 \activate@excursion[#1]{#3}
8399 \excursionref{#2}{#4}
8400 }
8401 }

```

(End of definition for **\excursion**. This function is documented on page 104.)

\excursiongroup

```

8402 \keys_define:nn{notesslides / excursiongroup }{
8403 id .str_set_x:N = \l__notesslides_excursion_id_str,
8404 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
8405 archive .str_set_x:N = \l__notesslides_excursion_mhrepos_str
8406 }
8407 \cs_new_protected:Nn \l__notesslides_excursion_args:n {
8408 \tl_clear:N \l__notesslides_excursion_intro_tl
8409 \str_clear:N \l__notesslides_excursion_id_str
8410 \str_clear:N \l__notesslides_excursion_mhrepos_str
8411 \keys_set:nn {notesslides / excursiongroup }{ #1 }
8412 }
8413 \newcommand\excursiongroup[1][]{
8414 \l__notesslides_excursion_args:n{ #1 }
8415 \tl_if_empty:NF\printexcursions
8416 {\IfInputref{}{\begin{note}
8417 \begin{sfragment}{Excursions}% TODO pass on id
8418 \ifdefempty\l__notesslides_excursion_intro_tl{\
8419 \exp_args:NNe \use:nn \inputref{\l__notesslides_excursion_mhrepos_str}{
8420 \l__notesslides_excursion_intro_tl
8421 }}
8422 }

```

```

8423     \printexcursions%
8424     \end{sfragment}
8425     \end{note}}}}
8426 }
8427 \ifcurname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi

(End of definition for \excursiongroup. This function is documented on page 104.)

8428 \prop_new:N \g__notesslides_variables_prop
8429 \cs_set_protected:Npn \setSGvar #1 #2 {
8430   \prop_gput:Nnn \g__notesslides_variables_prop {#1}{#2}
8431 }
8432 \cs_set_protected:Npn \useSGvar #1 {
8433   \prop_item:Nn \g__notesslides_variables_prop {#1}
8434 }
8435 \cs_set_protected:Npn \ifSGvar #1 #2 #3 {
8436   \prop_get:NnNF \g__notesslides_variables_prop {#1} \l__notesslides_tmp {
8437     \PackageError{document-structure}
8438       {The sTeX Global variable #1 is undefined}
8439       {set it with \protect\setSGvar}\TODO better error
8440   }
8441   \tl_if_eq:NnT \l__notesslides_tmp {#2}{ #3 }
8442 }
8443
8444
8445 \end{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.

```

8446 (*package)
8447 (@@=problems)
8448 \ProvidesExplPackage{problem}{2025/11/11}{4.0.0}{Semantic Markup for Problems}
8449 \RequirePackage{l3keys2e}
8450
8451 \keys_define:nn { problem / pkg }{
8452   notes      .default:n      = { true },
8453   notes      .bool_set:N     = \c__problems_notes_bool,
8454   gnotes     .default:n      = { true },
8455   gnotes     .bool_set:N     = \c__problems_gnotes_bool,
8456   hints      .default:n      = { true },
8457   hints      .bool_set:N     = \c__problems_hints_bool,
8458   solutions  .default:n      = { true },
8459   solutions  .bool_set:N     = \c__problems_solutions_bool,
8460   pts        .default:n      = { true },
8461   pts        .bool_set:N     = \c__problems_pts_bool,
8462   min        .default:n      = { true },
8463   min        .bool_set:N     = \c__problems_min_bool,
8464   %boxed     .default:n      = { true },
8465   %boxed     .bool_set:N     = \c__problems_boxed_bool,

```

```

8466 test .default:n = { true },
8467 test .bool_set:N = \c__problems_test_bool,
8468 unknown .code:n = {
8469 \PassOptionsToPackage{\CurrentOption}{stex}
8470 }
8471 }
8472 \newif\ifsolutions
8473
8474 \ProcessKeysOptions{ problem / pkg }
8475 \bool_if:NTF \c__problems_solutions_bool {
8476 \solutionstrue
8477 }{
8478 \solutionsfalse
8479 }
8480 \newif\ifintest
8481 \bool_if:NTF \c__problems_test_bool {
8482 \intesttrue
8483 }{
8484 \intestfalse
8485 }
8486
8487 \RequirePackage{stex}
8488
8489 \newcommand\precondition[2]{
8490 \stex_get_symbol:n{#2}
8491 \tl_if_empty:NF \l_stex_get_symbol_uri {
8492 \str_case:nnTF {#1}{
8493 {remember}{}
8494 {understand}{}
8495 {analyze}{}
8496 {evaluate}{}
8497 {apply}{}
8498 {create}{}
8499 }{
8500 \ifstexhtml\else\bool_if:NT\c_stex_metadata_bool {
8501 \marginpar{\tiny \textbf{Precondition:}~#1~
8502 \exp_args:Ne\symrefemph@uri{\stex_symbol_uri_name:N \l_stex_get_symbol_uri}{\l_st
8503 }
8504 }\fi
8505 \stex_annotate_invisible:nn{
8506 data-ftml-preconditionsymbol={\stex_use_symbol_uri:N \l_stex_get_symbol_uri},
8507 data-ftml-preconditiondimension={#1}
8508 }{}
8509 }{\errmessage{Unknown~cognitive~dimension~#1}}
8510 }
8511 }
8512 \newcommand\objective[2]{
8513 \stex_get_symbol:n{#2}
8514 \tl_if_empty:NF \l_stex_get_symbol_uri {
8515 \str_case:nnTF {#1}{
8516 {remember}{}
8517 {understand}{}
8518 {analyze}{}
8519 {evaluate}{}

```

```

8520     {apply}{}
8521     {create}{}
8522   }{
8523     \ifstexhtml\else\bool_if:NT\c_stex_metadata_bool {
8524       \marginpar{\tiny \textbf{Objective:}~\#1~
8525       \exp_args:Ne\symrefemph@uri{\stex_symbol_uri_name:N \l_stex_get_symbol_uri}{\l_stex
8526       }
8527     }\fi
8528     \stex_annotate_invisible:nn{
8529       data-ftml-objectivesymbol={\stex_use_symbol_uri:N \l_stex_get_symbol_uri},
8530       data-ftml-objectivedimension={\#1}
8531     }{}
8532   }{\errmessage{Unknown~cognitive~dimension~\#1}}
8533 }
8534 }

```

`\problem@kw@*` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```

8535 \AddToHook{begindocument}{
8536   \ExplSyntaxOn\makeatletter
8537   \input{problem-english.ldf}
8538   \ltx@ifpackageloaded{babel}{
8539     \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
8540     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8541       \input{problem-ngerman.ldf}
8542     }
8543     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8544       \input{problem-finnish.ldf}
8545     }
8546     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8547       \input{problem-french.ldf}
8548     }
8549     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8550       \input{problem-russian.ldf}
8551     }
8552   }{}
8553   \makeatother\ExplSyntaxOff
8554 }

```

(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.

```

8555 \bool_new:N \l_stex_key_autogradable_bool
8556 \stex_keys_define:nnnn{ problem }{
8557   \tl_clear:N \l_stex_key_pts_tl
8558   \tl_set:Nn \l_stex_key_min_tl 0
8559   \str_clear:N \l_stex_key_name_str
8560   \str_clear:N \l_stex_key_mhrepos_str
8561   \bool_set_false:N \l_stex_key_autogradable_bool
8562 }{

```



```

8563 pts .tl_set:N = \l_stex_key_pts_tl,
8564 min .tl_set:N = \l_stex_key_min_tl,
8565 name .str_set:N = \l_stex_key_name_str,
8566 autogradable .bool_set:N = \l_stex_key_autogradable_bool,
8567 archive .code:n = {},
8568 %archive .str_set:N = \l_stex_key_mhrepos_str,
8569 %creators .code:n = {}
8570 %imports .tl_set:N = \l__problems_prob_imports_tl,
8571 %refnum .int_set:N = \l__problems_prob_refnum_int,
8572 }{id,title,style,uses}

```

Then we set up a counter for problems.

`\numberproblemsin`

```

8573 \newcounter{sproblem}[section]
8574 \newcommand\numberproblemsin[1]{
8575   \@addtoreset{sproblem}{#1}
8576   \def\thesproblem{\arabic{#1}.\arabic{sproblem}}
8577 }
8578 \numberproblemsin[section]
8579 %\def\theplainsproblem{\arabic{sproblem}}
8580 %\def\thesproblem{\thesection.\theplainsproblem}

```

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

`sproblem (env.)`

```

8581 \cs_new:Nn \__problems_activate_macros: {
8582   \stex_reactivate_macro:N \solution
8583   \stex_reactivate_macro:N \mcb
8584   \stex_reactivate_macro:N \scb
8585   \stex_reactivate_macro:N \fillinsol
8586   \stex_reactivate_macro:N \hint
8587   \stex_reactivate_macro:N \exnote
8588   \stex_reactivate_macro:N \gnote
8589 }
8590
8591 \fp_new:N \g_problem_total_pts_fp
8592 \fp_new:N \g_problem_total_min_fp
8593
8594 \bool_new:N \l__problems_in_problem_bool
8595 \bool_new:N \l__problems_has_pts_bool
8596 \bool_new:N \l__problems_has_min_bool
8597 \bool_set_false:N \l__problems_in_problem_bool
8598 \stex_new_stylable_env:nnnnnn {problem} {0{}}{
8599   \bool_if:NT \l__problems_in_problem_bool {
8600     \msg_error:nn{stex}{error/nestedproblem}
8601   }
8602   \cs_if_exist:NTF \l_problem_inputproblem_keys_tl {
8603     \tl_put_left:Nn \l_problem_inputproblem_keys_tl {#1,}
8604     \exp_args:Nno \stex_keys_set:nn{problem}{
8605       \l_problem_inputproblem_keys_tl
8606     }
8607   }{
8608     \stex_keys_set:nn{problem}{#1}
8609   }

```

```

8610 \refstepcounter{sproblem}
8611
8612 \stex_if_do_html:T {
8613   \str_if_empty:NT \l_stex_key_name_str {
8614     \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8615     \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8616   }
8617   \exp_args:Nne \begin{stex_annotate_env} {
8618     data-ftml-problem={\l_stex_key_name_str},
8619     %data-ftml-language={ \l_stex_current_language_str},
8620     data-ftml-autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
8621     \tl_if_empty:NF \l_stex_key_pts_tl{,data-ftml-problempoints={\l_stex_key_pts_tl}}
8622   }
8623   \noindent \stex_annotate:nn{data-ftml-title={}}{\stex_annotate_force_break:n{\l_stex_k
8624   \tl_if_empty:NF \l_stex_key_title_tl {
8625     \exp_args:No \stexdoctitle \l_stex_key_title_tl
8626   }
8627   \stex_annotate_force_break:n{}}
8628 }
8629 \tl_set_eq:NN \thistitle \l_stex_key_title_tl
8630 \tl_if_empty:NT \l_stex_key_pts_tl { \tl_set:Nn \l_stex_key_pts_tl{0} }
8631
8632
8633 \bool_set_true:N \l__problems_in_problem_bool
8634 \tl_set_eq:NN \l__problems_pts_tl \l_stex_key_pts_tl
8635 \tl_set_eq:NN \l__problems_min_tl \l_stex_key_min_tl
8636 \tl_if_eq:NnTF \l__problems_pts_tl {0}
8637   {\bool_set_false:N \l__problems_has_pts_bool}
8638   {\bool_set_true:N \l__problems_has_pts_bool}
8639 \tl_if_eq:NnTF \l__problems_min_tl {0}
8640   {\bool_set_false:N \l__problems_has_min_bool}
8641   {\bool_set_true:N \l__problems_has_min_bool}
8642 \int_gzero:N \g__problems_subproblem_int
8643
8644 \stex_if_do_html:F\stex_style_apply:
8645 \stex_do_id:
8646 \__problems_activate_macros:
8647 }{
8648 \fp_gadd:Nn \g_problem_total_pts_fp { \l__problems_pts_tl}
8649 \fp_gadd:Nn \g_problem_total_min_fp { \l__problems_min_tl}
8650 \__problems_record_problem:
8651 \stex_if_do_html:F\stex_style_apply:
8652 \stex_if_do_html:T{ \end{stex_annotate_env} }
8653 }{
8654 \par\noindent\problemheader
8655 \stex_if_do_html:F{
8656   \bool_if:NT \c__problems_pts_bool {
8657     \tl_if_eq:NnF \l__problems_pts_tl {0}{
8658       \marginpar{\l__problems_pts_tl}{~\problem@kw@pts\smallskip}
8659     }
8660   }
8661   \bool_if:NT \c__problems_min_bool {
8662     \tl_if_eq:NnF \l__problems_min_tl {0} {
8663       \marginpar{\l__problems_min_tl}{~\problem@kw@minutes\smallskip}

```

```

8664     }
8665   }
8666 }
8667 \par
8668 \stex_ignore_spaces_and_pars:
8669 ){
8670   \par\bigskip
8671   % \bool_if:NT \c__problems_test_bool \pagebreak
8672 }{s}
8673
8674 \tl_set:Nn \problemheader {
8675   \stex_if_do_html:TF{
8676     \tl_if_empty:NF \thistitle {
8677       \stex_annotate:nn{data-ftml-title={}}{\textbf{\thistitle}}
8678     }
8679   }{
8680     \textbf{\sproblemautorefname{~}\thesproblem
8681       \tl_if_empty:NF \thistitle {
8682         {~}(\thistitle)
8683       }
8684     }
8685   }
8686 }
8687
8688 \cs_new_protected:Nn \__problems_record_problem: {
8689   \exp_args:NNe \iow_now:Nn \@auxout {
8690     \problem@restore {\thesproblem}{\l__problems_pts_tl}{\l__problems_min_tl}
8691   }
8692 }
8693
8694 \cs_new_protected:Npn \problem@restore #1 #2 #3 {}

```

subproblem (env.)

```

8695 \int_new:N \g__problems_subproblem_int
8696
8697 \stex_new_stylable_env:nnnnnn {subproblem} {0{}}{
8698   \stex_keys_set:nn{problem}{#1}
8699   \bool_if:NF \l__problems_in_problem_bool{
8700     \ifstexhtml\else
8701       \par{\bfseries WARNING~subproblem~to~be~used~in~some~problem}\par
8702     \fi
8703     \__problems_activate_macros:
8704     \bool_set_true:N \l__problems_in_problem_bool
8705     \tl_set:Nn \l__problems_pts_tl{0}
8706     \tl_set:Nn \l__problems_min_tl{0}
8707   }
8708   \str_if_empty:NT \l_stex_key_name_str {
8709     \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8710     \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str
8711   }
8712   \stex_if_do_html:T {
8713     \str_if_empty:NT \l_stex_key_name_str {
8714       \stex_file_split_off_lang:NN \l__problems_path_seq \g_stex_current_file
8715       \seq_get_right:NN \l__problems_path_seq \l_stex_key_name_str

```

```

8716 }
8717 \exp_args:Nne \begin{stex_annotate_env} {
8718   data-ftml-subproblem={\l_stex_key_name_str},
8719   %data-ftml-language={ \l_stex_current_language_str},
8720   data-ftml-autogradable={\bool_if:NTF \l_stex_key_autogradable_bool {true}{false}}
8721   \tl_if_empty:NF \l_stex_key_pts_tl{,data-ftml-problempoints={\l_stex_key_pts_tl}}
8722 }
8723 \noindent \stex_annotate:nn{data-ftml-title={}}{\stex_annotate_force_break:n{\l_stex_k
8724 \tl_if_empty:NF \l_stex_key_title_tl {
8725   \exp_args:No \stexdoctitle \l_stex_key_title_tl
8726 }
8727 \tl_if_empty:NF \l_stex_key_title_tl {
8728   \exp_args:No \stexdoctitle \l_stex_key_title_tl
8729 }
8730 \stex_annotate_force_break:n{}}
8731 }
8732 \tl_if_empty:NT \l_stex_key_pts_tl { \tl_set:Nn \l_stex_key_pts_tl{0} }
8733 \int_gincr:N \g__problems_subproblem_int
8734 \bool_if:NF \l__problems_has_pts_bool {
8735   \tl_gset:Ne \l__problems_pts_tl {\fp_to_decimal:n {\l__problems_pts_tl + \l_stex_key_pt
8736 }
8737 \bool_if:NF \l__problems_has_min_bool {
8738   \tl_gset:Ne \l__problems_min_tl {\fp_to_decimal:n {\l__problems_min_tl + \l_stex_key_mi
8739 }
8740 \stex_if_smsmode:F {\stex_if_do_html:F \stex_style_apply: }
8741 }{
8742 \stex_if_do_html:TF{ \end{stex_annotate_env} }{\stex_if_smsmode:F\stex_style_apply:}
8743 }{
8744 \begin{list}{}{
8745   \setlength\topsep{0pt}
8746   \setlength\parsep{0pt}
8747   \setlength\rightmargin{0pt}
8748 } \item[\int_use:N \g__problems_subproblem_int .]
8749 \bool_if:NT \c__problems_pts_bool {
8750   \bool_if:NF \l__problems_has_pts_bool {
8751     \marginpar{\smallskip\l_stex_key_pts_tl{}}~\problem@kw@pts}
8752   }
8753 }
8754 \bool_if:NT \c__problems_min_bool {
8755   \bool_if:NF \l__problems_has_min_bool{
8756     \marginpar{\smallskip\l_stex_key_min_tl{}}~\problem@kw@minutes}
8757   }
8758 }
8759 \ignorespaces
8760 }{
8761 \end{list}
8762 }{}

```

\includeproblem

```

8763 \stex_keys_define:nnnn{ includeproblem }{
8764   \str_clear:N \l_stex_key_mhrepos_str
8765 }{
8766   archive .str_set:N = \l_stex_key_mhrepos_str,
8767   unknown .code:n = {}

```

```

8768 }{}
8769
8770 \NewDocumentCommand\includeproblem{O{} m}{
8771   \group_begin:
8772   \tl_set:Nn \l_problem_inputproblem_keys_tl {#1}
8773   \stex_keys_set:nn{includeproblem}{#1}
8774   \exp_args:Nno \use:nn{\inputref[]\l_stex_key_mhrepos_str}{#2}
8775   \group_end:
8776 }
8777

```

(End of definition for \includeproblem. This function is documented on page 109.)

solution (env.)

```

8778 \int_new:N \g_problem_id_counter
8779 \dim_new:N \l_stex_key_testspace_dim
8780 \stex_keys_define:nnnn{ solution }{
8781   \str_clear:N \l_stex_key_answerclass_str
8782   \dim_zero:N \l_stex_key_testspace_dim
8783 }{
8784   testspace .dim_set:N = \l_stex_key_testspace_dim,
8785   answerclass .str_set:N = \l_stex_key_answerclass_str
8786 }{id,title,style}
8787
8788 \cs_new_protected:Nn \__problems_solution_start:n {
8789   \stex_keys_set:nn{ solution }{#1}
8790   \str_if_empty:NT \l_stex_key_id_str {
8791     \int_gincr:N \g_problem_id_counter
8792     \str_set:Nx \l_stex_key_id_str {
8793       SOLUTION_\int_use:N \g_problem_id_counter
8794     }
8795   }
8796   \stex_if_do_html:T{
8797     \begin{stex_annotate_env}{
8798       data-ftml-solution=\l_stex_key_id_str,
8799       data-ftml-answerclass={\l_stex_key_answerclass_str}
8800     }
8801   }
8802 }
8803
8804 \stex_new_stylable_env:nnnnnnn { solution }{ O{} }{
8805   \stex_if_do_html:TF{
8806     \__problems_solution_start:n{#1}
8807   }{
8808     \ifsolutions
8809       \__problems_solution_start:n{#1}
8810       \stex_style_apply:
8811     \else
8812       \stex_keys_set:nn{ solution }{#1}
8813       \testspace{\l_stex_key_testspace_dim}
8814       \setbox\l_tmpa_box\vbox\bgroup
8815       \fi
8816     }
8817   }{

```

```

8818 \stex_if_do_html:TF{
8819 \end{stex_annotate_env}
8820 }{
8821 \ifsolutions
8822 \stex_style_apply:
8823 \stex_if_do_html:T{
8824 \end{stex_annotate_env}
8825 }
8826 \else
8827 \egroup
8828 \fi
8829 }
8830 }{
8831 \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8832 \noindent\emph{\problem@kw@solution\tl_if_empty:NF \l_stex_key_title_tl{
8833 {~}\l_stex_key_title_tl \str_if_empty:NF \l_stex_key_answerclass_str {
8834 (Answer Class: \l_stex_key_answerclass_str)
8835 }
8836 } :~}
8837 }{
8838 \par\rule[.3em]{\linewidth}{0.4pt}\newline
8839 }{ }
8840
8841 \stex_deactivate_macro:Nn \solution {sproblem~environments}

```

\startsolutions
\stopsolutions

```

8842 \cs_new_protected:Npn \startsolutions{
8843 \global\solutionstrue
8844 }
8845 \cs_new_protected:Npn \stopsolutions{
8846 \global\solutionsfalse
8847 }

```

(End of definition for \startsolutions and \stopsolutions. These functions are documented on page 106.)

hint (env.)

```

8848
8849 \stex_keys_define:nnnn{ problemenv }{{}{id,title,style}
8850
8851 \cs_new_protected:Nn \__problems_hint_start:n {
8852 \stex_keys_set:nn{ problemenv }{#1}
8853 \str_if_empty:NT \l_stex_key_id_str {
8854 \int_gincr:N \g_problem_id_counter
8855 \str_set:Nx \l_stex_key_id_str {
8856 HINT_\int_use:N \g_problem_id_counter
8857 }
8858 }
8859 \stex_if_do_html:T{
8860 \begin{stex_annotate_env}{
8861 data-ftml-problemhint=\l_stex_key_id_str
8862 }
8863 }
8864 }

```

```

8865
8866 \stex_new_stylable_env:nnnnnnn { hint }{ 0{ } }{
8867   \stex_if_do_html:TF{
8868     \__problems_hint_start:n{#1}
8869   }{
8870     \bool_if:NTF \c__problems_hints_bool {
8871       \__problems_hint_start:n{#1}
8872       \stex_style_apply:
8873     }{
8874       \setbox\l_tmpa_box\vbox\bgroup
8875     }
8876   }
8877 }{
8878   \stex_if_do_html:TF{
8879     \end{stex_annotate_env}
8880   }{
8881     \bool_if:NTF \c__problems_hints_bool {
8882       \stex_style_apply:
8883       \stex_if_do_html:T{
8884         \end{stex_annotate_env}
8885       }
8886     }{
8887       \egroup
8888     }
8889   }
8890 }{
8891   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8892   \noindent\emph{\problem@kw@hint\tl_if_empty:NF \l_stex_key_title_tl{
8893     {~}\l_stex_key_title_tl
8894   } :~}
8895 }{
8896   \par\rule[.3em]{\linewidth}{0.4pt}\newline
8897 }{ }
8898 \stex_deactivate_macro:Nn \hint {sproblem-environments}

```

exnote (*env.*)

```

8899 \cs_new_protected:Nn \__problems_exnote_start:n {
8900   \stex_keys_set:nn{ problemenv }{#1}
8901   \str_if_empty:NT \l_stex_key_id_str {
8902     \int_gincr:N \g_problem_id_counter
8903     \str_set:Nx \l_stex_key_id_str {
8904       EXNOTE_\int_use:N \g_problem_id_counter
8905     }
8906   }
8907   \stex_if_do_html:T{
8908     \begin{stex_annotate_env}{
8909       data-ftml-problemnote=\l_stex_key_id_str
8910     }
8911   }
8912 }
8913
8914 \stex_new_stylable_env:nnnnnnn { exnote }{ 0{ } }{
8915   \stex_if_do_html:TF{
8916     \__problems_exnote_start:n{#1}

```

```

8917   }{
8918     \bool_if:NTF \c__problems_notes_bool {
8919       \__problems_exnote_start:n{#1}
8920       \stex_style_apply:
8921     }{
8922       \setbox\l_tmpa_box\vbox\bgroup
8923     }
8924   }
8925 }{
8926   \stex_if_do_html:TF{
8927     \end{stex_annotate_env}
8928   }{
8929     \bool_if:NTF \c__problems_notes_bool {
8930       \stex_style_apply:
8931       \stex_if_do_html:T{
8932         \end{stex_annotate_env}
8933       }
8934     }{
8935       \egroup
8936     }
8937   }
8938 }{
8939   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8940   \noindent\emph{\problem@kw@note\tl_if_empty:NF \l_stex_key_title_tl{
8941     {~}\l_stex_key_title_tl
8942   } :~}
8943 }{
8944   \par\rule[.3em]{\linewidth}{0.4pt}\newline
8945 }{}
8946 \stex_deactivate_macro:Nn \exnote {sproblem~environments}

```

gnote (env.)

```

8947 \int_new:N \l__problems_anscls_int
8948
8949 \cs_new_protected:Nn \__problems_gnote_start:n {
8950   \stex_keys_set:nn{ problemenv }{#1}
8951   \str_if_empty:NT \l_stex_key_id_str {
8952     \int_gincr:N \g_problem_id_counter
8953     \str_set:Nx \l_stex_key_id_str {
8954       GNOTE_\int_use:N \g_problem_id_counter
8955     }
8956   }
8957
8958   \stex_if_do_html:TF{
8959     \begin{stex_annotate_env}{
8960       data-ftml-problemgnote=\l_stex_key_id_str
8961     }
8962   }
8963   \stex_style_apply:
8964 }
8965
8966 \stex_new_stylable_env:nnnnnnn { gnote }{ 0{} }{
8967   \stex_if_do_html:TF{
8968     \__problems_gnote_start:n{#1}

```



```

8969   }{
8970     \bool_if:NTF \c__problems_gnotes_bool {
8971       \__problems_gnote_start:n{#1}
8972     }{
8973       \setbox\l_tmpa_box\vbox\bgroup
8974     }
8975   }
8976   \stex_reactivate_macro:N \anscls
8977 }{
8978   \stex_if_do_html:TF{
8979     \end{stex_annotate_env}
8980   }{
8981     \bool_if:NTF \c__problems_gnotes_bool {
8982       \stex_style_apply:
8983       \stex_if_do_html:T{
8984         \end{stex_annotate_env}
8985       }
8986     }{
8987       \egroup
8988     }
8989   }
8990 }{
8991   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
8992   \noindent\emph{\problem@kw@grading\str_if_empty:NF \l_stex_key_title_tl{
8993     {~}\l_stex_key_title_tl
8994   } :~}
8995 }{
8996   \par\rule[.3em]{\linewidth}{0.4pt}\newline
8997 }{}
8998 \stex_deactivate_macro:Nn \gnote {sproblem~environments}
8999
9000
9001 \stex_keys_define:nnnn{ anscls }{
9002   \str_clear:N \l_stex_key_pts_str
9003   \tl_clear:N \l_stex_key_feedback_tl
9004 }{
9005   pts      .str_set:N = \l_stex_key_pts_str,
9006   feedback .tl_set:N = \l_stex_key_feedback_tl,
9007   update   .code:n    = {}
9008 }{id}
9009 \newcommand \anscls [2] [] {
9010   \stex_keys_set:nn{ anscls }{#1}
9011   \str_if_empty:NT \l_stex_key_id_str {
9012     \int_incr:N \l__problems_anscls_int
9013     \str_set:Nx \l_stex_key_id_str {
9014       AC\int_use:N \l__problems_anscls_int
9015     }
9016   }
9017   \stex_if_do_html:TF{
9018     \begin{list}{}{
9019       \setlength\topsep{0pt}
9020       \setlength\parsep{0pt}
9021       \setlength\rightmargin{0pt}
9022     }\item[] \exp_args:Ne \stex_annotate:nn{

```

```

9023     data-ftml-answerclass={\l_stex_key_id_str}
9024     \str_if_empty:NF \l_stex_key_pts_str{
9025       ,data-ftml-answerclass-pts={\l_stex_key_pts_str}
9026     }
9027   }{
9028     #2~
9029     \tl_if_empty:NF \l_stex_key_feedback_tl{
9030       \stex_annotate:nn{
9031         data-ftml-answerclass-feedback={}
9032       }{\l_stex_key_feedback_tl}
9033     }
9034   }
9035   \end{list}
9036 }{
9037   \begin{list}{}{
9038     \setlength\topsep{0pt}
9039     \setlength\parsep{0pt}
9040     \setlength\rightmargin{0pt}
9041   }\item[\l_stex_key_id_str] #2
9042     \str_if_empty:NF \l_stex_key_pts_str {\par
9043       ~ \problem@kw@points :~\l_stex_key_pts_str
9044     }
9045     \tl_if_empty:NF \l_stex_key_feedback_tl {\par
9046       ~ \problem@kw@feedback :~\l_stex_key_feedback_tl
9047     }
9048   \end{list}
9049 }
9050 }
9051 \stex_deactivate_macro:Nn \anscls {gnote~environments}

```

The margin pars are reader-visible, so we need to translate

```

9052 \def\pts#1{
9053   \bool_if:NT \c__problems_pts_bool {
9054     \stex_annotate:nn{data-ftml-problempoints={#1}}{\marginpar{#1~\problem@kw@pts}}
9055   }\hbox_unpack:N\c_empty_box
9056 }
9057 \def\min#1{
9058   \bool_if:NT \c__problems_min_bool {
9059     \stex_annotate:nn{data-ftml-problemminutes={}}{\marginpar{#1~\problem@kw@minutes}}
9060   }\hbox_unpack:N\c_empty_box
9061 }

```

mcb (env.)

```

9062 \stex_new_stylable_env:nnnnnnn{mcb}{0{}}{
9063   \stex_keys_set:nn{style}{#1}
9064   \cs_set:Nn \__problems_mccline:n{
9065     \begin{list}{}{
9066       \setlength\topsep{0pt}
9067       \setlength\parsep{0pt}
9068       \setlength\rightmargin{0pt}
9069     }\item[\problem_mcc_box_tl] ##1 \end{list}
9070   }
9071
9072   \stex_if_do_html:T{

```

```

9073 \tl_set:Nn\problem_mcc_box_tl{}
9074 \__problems_maybe_inline:n{
9075   \exp_args:Nne \begin{stex_annotate_env}{
9076     data-ftml-multiple-choice-block={
9077       \clist_if_empty:NF \l_stex_key_style_clist {,
9078         data-ftml-styles={\l_stex_key_style_clist}
9079       }
9080     }
9081   }
9082   \clist_if_in:NnTF \l_stex_key_style_clist {inline} {
9083     \cs_set:Nn \__problems_mccline:n {##1}
9084   }{
9085     \clist_if_in:NnT \l_stex_key_style_clist {dropdown} {
9086       \cs_set:Nn \__problems_mccline:n {##1}
9087     }
9088   }
9089 }
9090 \stex_deactivate_macro:Nn \mcb {sproblem~environments}
9091 \stex_deactivate_macro:Nn \scb {sproblem~environments}
9092 \stex_deactivate_macro:Nn \solution {sproblem~environments}
9093 \stex_deactivate_macro:Nn \hint {sproblem~environments}
9094 \stex_deactivate_macro:Nn \exnote {sproblem~environments}
9095 \stex_deactivate_macro:Nn \gnote {sproblem~environments}
9096 \stex_reactivate_macro:N \mcc
9097 \stex_if_do_html:F \stex_style_apply:
9098 }{
9099   \stex_if_do_html:TF{
9100     \__problems_maybe_inline:n{\end{stex_annotate_env}}
9101   }\stex_style_apply:
9102 }{\par}{\par}
9103
9104 \stexstylemcb[inline]{
9105   {\Large[]\cs_set:Nn \__problems_mccline:n{\problem_mcc_box_tl{~} #1}
9106 }{\Large[]}}
9107
9108 \stexstylemcb[dropdown]{
9109   {\Large[]\cs_set:Nn \__problems_mccline:n{\problem_mcc_box_tl{~} #1}
9110 }{\Large[]}}
9111
9112 \stex_deactivate_macro:Nn \mcb {sproblem~environments}

```

we define the keys for the mcc macro

```

9113 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
9114   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
9115     \bool_set_true:N #1
9116   }{
9117     \bool_set_false:N #1
9118   }
9119 }
9120 \stex_keys_define:nnnn{mcc}{
9121   \tl_clear:N \l_stex_key_feedback_tl
9122   \bool_set_false:N \l_stex_key_T_bool
9123   \tl_clear:N \l_stex_key_Ttext_tl
9124   \tl_clear:N \l_stex_key_Ftext_tl

```

```

9125 }{
9126   feedback .tl_set:N      = \l_stex_key_feedback_tl ,
9127   T         .code:n        = {\bool_set_true:N \l_stex_key_T_bool} ,
9128   F         .code:n        = {\bool_set_false:N \l_stex_key_T_bool} ,
9129   Ttext     .tl_set:N      = \l_stex_key_Ttext_tl ,
9130   Ftext     .tl_set:N      = \l_stex_key_Ftext_tl ,
9131 }{id}
9132

```

\mcc

```

9133
9134 \cs_set:Nn \__problems_maybe_newline: { \ \ }
9135
9136 \stex_if_html_backend:TF{
9137   \tl_set:Nn \problem_mcc_box_tl {
9138     \ltx@ifpackageloaded{amssymb}{\square$}{
9139       \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
9140     }
9141   }
9142
9143   \newcommand\mcc[2][]{
9144     \stex_keys_set:nn{mcc}{#1}
9145     \__problems_mccline:n{
9146       \stex_if_do_html:T{
9147         \stex_annotate:nn{data-ftml-problem-choice={
9148           \bool_if:NTF \l_stex_key_T_bool {true}{false}
9149         }}{
9150           #2~
9151           \stex_annotate:nn{data-ftml-problem-choice-verdict={}}{
9152             \bool_if:NTF \l_stex_key_T_bool {
9153               \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
9154             }{
9155               \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
9156             }
9157           }
9158           \tl_if_empty:NF \l_stex_key_feedback_tl {
9159             \stex_annotate:nn{data-ftml-problem-choice-feedback={}}{
9160               \l_stex_key_feedback_tl
9161             }
9162           }
9163         }
9164       }
9165     }
9166   }
9167 }{
9168   \tl_set:Nn \problem_mcc_box_default_tl {
9169     \ltx@ifpackageloaded{amssymb}{\square$}{
9170       \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
9171     }
9172   }
9173   \tl_set:Nn \problem_mcc_box_tl {
9174     \ifsolutions
9175       \bool_if:NTF \l_stex_key_T_bool {
9176         \makebox[0pt][l]{\problem_mcc_box_default_tl}

```

```

9177         \hspace{0.1em}$\cs_if_exist:NTF\checkmark{
9178             \raisebox{.15ex}{\checkmark}
9179         } X$
9180     }{\problem_mcc_box_default_tl}
9181 \else
9182     \problem_mcc_box_default_tl
9183 \fi
9184 }
9185 \newcommand\mcc[2][]{
9186     \stex_keys_set:nn{mcc}{#1}
9187     \ifsolutions
9188         \tl_set:Nx \l_tmpa_tl{
9189             \bool_if:NTF \l_stex_key_T_bool {
9190                 \tl_if_empty:NF \l_stex_key_Ttext_tl {\exp_not:N \l_stex_key_Ttext_tl}
9191             }{
9192                 \tl_if_empty:NF \l_stex_key_Ftext_tl {\exp_not:N \l_stex_key_Ftext_tl}
9193             }
9194             \tl_if_empty:NF \l_stex_key_feedback_tl {
9195                 \exp_not:n{\emph{\l_stex_key_feedback_tl}}
9196             }
9197         }
9198     \fi
9199
9200     \__problems_mccline:n{ #2
9201         \ifsolutions
9202             \tl_if_empty:NF \l_tmpa_tl { \footnote{\l_tmpa_tl} }
9203         \fi
9204     }
9205 }
9206 }
9207 \stex_deactivate_macro:Nn \mcc {mcb~environments}

```

(End of definition for \mcc. This function is documented on page 107.)

scb (*env.*)

```

9208
9209 \cs_new_protected:Nn \__problems_maybe_inline:n {
9210     \clist_if_in:NnTF \l_stex_key_style_clist {inline} {
9211         \let\__problems_oldpar:\stex_par:
9212         \cs_set:Nn\stex_par:{~}
9213         \cs_set:Nn\__problems_maybe_newline:{}
9214         #1
9215         \let\stex_par:\__problems_oldpar:
9216     }{
9217         \clist_if_in:NnTF \l_stex_key_style_clist {dropdown} {
9218             \let\__problems_oldpar:\stex_par:
9219             \cs_set:Nn\stex_par:{~}
9220             \cs_set:Nn\__problems_maybe_newline:{}
9221             #1
9222             \let\stex_par:\__problems_oldpar:
9223         }{\par #1}
9224     }
9225 }
9226

```

```

9227 \stex_new_stylable_env:nnnnnnn{scb}{0{}}{
9228   \stex_keys_set:nn{style}{#1}
9229   \cs_set:Nn \__problems_sccline:n{
9230     \begin{list}{}{
9231       \setlength\topsep{0pt}
9232       \setlength\parsep{0pt}
9233       \setlength\rightmargin{0pt}
9234     }\item[\problem_scc_box_tl] ##1 \end{list}
9235   }
9236
9237   \stex_if_do_html:T{
9238     \__problems_maybe_inline:n{
9239       \exp_args:Ne\stex_annotate_env{
9240         data-ftml-single-choice-block={}
9241         \clist_if_empty:NF \l_stex_key_style_clist {,
9242           data-ftml-styles={\l_stex_key_style_clist}
9243         }
9244       }
9245     }
9246     \tl_set:Nn\problem_scc_box_tl{}
9247     \clist_if_in:NnTF \l_stex_key_style_clist {inline} {
9248       \cs_set:Nn \__problems_sccline:n {##1}
9249     }{
9250       \clist_if_in:NnT \l_stex_key_style_clist {dropdown} {
9251         \cs_set:Nn \__problems_sccline:n {##1}
9252       }
9253     }
9254   }
9255   \stex_deactivate_macro:Nn \mcb {sproblem~environments}
9256   \stex_deactivate_macro:Nn \scb {sproblem~environments}
9257   \stex_deactivate_macro:Nn \solution {sproblem~environments}
9258   \stex_deactivate_macro:Nn \hint {sproblem~environments}
9259   \stex_deactivate_macro:Nn \exnote {sproblem~environments}
9260   \stex_deactivate_macro:Nn \gnote {sproblem~environments}
9261   \stex_reactivate_macro:N \scc
9262   \stex_if_do_html:F \stex_style_apply:
9263 }{
9264   \stex_if_do_html:TF{
9265     \__problems_maybe_inline:n { \endstex_annotate_env }
9266   }\stex_style_apply:
9267 }{\par}{}{
9268
9269 \stexstylescb[inline]{
9270   {\Large[]\cs_set:Nn \__problems_sccline:n{\problem_scc_box_tl{~} #1}
9271 }{\{\Large[]}}
9272
9273 \stexstylescb[dropdown]{
9274   {\Large[]\cs_set:Nn \__problems_sccline:n{\problem_scc_box_tl{~} #1}
9275 }{\{\Large[]}}
9276
9277 \stex_deactivate_macro:Nn \scb {sproblem~environments}

```

\scc

9278

```

9279 \stex_if_html_backend:TF{
9280   \tl_set:Nn\problem_scc_box_tl{\$ \bigcirc\$}
9281   \newcommand\scc[2] []{
9282     \stex_keys_set:nn{mcc}{#1}
9283     \__problems_sccline:n{
9284       \stex_if_do_html:T{
9285         \stex_annotate:nn{data-ftml-problem-choice={
9286           \bool_if:NTF \l_stex_key_T_bool {true}{false}
9287         }}{
9288           #2~
9289           \stex_annotate:nn{data-ftml-problem-choice-verdict={}}{
9290             \bool_if:NTF \l_stex_key_T_bool {
9291               \tl_if_empty:NTF \l_stex_key_Ttext_tl \problem@kw@correct \l_stex_key_Ttext_tl
9292             }{
9293               \tl_if_empty:NTF \l_stex_key_Ftext_tl \problem@kw@wrong \l_stex_key_Ftext_tl
9294             }
9295           }
9296           \tl_if_empty:NF \l_stex_key_feedback_tl {
9297             \stex_annotate:nn{data-ftml-problem-choice-feedback={}}{
9298               \l_stex_key_feedback_tl
9299             }
9300           }
9301         }
9302       }
9303     }
9304   }
9305 }{
9306   \tl_set:Nn\problem_scc_box_default_tl{\$ \bigcirc\$}
9307   \tl_set:Nn\problem_scc_box_tl{
9308     \ifsolutions
9309       \bool_if:NTF \l_stex_key_T_bool {
9310         \makebox[Opt][l]{\problem_scc_box_default_tl}
9311         \hspace{0.1em}\$ \cs_if_exist:NTF\checkmark{
9312           \raisebox{.15ex}{\checkmark}
9313         } X\$
9314       }{\problem_scc_box_default_tl}
9315     \else
9316       \problem_scc_box_default_tl
9317     \fi
9318   }
9319   \newcommand\scc[2] []{
9320     \stex_keys_set:nn{mcc}{#1}
9321     \ifsolutions
9322       \tl_set:Nx \l_tmpa_tl{
9323         \bool_if:NTF \l_stex_key_T_bool {
9324           \tl_if_empty:NF \l_stex_key_Ttext_tl {\exp_not:N \l_stex_key_Ttext_tl}
9325         }{
9326           \tl_if_empty:NF \l_stex_key_Ftext_tl {\exp_not:N \l_stex_key_Ftext_tl}
9327         }
9328         \tl_if_empty:NF \l_stex_key_feedback_tl {
9329           \exp_not:n{ \\\emph{\l_stex_key_feedback_tl} }
9330         }
9331       }
9332     \fi

```

```

9333
9334   \_problems_sccline:n{ #2
9335     \ifsolutions
9336       \tl_if_empty:NF \l_tmpa_tl { \footnote{\l_tmpa_tl} }
9337     \fi
9338   }
9339 }
9340 }
9341
9342 \stex_deactivate_macro:Nn \scc {scb-environments}
9343
9344
9345 \newcommand\yesTnoF{
9346   \begin{scb}[style=inline]
9347     \scc[T]{yes}~\scc[F]{no}
9348   \end{scb}
9349 }
9350 \newcommand\yesFnoT{
9351   \begin{scb}[style=inline]
9352     \scc[F]{yes}~\scc[T]{no}
9353   \end{scb}
9354 }
9355 \newcommand\trueTfalseF{
9356   \begin{scb}[style=inline]
9357     \scc[T]{true}~\scc[F]{false}
9358   \end{scb}
9359 }
9360 \newcommand\trueFfalseT{
9361   \begin{scb}[style=inline]
9362     \scc[F]{true}~\scc[T]{false}
9363   \end{scb}
9364 }

```

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

\fillinsol

```

9365 \stex_keys_define:nnnn{fillinsol}{
9366   \tl_clear:N \l_problems_fillin_solution_tl
9367   \dim_zero:N \l_stex_key_testspace_dim
9368 }{
9369   testspace .dim_set:N = \l_stex_key_testspace_dim,
9370   exact .code:n = {\_problems_parse_fillin_arg:nnnn{exact}#1 },
9371   numrange .code:n = {\_problems_parse_fillin_arg:nnnn{numrange}#1 },
9372   regex .code:n = {\_problems_parse_fillin_arg:nnnn{regex}#1 }
9373 }{
9374
9375   \tl_const:Nn \c_problems_true_tl{T}
9376   \tl_const:Nn \c_problems_false_tl{F}
9377
9378   \msg_set:nnn{problem}{error/neither-true-nor-false}{
9379     Fillinsol~verdict~#1~is~neither~T~nor~F!
9380   }
9381
9382   \stex_if_html_backend:TF{

```



```

9383 \cs_new:Nn \__problems_parse_fillin_arg:nnnn {
9384   \tl_set:Nn \l_tmpa_tl{#3}
9385   \tl_if_eq:NNF \l_tmpa_tl\c__problems_true_tl{
9386     \tl_if_eq:NNF \l_tmpa_tl\c__problems_false_tl {
9387       \msg_error:nnn{problem}{error/neither-true-nor-false}{#3}
9388     }
9389   }
9390   \tl_put_right:Ne \l__problems_fillin_solution_tl {
9391     \stex_annotate:nn{
9392       data-ftml-fillin-case={#1},
9393       data-ftml-fillin-case-value={\tl_to_str:n{#2}},
9394       data-ftml-fillin-case-verdict={
9395         \tl_if_eq:NNTF\l_tmpa_tl\c__problems_true_tl{true}{false}
9396       },
9397       }{\_stex_annotate_force_break:n{\exp_not:n{#4}}}}
9398   }
9399 }
9400 }{
9401   \cs_new:Nn \__problems_parse_fillin_arg:nnnn {
9402     \tl_put_right:Nn \l__problems_fillin_solution_tl {
9403       #1 & \tl_to_str:n{#2} & #3 & #4 \crrc
9404     }
9405   }
9406 }
9407
9408
9409 \newcommand\fillinsol[2][]{
9410   \stex_if_do_html:F \quad
9411   \mode_if_math:TF{
9412     \hbox{\__problems_fillinsol:nn{#1}{#2}}
9413   }{
9414     \__problems_fillinsol:nn{#1}{#2}
9415   }
9416   \stex_if_do_html:F \quad
9417 }
9418
9419 \stex_if_html_backend:TF{
9420   \cs_new_protected:Nn \__problems_fillinsol:nn {
9421     \stex_keys_set:nn{fillinsol}{#1}
9422     \tl_if_empty:nF{#2}{
9423       \__problems_parse_fillin_arg:nnnn{exact}{#2}{T}{}}
9424   }
9425   \hbox_set:Nn \l_tmpa_box{\fbox{\huge{\texttt{\tl_to_str:n{#2}}}}\hskip\l_stex_key_testsp
9426   \exp_args:Ne \stex_annotate:nn{
9427     data-ftml-fillinsol={},
9428     data-ftml-fillinsol-width={\dim_to_decimal:n{1.5 \box_wd:N \l_tmpa_box }}
9429   }{
9430     \_stex_annotate_force_break:n{
9431       \l__problems_fillin_solution_tl
9432       #2
9433     }
9434   }
9435 }
9436 }{

```

```

9437 \cs_new_protected:Nn \__problems_fillinsol:nn {
9438   \stex_keys_set:nn{fillinsol}{#1}
9439   \ifsolutions
9440
9441     \cs_if_exist:NT\textcolor{\textcolor{red}}{\fbox{\texttt{\tl_to_str:n{#2}}}}
9442     \tl_if_empty:NF \l__problems_fillin_solution_tl {
9443       \footnote{
9444         \halign{ ~##~\hfil & ~##~\hfil & ~##~\hfil & ~##~\hfil \cr
9445           \textbf{type}&\textbf{case}&\textbf{verdict}&\textbf{feedback}\cr
9446           \tl_if_empty:nF{#2}{
9447             exact & #2 & T & \cr
9448           }
9449           \l__problems_fillin_solution_tl
9450         }
9451       }
9452     }
9453   \else
9454     \fbox{\dim_compare:nNnTF\l_stex_key_testspace_dim={0pt}{
9455       \phantom{\huge{\tl_to_str:n{#2}}}}
9456     }{
9457       \hspace{\l_stex_key_testspace_dim}\vphantom{\huge{A \tl_to_str:n{#2}}}
9458     }}
9459   \fi
9460 }
9461 }
9462
9463 \stex_deactivate_macro:Nn \fillinsol {sproblem-environments}

```

(End of definition for \fillinsol. This function is documented on page 109.)

\testemptypage

```

9464 \newcommand\testemptypage[1][\%
9465   \bool_if:NT \c__problems_test_bool {\ \vfill\begin{center}\hwexam@kw@testemptypage\end{center}
9466 }

```

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

\testspace

```

9467 \newcommand\testspace[1]{\bool_if:NT \c__problems_test_bool {\vspace*{#1}}}
9468 \newcommand\testsmallspace{\testspace{1cm}}
9469 \newcommand\testmedspace{\testspace{2cm}}
9470 \newcommand\testbigspace{\testspace{3cm}}

```

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

\testnewpage

```

9471 \newcommand\testnewpage{\bool_if:NT \c__problems_test_bool {\newpage}}

```

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

\testnewpage

```

9472 \newcommand{\testnewpageInProblem}%
9473 {\ifintest\vfill\begin{center}\emph{continued on next page}\end{center}\testnewpage\fi}%

```

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

```

9474 \</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.

```
9475 \*package)
9476 \ProvidesExplPackage{hwexam}{2025/11/11}{4.0.0}{homework assignments and exams}
9477 \RequirePackage{13keys2e}
9478
9479 \keys_define:nn {hwexam / pkg}{
9480   multiple .default:n = { false },
9481   multiple .bool_set:N = \c_hwexam_multiple_bool,
9482   qrcode .default:n = { false },
9483   qrcode .bool_set:N = \c_hwexam_qrcode_bool,
9484   unknown .code:n = {
9485     \PassOptionsToPackage{CurrentOption}{problem}
9486   }
9487 }
9488 \ProcessKeysOptions{ hwexam /pkg }
9489 \RequirePackage{problem}
```

`\hwexam_kw_*` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
9490 \AddToHook{begindocument}{
9491   \ExplSyntaxOn\makeatletter
9492   \input{hwexam-english.ldf}
9493   \ltx@ifpackageloaded{babel}{
9494     \clist_set:Nx \l_tmpa_clist {\exp_args:No \tl_to_str:n \bbl@loaded}
9495     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
9496       \input{hwexam-ngerman.ldf}
9497     }
9498     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
9499       \input{hwexam-finnish.ldf}
9500     }
9501     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
9502       \input{hwexam-french.ldf}
9503     }
9504     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
9505       \input{hwexam-russian.ldf}
9506     }
9507   }{}
9508   \makeatother\ExplSyntaxOff
9509 }
```

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

39.3.2 QR Codes

```
9510 \group_begin:
9511   \escapechar=-1
9512   \xdef\__qr_backslash{\string\}
9513 \group_end:
```

```

9514
9515 \bool_if:NT \c_hwexam_qrcode_bool {
9516   \RequirePackage{qrcode}
9517   \RequirePackage{marginnote}
9518   \str_new:N \g_@@_qr_json_str
9519   \bool_new:N \g_@@_qr_in_problems_json_bool
9520   \bool_set_false:N \g_@@_qr_in_problems_json_bool
9521   \bool_new:N \g_@@_qr_in_subproblems_json_bool
9522   \bool_set_false:N \g_@@_qr_in_subproblems_json_bool
9523   \bool_new:N \g_@@_qr_in_anscls_json_bool
9524   \bool_set_false:N \g_@@_qr_in_anscls_json_bool
9525   \bool_if:NTF \c__problems_gnotes_bool {
9526     \gdef \qrjson { \str_gput_right:Nx \g_@@_qr_json_str}
9527   }{
9528     \gdef \qrjson #1 {}
9529   }
9530   \qrjson{[]}
9531   \AtEndDocument{\qrjson{}}
9532   \def\__qr_escape_char:n #1 {
9533     #1\exp_args:Nno\use:nn{\if_charcode:w#1}\_@@_qr_backslash#1\fi
9534   }
9535
9536   \gdef\__qr_escape:n #1{\exp_args:Ne\str_map_function:nN{\tl_to_str:n{#1}}\_@@_qr_escape_
9537   \gdef \__qr_escape:o #1{\exp_args:No\_@@_qr_escape:n#1}
9538
9539   \def\qrschema{T0:D0:\examnumber}
9540
9541   \bool_if:NTF \c__problems_test_bool {
9542     \def\doproblemqr{
9543       \ifstexhtml\else{
9544         \reversemarginpar\marginnote{\qrcode[height=1.5cm]{\qrschema}}
9545         \normalmarginpar
9546       }\fi
9547     }
9548     \def\insertexamnumber{
9549       \ifstexhtml\else
9550         \tl_if_exist:NTF \examnumber {
9551           {\Large\bfseries ID:~\examnumber}
9552         }{
9553           {\color{red}\Large\bfseries!!!~ WARNING:~NO~{\string\examnumber}~SET~!!!}\gdef\examnum
9554         }
9555         \global\def\insertexamnumber{}
9556       \fi
9557     }
9558   }{
9559     \def\doproblemqr{}
9560     \def\insertexamnumber{}
9561   }
9562
9563   \stexstyleproblem[noqr]{
9564     \par\noindent\problemheader \xdef\grid{\thesproblem}
9565     \bool_if:NT \c__problems_pts_bool {
9566       \tl_if_eq:NnF \l__problems_pts_tl {0}{
9567         \marginpar{\l__problems_pts_tl}{~\problem@kw@points\smallskip}

```

```

9568     }
9569   }
9570   \bool_if:NT \c__problems_min_bool {
9571     \tl_if_eq:NnF \l__problems_min_tl {0} {
9572       \marginpar{\l__problems_min_tl}{~\problem@kw@minutes\smallskip}
9573     }
9574   }
9575   \par
9576   \stex_ignore_spaces_and_pars:
9577 }{
9578   \par\bigskip
9579 }
9580
9581 \stexstyleproblem{
9582   \par\noindent\problemheader \xdef\grid{\thesproblem}
9583   \doproblemqr
9584   \bool_if:NT \c__problems_pts_bool {
9585     \tl_if_eq:NnF \l__problems_pts_tl {0}{
9586       \marginpar{\l__problems_pts_tl}{~\problem@kw@points\smallskip}
9587     }
9588   }
9589   \bool_if:NT \c__problems_min_bool {
9590     \tl_if_eq:NnF \l__problems_min_tl {0} {
9591       \marginpar{\l__problems_min_tl}{~\problem@kw@minutes\smallskip}
9592     }
9593   }
9594
9595   \bool_if:NTF \g_@@_qr_in_problems_json_bool {
9596     \qrjson{,}
9597   }{
9598     \bool_gset_true:N \g_@@_qr_in_problems_json_bool
9599   }
9600
9601   \qrjson {
9602     \c_left_brace_str
9603     "id": "\_@@_qr_escape:o\l_stex_key_id_str", "title": "\_@@_qr_escape:o\l_stex_key_title_tl
9604     "number": "\thesproblem"
9605   }
9606
9607   \tl_if_eq:NnF \l__problems_pts_tl {0}{
9608     \qrjson {
9609       , "pts": \l__problems_pts_tl
9610     }
9611   }
9612   \par
9613   \stex_ignore_spaces_and_pars:
9614 }{
9615   \bool_if:NT \g_@@_qr_in_subproblems_json_bool {
9616     \qrjson {}
9617   }
9618   \bool_gset_false:N \g_@@_qr_in_subproblems_json_bool
9619   \qrjson {\c_right_brace_str}
9620   \par\bigskip
9621 }

```

```

9622
9623 \stexstylesubproblem[noqr]{
9624   \begin{list}{}{
9625     \setlength\topsep{0pt}
9626     \setlength\parsep{0pt}
9627     \setlength\rightmargin{0pt}
9628   }\item[\int_use:N \g__problems_subproblem_int .]
9629   \xdef\grid{\thesproblem.\int_use:N \g__problems_subproblem_int}
9630   \bool_if:NT \c__problems_pts_bool {
9631     \bool_if:NF \l__problems_has_pts_bool {
9632       \marginpar{\smallskip\l_stex_key_pts_tl}{~\problem@kw@points}
9633     }
9634   }
9635   \bool_if:NT \c__problems_min_bool {
9636     \bool_if:NF \l__problems_has_min_bool{
9637       \marginpar{\smallskip\l_stex_key_min_tl}{~\problem@kw@minutes}
9638     }
9639   }
9640 }{\end{list}}
9641
9642 \stexstylesubproblem{
9643   \begin{list}{}{
9644     \setlength\topsep{0pt}
9645     \setlength\parsep{0pt}
9646     \setlength\rightmargin{0pt}
9647   }\item[\int_use:N \g__problems_subproblem_int .]
9648   \xdef\grid{\thesproblem.\int_use:N \g__problems_subproblem_int}
9649   \doproblemqr
9650   \bool_if:NT \c__problems_pts_bool {
9651     \bool_if:NF \l__problems_has_pts_bool {
9652       \marginpar{\smallskip\l_stex_key_pts_tl}{~\problem@kw@points}
9653     }
9654   }
9655   \bool_if:NT \c__problems_min_bool {
9656     \bool_if:NF \l__problems_has_min_bool{
9657       \marginpar{\smallskip\l_stex_key_min_tl}{~\problem@kw@minutes}
9658     }
9659   }
9660   \bool_if:NTF \g_@@_qr_in_subproblems_json_bool {
9661     \qrjson {,}
9662   }{
9663     \qrjson {
9664       , "subproblems": [
9665     ]
9666     \bool_gset_true:N \g_@@_qr_in_subproblems_json_bool
9667   }
9668   \qrjson {
9669     \c_left_brace_str
9670     "id": "\_@@_qr_escape:o\l_stex_key_id_str", "title": "\_@@_qr_escape:o\l_stex_key_title_tl
9671     "number": "\thesproblem.\int_use:N \g__problems_subproblem_int"
9672   }
9673   \bool_if:NF \l__problems_has_pts_bool {
9674     \qrjson {
9675       , "pts": \l_stex_key_pts_tl

```

```

9676     }
9677   }
9678 }{
9679   \qrjson {\c_right_brace_str}
9680   \end{list}
9681 }
9682
9683 \stexstylegnote{
9684   \par\smallskip\rule[.3em]{\linewidth}{0.4pt}\newline\smallskip
9685   \noindent\emph{\problem@kw@grading\str_if_empty:NF \l_stex_key_title_tl{
9686     {~}\l_stex_key_title_tl
9687   } :~}
9688   \qrjson {
9689     ,"gnote": \c_left_brace_str
9690     "id": "\_@@_qr_escape:o\l_stex_key_id_str", "title": "\_@@_qr_escape:o\l_stex_key_title_tl{
9691       "anscls": [
9692     }
9693   }{
9694     \qrjson {
9695       ]\c_right_brace_str
9696     }
9697     \par\rule[.3em]{\linewidth}{0.4pt}\newline
9698   }
9699
9700   \renewcommand \anscls [2][ ] {
9701     \stex_keys_set:nn{ anscls }{#1}
9702     \str_if_empty:NT \l_stex_key_id_str {
9703       \int_incr:N \l__problems_anscls_int
9704       \str_set:Nx \l_stex_key_id_str {
9705         AC\int_use:N \l__problems_anscls_int
9706       }
9707     }
9708     \bool_if:NTF \g_@@_qr_in_anscls_json_bool {
9709       \qrjson{,}
9710     }{
9711       \bool_set_true:N \g_@@_qr_in_anscls_json_bool
9712     }
9713     \qrjson{
9714       \c_left_brace_str
9715       "id": "\_@@_qr_escape:o\l_stex_key_id_str", "description": "\_@@_qr_escape:n{#2}",
9716       "feedback": "\_@@_qr_escape:o\l_stex_key_feedback_tl",
9717       "pts": "\l_stex_key_pts_str"
9718     ]\c_right_brace_str
9719   }
9720   \begin{list}{}{
9721     \setlength\topsep{0pt}
9722     \setlength\parsep{0pt}
9723     \setlength\rightmargin{0pt}
9724   } \item[\l_stex_key_id_str]
9725     \stex_if_do_html:TF{
9726       \exp_args:Ne \stex_annotate:nn{
9727         data-ftml-answerclass={\l_stex_key_id_str}
9728         \str_if_empty:NF \l_stex_key_pts_str{
9729           ,data-ftml-answerclass-pts={\l_stex_key_pts_str}

```

```

9730     }
9731   }{
9732     #2
9733     \tl_if_empty:NF \l_stex_key_feedback_tl{
9734       \stex_annotate_invisible:nn{
9735         data-ftml-answerclass-feedback={true}
9736       }{\l_stex_key_feedback_tl}
9737     }
9738   }
9739 }{#2}
9740 \str_if_empty:NF \l_stex_key_pts_str {\par
9741 ~ \problem@kw@points :~\l_stex_key_pts_str
9742 }
9743 \str_if_empty:NF \l_stex_key_feedback_tl {\par
9744 ~ \problem@kw@feedback :~\l_stex_key_feedback_tl
9745 }
9746 \end{list}
9747 }
9748 \stex_deactivate_macro:Nn \anscls {gnote-environments}
9749
9750 \AtEndDocument{
9751   \message{^^J^^J\g_@@_qr_json_str^^J^^J}
9752   \bool_if:NT \c__problems_gnotes_bool {
9753     \iow_new:N \c_@@_qr_json_iow
9754     \iow_open:Nn \c_@@_qr_json_iow {\jobname-vollkorn.json}
9755     \iow_now:Nx \c_@@_qr_json_iow {\g_@@_qr_json_str}
9756     \iow_close:N \c_@@_qr_json_iow
9757   }
9758 }
9759
9760 \renewcommand\testemptypage[1][\%
9761 \bool_if:NT \c__problems_test_bool {\
9762 \xdef\grid{P\thepage}
9763 \doproblemqr
9764 \vfill\begin{center}\hwexam@kw@testemptypage\end{center}\eject
9765 }
9766 }
9767 }

```

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.)`

```

9768 \stex_keys_define:nnnn{ assignment }{
9769   \tl_clear:N \l_stex_key_number_tl
9770   \tl_clear:N \l_stex_key_given_tl
9771   \tl_clear:N \l_stex_key_due_tl
9772 }{
9773   number .tl_set:N      = \l_stex_key_number_tl,
9774   given   .tl_set:N      = \l_stex_key_given_tl,
9775   due     .tl_set:N      = \l_stex_key_due_tl,

```



```

9776   unknown .code:n = {}
9777 }{id,title,style}
9778
9779 \newcounter{assignment}
9780 \stex_new_stylable_env:nnnnnnn {assignment}{0}{}{
9781   \cs_if_exist:NTF \l_hwexam_includeassignment_keys_tl {
9782     \tl_put_left:Nn \l_hwexam_includeassignment_keys_tl {#1,}
9783     \exp_args:Nno \stex_keys_set:nn{assignment}{
9784       \l_hwexam_includeassignment_keys_tl
9785     }
9786   }{
9787     \stex_keys_set:nn{assignment}{#1}
9788   }
9789   \tl_if_empty:NF \l_stex_key_number_tl {
9790     \global\setcounter{assignment}{\int_eval:n{\l_stex_key_number_tl-1}}
9791   }
9792   \global\refstepcounter{assignment}
9793   \setcounter{sproblem}{0}
9794   \def\thesproblem{\theassignment.\arabic{sproblem}}
9795   \stex_if_do_html:T{
9796     \tl_if_empty:NF \l_stex_key_title_tl {
9797       \stexdoctitle \l_stex_key_title_tl
9798     }
9799   }
9800   \stex_style_apply:
9801   \_stex_do_id:
9802 }{
9803   \stex_style_apply:
9804 }{
9805   \par\begin{center}
9806   \textbf{\Large\assignmentautorefname~\theassignment}
9807   \tl_if_empty:NF \l_stex_key_title_tl {
9808     {~}---\l_stex_key_title_tl
9809   }
9810 } \par\smallskip
9811 \textbf{
9812   \tl_if_empty:NF \l_stex_key_given_tl {
9813     \hwexam@kw@given :~\l_stex_key_given_tl\quad
9814   }
9815   \tl_if_empty:NF \l_stex_key_due_tl {
9816     \hwexam@kw@due :~\l_stex_key_due_tl\quad
9817   }
9818 }
9819 \end{center}
9820 \par\bigskip
9821 }{
9822 \par\pagebreak
9823 }{}}

```

\includeassignment

```

9824 \NewDocumentCommand\includeassignment{0}{ m}{
9825   \group_begin:
9826   \tl_set:Nn \l_hwexam_includeassignment_keys_tl {#1}
9827   \stex_keys_set:nn{includeproblem}{#1}

```

```

9828 \exp_args:Nno \use:nn{\inputref[]\l_stex_key_mhrepos_str}{#2}
9829 \group_end:
9830 }

```

(End of definition for \includeassignment. This function is documented on page 69.)

Restoring information about problems:

```

9831 \prop_new:N \c_@@_problems_prop
9832 \tl_set:Nn \c_@@_total_mins_tl {0pt}
9833 \tl_set:Nn \c_@@_total_pts_tl {0pt}
9834 \int_new:N \c_@@_total_problems_int
9835 \cs_set_protected:Npn \problem@restore #1 #2 #3 {
9836   \int_gincr:N \c_@@_total_problems_int
9837   \prop_gput:Nnn \c_@@_problems_prop {#1}{#2}{#3}
9838   \tl_gset:Ne \c_@@_total_pts_tl { \dim_eval:n { \c_@@_total_pts_tl + #2pt }}
9839   \tl_gset:Ne \c_@@_total_mins_tl { \dim_eval:n { \c_@@_total_mins_tl + #2pt }}
9840 }

```

\correction@table This macro generates the correction table

```

9841 \newcommand\correction@table{
9842   \int_compare:nNnT \c_@@_total_problems_int = 0 {
9843     \int_incr:N \c_@@_total_problems_int
9844     \prop_put:Nnn \c_@@_problems_prop {-}{-}{-}
9845   }
9846   \tl_clear:N \l_tmpa_tl
9847   \tl_clear:N \l_tmpb_tl
9848   \tl_clear:N \l_tmpc_tl
9849   \prop_map_inline:Nn \c_@@_problems_prop {
9850     \tl_put_right:Nn \l_tmpa_tl { ##1 & }
9851     \tl_put_right:Nx \l_tmpb_tl { \use_i:nn ##2 & }
9852     \tl_put_right:Nn \l_tmpc_tl { & }
9853   }
9854   \resizebox{\textwidth}{!}{%
9855     \exp_args:Nne \begin{tabular}{|l|*{\int_use:N \c_@@_total_problems_int}{c|}c||l|}\hline
9856     &\exp_args:Ne \multicolumn{\int_eval:n{ \c_@@_total_problems_int + 1}}{c||}
9857     {\footnotesize\hwexam@kw@forgrading} &\\ \hline
9858     \hwexam@kw@probs & \l_tmpa_tl \hwexam@kw@sum & \hwexam@kw@grade\\ \hline
9859     \hwexam@kw@pts & \l_tmpb_tl \dim_to_decimal:n{\c_@@_total_pts_tl} & \\ \hline
9860     \hwexam@kw@reached & \l_tmpc_tl & \\ [.7cm] \hline
9861     \end{tabular}}

```

(End of definition for \correction@table. This function is documented on page ??.)

\testheading

```

9862 \def\hwexamheader{\input{hwexam-default.header}}
9863
9864 \def\hwexamminutes{
9865   \tl_if_empty:NTF \hwexam@duration {
9866     {\hwexam@min}~\hwexam@minutes@kw
9867   }{
9868     \hwexam@duration
9869   }
9870 }
9871
9872 \stex_keys_define:nnnn{ hwexam / testheading }{

```

```

9873 \tl_clear:N \hwexam@min
9874 \tl_clear:N \hwexam@duration
9875 \tl_clear:N \hwexam@reqpts
9876 \tl_clear:N \hwexam@tools
9877 }{
9878   min .tl_set:N = \hwexam@min,
9879   duration .tl_set:N = \hwexam@duration,
9880   reqpts .tl_set:N = \hwexam@reqpts,
9881   tools .tl_set:N = \hwexam@tools
9882 }{}
9883
9884 \newenvironment{testheading}[1][]{
9885   \stex_keys_set:nn { hwexam / testheading}{#1}
9886
9887   \tl_set:Nx \hwexam@totalpts {\dim_to_decimal:n \c_@@_total_pts_tl}
9888   \tl_set:Nx \hwexam@totalmin {\dim_to_decimal:n \c_@@_total_mins_tl}
9889   \tl_set:Nx \hwexam@checktime {\dim_to_decimal:n { \hwexam@min pt - \hwexam@totalmin pt }}
9890
9891   \newif\if@bonuspoints
9892   \tl_if_empty:NTF \hwexam@reqpts {
9893     \@bonuspointsfalse
9894   }{
9895     \tl_set:Nx \hwexam@bonuspts {
9896       \dim_to_decimal:n{\hwexam@totalpts pt - \hwexam@reqpts pt}
9897     }
9898     \@bonuspointstrue
9899   }
9900
9901   \makeatletter\hwexamheader\makeatother
9902 }{
9903   \newpage
9904 }

```

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

```

examdata
quizdata
homeworkdata
9905 \bool_new:N \g_@@_allow_data_bool
9906 \bool_gset_true:N \g_@@_allow_data_bool
9907 \AtBeginDocument{
9908   \bool_gset_false:N \g_@@_allow_data_bool
9909 }
9910
9911 \msg_set:nnn{stex}{hwexam/doubledata}{
9912   Exam/Homework/Quiz~Data~already~set
9913 }
9914
9915 \msg_set:nnn{stex}{hwexam/nodate}{
9916   Exam/Homework/Quiz~Data~missing~date~value
9917 }
9918
9919 \msg_set:nnn{stex}{hwexam/nocourse}{
9920   Exam/Homework/Quiz~Data~missing~course~value
9921 }
9922

```

```

9923 \stex_keys_define:nnnn{ hwexam / commondata }{
9924 % https://docs.rs/dateparser/latest/dateparser/#accepted-date-formats
9925 \str_clear:N \l_@@_key_exam_date_str
9926 \str_clear:N \l_@@_key_exam_course_id_str
9927 \str_clear:N \l_@@_key_exam_term_str
9928 \int_set:Nn \l_@@_key_exam_num_int {1}
9929 }{
9930 date .str_set:N = \l_@@_key_exam_date_str,
9931 course .str_set:N = \l_@@_key_exam_course_id_str,
9932 term .str_set:N = \l_@@_key_exam_term_str,
9933 num .int_set:N = \l_@@_key_exam_num_int
9934 }{}
9935
9936 \stex_keys_define:nnnn{ hwexam / examdata }{
9937 \bool_set_false:N \l_@@_key_exam_retake_bool
9938 }{
9939 retake .bool_set:N = \l_@@_key_exam_retake_bool
9940 }{hwexam/commondata}
9941
9942 \cs_set_protected:Nn \_@@_do_metadata:nnnn {
9943 \bool_if:NF \g_@@_allow_data_bool {
9944 \msg_fatal:nn{stex}{hwexam/doubledata}
9945 }
9946 \bool_gset_false:N \g_@@_allow_data_bool
9947 \stex_keys_set:nn { hwexam / #1}{#2}
9948 \str_set:Nn \g_stex_document_kind_str{#4}
9949 \str_if_empty:NT\l_@@_key_exam_date_str{
9950 \msg_fatal:nn{stex}{hwexam/nodate}
9951 }
9952 \str_if_empty:NT\l_@@_key_exam_course_id_str{
9953 \msg_fatal:nn{stex}{hwexam/nocourse}
9954 }
9955 \cs_if_exist:NT \date {
9956 \exp_args:No \date \l_@@_key_exam_date_str
9957 }
9958
9959 \tl_set:Ne \g_stex_document_kind_parameters_tl{
9960 ,data-ftml-document-kind-date={\l_@@_key_exam_date_str}
9961 ,data-ftml-document-kind-num=\int_use:N \l_@@_key_exam_num_int
9962 ,data-ftml-document-kind-course=\l_@@_key_exam_course_id_str
9963 \str_if_empty:NF \l_@@_key_exam_term_str{
9964 ,data-ftml-document-kind-term=\l_@@_key_exam_term_str
9965 }
9966 #3
9967 }
9968 }
9969
9970 \newcommand\examdata[1]{
9971 \_@@_do_metadata:nnnn{examdata}{#1}{
9972 ,data-ftml-document-kind-retake=\bool_if:NTF \l_@@_key_exam_retake_bool{true}{false}
9973 }{exam}
9974 }
9975
9976 \newcommand\homeworkdata[1]{

```

```

9977 \_@@_do_metadata:nnnn{commondata}{#1}{homework}
9978 }
9979
9980 \newcommand\quizdata[1]{
9981 \_@@_do_metadata:nnnn{commondata}{#1}{quiz}
9982 }
9983

```

(End of definition for examdata, quizdata, and homeworkdata. These functions are documented on page ??.)

```

9984 \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\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```

39.4 Tikzinput Implementation

```

9985 \begin{tikzinput}
9986 \begin{package}
9987
9988 %%%%%%%%%%% tikzinput.dtx %%%%%%%%%%%
9989
9990 \ProvidesExplPackage{tikzinput}{2025/11/11}{4.0.0}{tikzinput package}
9991 \RequirePackage{l3keys2e}
9992
9993 \keys_define:nn { tikzinput } {
9994   image .bool_set:N = \c_tikzinput_image_bool,
9995   image .default:n = false ,
9996   unknown .code:n = {}
9997 }
9998
9999 \ProcessKeysOptions { tikzinput }
10000
10001 \bool_if:NTF \c_tikzinput_image_bool {
10002   \RequirePackage{graphicx}

```

```

10003 \providecommand\usetikzlibrary[]{}
10004 \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
10005 }{
10006 \RequirePackage{tikz}
10007 \RequirePackage{standalone}
10008
10009
10010 \newcommand \tikzinput [2] [] {
10011 \setkeys{Gin}{#1}
10012 \ifx \Gin@ewidth \Gin@exclamation
10013 \ifx \Gin@eheight \Gin@exclamation
10014 \input { #2 }
10015 \else
10016 \resizebox{!}{ \Gin@eheight }{
10017 \input { #2 }
10018 }
10019 \fi
10020 \else
10021 \ifx \Gin@eheight \Gin@exclamation
10022 \resizebox{ \Gin@ewidth }{!}{
10023 \input { #2 }
10024 }
10025 \else
10026 \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
10027 \input { #2 }
10028 }
10029 \fi
10030 \fi
10031 }
10032 }
10033
10034 \newcommand \ctikzinput [2] [] {
10035 \begin{center}
10036 \tikzinput [#1] {#2}
10037 \end{center}
10038 }
10039 \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	
\!	6514
\\$	2098, 2101, 2105
* commands:	
*:n	137
\,	6514
\;	7790, 7820, 7822, 7835
@@ commands:	
\g_@@_allow_data_bool	9905, 9906, 9908, 9943, 9946
\@@_do_metadata:nnnn	9942, 9971, 9977, 9981
\l_@@_key_exam_course_id_str	9926, 9931, 9952, 9962
\l_@@_key_exam_date_str	9925, 9930, 9949, 9956, 9960
\l_@@_key_exam_num_int	9928, 9933, 9961
\l_@@_key_exam_retake_bool	9937, 9939, 9972
\l_@@_key_exam_term_str	9927, 9932, 9963, 9964
\c_@@_problems_prop	9831, 9837, 9844, 9849
\@@_qr_escape:n	9536, 9537, 9603, 9670, 9690, 9715, 9716
\@@_qr_escape_char:n	9532, 9536
\g_@@_qr_in_anscls_json_bool	9523, 9524, 9708, 9711
\g_@@_qr_in_problems_json_bool	9519, 9520, 9595, 9598
\g_@@_qr_in_subproblems_json_- bool	9521, 9522, 9615, 9618, 9660, 9666
\c_@@_qr_json_iow	9753, 9754, 9755, 9756
\g_@@_qr_json_str	9518, 9526, 9751, 9755
\c_@@_total_mins_tl	9832, 9839, 9888
\c_@@_total_problems_int	9834, 9836, 9842, 9843, 9855, 9856
\c_@@_total_pts_tl	9833, 9838, 9859, 9887
\\	88, 115, 827, 1421, 7958, 7963, 8373, 8376, 9134, 9195, 9329, 9512, 9857, 9858, 9859, 9860
\{	7825
\}	7825
_	19, 79, 604, 741, 7990, 9465, 9761
_@@_qr_backslash	9512, 9533
_comp	3777, 4516, 5774, 5989, 6121, 6869
_customthiscomp	4876, 4877, 4889, 4894, 4897, 4902
_defcomp	5807, 5973
_thiscomp	4645, 4646, 5778
_varcomp	4218, 5133, 5154, 5704, 5796, 5904, 5918, 5932
\	2097, 2100, 2104
A	
\activateexcursion	104
\addbibresource	75, 133, 2114
\addcontentsline	7988
\addmhbibresource	75, 133, <u>2107</u>
\addtocounter	8183
\AddToHook	7517, 7610, 7614, 7618, 7720, 8535, 9490
\aftergroup	117, 118, 2867, 2871, 4479, 7716
\afterprematurestop	103, 8052, 8063
\anscls	8976, 9009, 9051, 9700, 9748
\apply	83
\arabic	8576, 8579, 9794
\arg	39, 88, 5254
\argarraymap	87, 146, 6201, <u>6759</u>
\argmap	87, 146, 285, 6200, 6517, <u>6735</u>
\argsep	35, 87, 146, 5637, 6199, 6512, <u>6716</u> , 7790, 7791, 7792, 7798
\assign	56, 137, 3202, 3461
assignment (env.)	69, 111, <u>9768</u>
\assignmentautorefname	9806
\assignMorphism	138, 3203, <u>3566</u>
\assumption	150, 7445, <u>7724</u>
\ast	7813
\AtBeginDocument	69, 77, 791, 799, 1440, 1467, 1648, 1673, 1745, 1749, 7860, 8007, 9907
\AtEndDocument	612, 1750, 9531, 9750
\AtEndOfPackageFile	2150, 2166, 2181
\author	8353
\autoref	79, 1814, 1830
B	
\backmatter	201, 1712, 1713, 1714

<code>\clist_if_empty:NTF</code>	<code>\cs_end:</code>
..... 454, 504, 542, 3944, 4706,	483, 486, 517, 520,
4753, 5073, 6288, 7152, 9077, 9241	605, 2380, 2381, 2382, 2390, 2394,
<code>\clist_if_in:NnTF</code>	2398, 2657, 6968, 6970, 7126, 7140
.. 104, 107, 131, 7247, 8540, 8543,	<code>\cs_generate_from_arg_count:Nnn</code>
8546, 8549, 9082, 9085, 9210, 9217, 3937,
9247, 9250, 9495, 9498, 9501, 9504	4290, 4434, 4611, 5372, 5408, 6644
<code>\clist_if_in:nnTF</code>	<code>\cs_generate_variant:Nn</code>
4837	56,
<code>\clist_item:Nn</code>	146, 176, 248, 331, 717, 1287, 2474,
5024, 6297	2480, 2917, 4484, 4506, 5367, 6347
<code>\clist_item:nn</code>	<code>\cs_if_eq:NNTF</code>
6778	65, 73, 787,
<code>\clist_map_function:NN</code> ...	795, 1167, 1645, 2687, 2701, 2704,
4579, 4756	2719, 2722, 4065, 5621, 5623, 6727
<code>\clist_map_function:nN</code>	<code>\cs_if_exist:NTF</code>
..... 3379, 3445, 4029, 6464
<code>\clist_map_inline:Nn</code> 660, 1520, 1524, 1567, 1571,
.. 126, 135, 457, 3114, 3946, 4299,	1587, 1590, 1619, 1623, 1657, 1663,
4443, 5074, 6892, 6927, 7050, 7085	1674, 1702, 1712, 1725, 1736, 1784,
<code>\clist_map_inline:nn</code>	1785, 1813, 1871, 1912, 1986, 1987,
363, 412, 5588, 5691, 5720, 6890, 7302	1989, 2073, 3805, 4060, 4548, 5103,
<code>\clist_pop:NN</code>	5106, 5114, 5117, 6376, 6381, 6413,
6299	6419, 6443, 6445, 6485, 6491, 7958,
<code>\clist_put_right:Nn</code>	7963, 7966, 7970, 7974, 7978, 7982,
... 127, 4592, 4596, 4735, 4855, 4857	8029, 8032, 8035, 8041, 8044, 8047,
<code>\clist_set:Nn</code> 42, 123, 4357, 8539, 9494	8208, 8216, 8228, 8238, 8373, 8376,
<code>\clist_set_eq:NN</code>	8602, 9177, 9311, 9441, 9781, 9955
121, 7102	<code>\cs_meaning:N</code> . 2379, 3971, 6350, 6499
<code>\l_tmpa_clist</code> 8539, 8540, 8543, 8546,	<code>\cs_new:Nn</code>
8549, 9494, 9495, 9498, 9501, 9504	.. 26, 34, 205, 306, 634, 748, 756,
<code>\clstininputmhlisting</code>	873, 880, 1067, 1074, 1086, 1089,
78, 133, 2166	1092, 1095, 1106, 1109, 1112, 1115,
<code>\cmhgraphics</code>	1118, 1124, 1130, 1195, 1198, 1201,
78, 133, 2150	1208, 1211, 1214, 1217, 1221, 1242,
<code>\cmhtikzininput</code>	1245, 1256, 1264, 1364, 1367, 1370,
113, 133, 2181	1378, 1381, 1384, 1387, 1390, 1393,
<code>\color</code>	1397, 1400, 1403, 1916, 1928, 2202,
9553	2205, 2208, 2211, 2214, 2217, 2220,
<code>\columnbox</code>	2223, 2226, 2406, 2411, 2416, 3042,
8224, 8226, 8244	3046, 3841, 4004, 4621, 4764, 4771,
<code>\comp</code>	4774, 4784, 5283, 5380, 5567, 5856,
30–32, 39, 85, 88, 98, 143,	5881, 6302, 6313, 6327, 6330, 6386,
3777, 4218, 4527, 4528, 4646, 4694,	6399, 6549, 6552, 6556, 6603, 6996,
4702, 4885, 4886, 4896, 5570, 5704,	7143, 7157, 7298, 8581, 9383, 9401
5766, 5818, 5819, 5904, 5918, 5932,	<code>\cs_new:Npn</code>
5973, 5989, 5991, 6121, 6498, 6592,	711,
6598, 6600, 6817, 6822, 6869, 7787,	712, 720, 721, 866, 869, 876, 975,
7789, 7796, 7798, 7800, 7801, 7806,	1121, 1127, 1248, 1924, 1953, 3190,
7808, 7813, 7815, 7816, 7821, 7822,	5616, 5860, 6020, 6392, 6405, 7045
7823, 7825, 7835, 7842, 7844, 7846	<code>\cs_new_nopar:Nn</code>
<code>\comp-macro</code>	359, 374
143	<code>\cs_new_protected:Nn</code>
<code>\compemph</code>	3, 4,
98, 143, 5766	42, 45, 53, 67, 73, 77, 87, 88, 97, 98,
<code>\conclude</code>	103, 113, 141, 153, 169, 177, 192,
150, 7444, 7725	208, 215, 245, 249, 313, 324, 333,
<code>\conclusion</code> ...	346, 482, 495, 503, 516, 541, 556,
43, 94, 96, 149, 7236, 7310	586, 598, 610, 615, 639, 684, 714,
<code>\copymod</code>	725, 726, 764, 810, 819, 883, 905,
53–55, 137, 3389, 3391, 3393	913, 926, 937, 954, 964, 979, 1133,
<code>copymodule (env.)</code>	
137, 3329	
<code>\copymodule</code>	
3364, 3366	
<code>\counterwithin</code>	
8036, 8048	
<code>\cr</code>	
9444, 9445, 9447	
<code>\crrcr</code>	
9403	
cs commands:	
<code>\cs:w</code>	
483, 486, 517, 520,	
605, 2380, 2381, 2382, 2388, 2392,	
2398, 2657, 6968, 6970, 7126, 7140	
<code>\cs_argument_spec:N</code>	
4063	

1139, 1157, 1228, 1232, 1270, 1289,	422, 694, 698, 701, 705, 706, 738,
1316, 1329, 1341, 1349, 1452, 1461,	1224, 1357, 1361, 1407, 1413, 1448,
1536, 1544, 1548, 1561, 1585, 1600,	1518, 1617, 1633, 1955, 1985, 2287,
1655, 1701, 1763, 1782, 1802, 1812,	2527, 2663, 2771, 2829, 2922, 2926,
1823, 1850, 1860, 1869, 1897, 1930,	3082, 3092, 3154, 3194, 3295, 3537,
1942, 1977, 2003, 2016, 2083, 2092,	4058, 4453, 4503, 4556, 4576, 4631,
2118, 2254, 2291, 2300, 2317, 2323,	4636, 4671, 4693, 4698, 4791, 4810,
2332, 2347, 2359, 2375, 2385, 2423,	4823, 5183, 5215, 5277, 5595, 5768,
2442, 2446, 2454, 2468, 2471, 2475,	5772, 5774, 5778, 5790, 5794, 5796,
2493, 2497, 2531, 2541, 2554, 2564,	5799, 5803, 5807, 5810, 5814, 6441,
2569, 2574, 2588, 2594, 2609, 2641,	6484, 6490, 6611, 6618, 6666, 6716,
2652, 2661, 2672, 2679, 2686, 2692,	6732, 6735, 6760, 6814, 6826, 6836,
2714, 2731, 2739, 2748, 2757, 2766,	7016, 7046, 7083, 7357, 7381, 7398,
2789, 2838, 2877, 2899, 2912, 2931,	7410, 7763, 7853, 8694, 8842, 8845
2969, 2978, 2988, 2999, 3007, 3020,	\cs_set:Nn 9064, 9083, 9086,
3050, 3071, 3096, 3104, 3112, 3122,	9105, 9109, 9134, 9212, 9213, 9219,
3147, 3162, 3167, 3172, 3178, 3182,	9220, 9229, 9248, 9251, 9270, 9274
3209, 3232, 3249, 3269, 3273, 3283,	\cs_set:Npe 3987
3303, 3305, 3310, 3353, 3420, 3473,	\cs_set:Npn
3489, 3517, 3529, 3548, 3617, 3680,	. 487, 524, 530, 601, 602, 885, 898,
3711, 3800, 3808, 3821, 3835, 3845,	904, 2619, 3212, 3226, 3339, 3344,
3856, 3907, 3954, 3970, 3977, 3983,	3405, 3410, 3938, 4009, 4011, 4014,
4007, 4024, 4033, 4045, 4052, 4078,	4027, 4080, 4291, 4335, 4435, 4611,
4093, 4106, 4117, 4119, 4131, 4172,	4830, 4836, 4881, 4905, 5373, 5409,
4196, 4233, 4246, 4259, 4309, 4317,	5718, 6194, 6195, 6199, 6200, 6201,
4323, 4333, 4381, 4395, 4408, 4466,	6462, 6498, 6518, 6629, 6644, 6741,
4485, 4514, 4523, 4541, 4545, 4550,	6766, 7768, 7769, 7770, 7934, 8257
4569, 4590, 4609, 4623, 4639, 4644,	\cs_set_eq:NN 54, 74, 315, 319, 1469,
4650, 4704, 4727, 4734, 4738, 4802,	1475, 1606, 1852, 1934, 3205, 3206,
4827, 4853, 4875, 4893, 4901, 4911,	3207, 4061, 4517, 4519, 4572, 5134,
4928, 4939, 4959, 4969, 5012, 5020,	5136, 5254, 5257, 5285, 5286, 5446,
5044, 5058, 5072, 5080, 5089, 5099,	5575, 5686, 5689, 6420, 6432, 6435,
5127, 5160, 5179, 5190, 5237, 5245,	6486, 6492, 6642, 6695, 6708, 8300
5299, 5360, 5368, 5384, 5390, 5427,	\cs_set_protected:Nn
5432, 5443, 5448, 5459, 5462, 5479, 1009, 7953, 7986, 8002, 9942
5482, 5497, 5500, 5515, 5519, 5529,	\cs_set_protected:Npn
5532, 5543, 5549, 5565, 5577, 5581, 69, 1485, 1495, 3801, 3802,
5641, 5652, 5684, 5716, 5762, 5817,	4902, 5766, 6512, 6517, 6523, 8030,
5976, 5997, 6050, 6058, 6061, 6081,	8033, 8042, 8045, 8301, 8324, 8328,
6142, 6174, 6223, 6235, 6262, 6282,	8339, 8343, 8429, 8432, 8435, 9835
6287, 6319, 6334, 6349, 6356, 6374,	\cs_set_protected:Npx 4877
6411, 6425, 6459, 6468, 6479, 6496,	\cs_undefine:N 351, 1040, 1786
6505, 6571, 6581, 6586, 6607, 6626,	\l_tmpa_cs 5718, 5736, 6194, 6204
6639, 6674, 6679, 6685, 6726, 6797,	\csname 349, 648, 2094,
6864, 6884, 6973, 6988, 7000, 7006,	2095, 2096, 2097, 2098, 2103, 2104,
7033, 7091, 7125, 7161, 7175, 7194,	2105, 2508, 3370, 3372, 3436, 3438
7204, 7210, 7262, 7277, 7290, 7368,	\ctikzinput 113, 10034
7424, 7459, 7473, 7488, 7494, 7534,	\currentgrouplevel . 325, 328, 334, 335,
7543, 7658, 7690, 7856, 7952, 8028,	337, 339, 341, 347, 349, 351, 353, 355
8040, 8079, 8117, 8154, 8159, 8165,	\CurrentOption 10, 7881, 7882,
8175, 8252, 8407, 8688, 8788, 8851,	7883, 7884, 7923, 7924, 8469, 9485
8899, 8949, 9113, 9209, 9420, 9437	\Currentsectionlevel 77, 131, 1485
\cs_new_protected:Npn	\currentsectionlevel 77, 131, 1485
. 16, 148, 303, 327,	

D

`\date` 9955, 9956
`\DeclareOption` 10
`\def` . 86, 654, 658, 661, 663, 1450, 2431,
 2457, 3720, 3777, 4218, 4225, 4374,
 4527, 4528, 4884, 4889, 4897, 4906,
 5704, 5789, 5819, 5879, 5896, 5904,
 5918, 5932, 5958, 5968, 5973, 5989,
 6121, 6146, 6166, 6869, 8051, 8053,
 8118, 8119, 8120, 8124, 8127, 8184,
 8292, 8309, 8317, 8353, 8354, 8356,
 8358, 8360, 8362, 8363, 8365, 8367,
 8369, 8372, 8373, 8375, 8376, 8379,
 8381, 8383, 8427, 8576, 8579, 8580,
 9052, 9057, 9532, 9539, 9542, 9548,
 9555, 9559, 9560, 9794, 9862, 9864
`\defemph` 98, 143, 5799
`\Definame` 95, 144, 5936, 7219, 7241
`\definame` 23, 32, 37,
 94, 95, 98, 144, 294, 5936, 7218, 7240
`\Definames` 5968
`\definames` 5958
`\definiendum` 23, 32,
 37, 94, 95, 98, 144, 5936, 7216, 7238
`\definiens` 39, 40,
 48, 56, 94–96, 7221, 7255, 7275, 7276
`definiens` 149, 7255
`\defnotation`
 32, 37, 95, 144, 5936, 7217, 7239
`\detokenize` 8540, 8543,
 8546, 8549, 9495, 9498, 9501, 9504
dim commands:
 `\dim_compare:nNnTF` 9454
 `\dim_eval:n` 9838, 9839
 `\dim_new:N` 8779
 `\dim_to_decimal:n`
 .. 9428, 9859, 9887, 9888, 9889, 9896
 `\dim_zero:N` 8782, 9367
`\dimexpr` 8288
do commands:
 `_do_comp:nNn` 143
 `_do_comp:nnNn` 143, 5775,
 5797, 5808, 5815, 5817, 6013, 6524
`\dobracket` 87
`\dobrackets` 147, 6805, 6814, 6837
`\document-URI` 117, 136
`\doproblemqr` . 9542, 9559, 9583, 9649, 9763
`\dowithbrackets` 147, 6836
`\due` 69, 111
`\duration` 70, 111

E

`\edef` 2096, 2097, 2098, 4646, 8135

`\egroup` 1855, 1937, 4306,
 4450, 7519, 7620, 8162, 8250, 8267,
 8373, 8376, 8827, 8887, 8935, 8987
`\eject` 9465, 9764
`\ellipses`
 90, 4591, 4593, 5692, 5730, 6745, 6746
`\else` 647, 659, 1426, 2032,
 2047, 2675, 2682, 5804, 8427, 8500,
 8523, 8700, 8811, 8826, 9181, 9315,
 9453, 9543, 9549, 10015, 10020, 10025
`\emph` 17, 18, 97, 1428,
 5814, 7512, 7529, 7644, 8832, 8892,
 8940, 8992, 9195, 9329, 9473, 9685
`\end` 120, 136, 1431, 1545,
 1614, 2164, 2179, 2199, 2558, 2704,
 2722, 2742, 3015, 6784, 7117, 7642,
 7996, 8003, 8055, 8064, 8137, 8162,
 8177, 8222, 8236, 8244, 8257, 8259,
 8272, 8273, 8274, 8275, 8276, 8277,
 8298, 8326, 8330, 8333, 8390, 8424,
 8425, 8652, 8742, 8761, 8819, 8824,
 8879, 8884, 8927, 8932, 8979, 8984,
 9035, 9048, 9069, 9100, 9234, 9348,
 9353, 9358, 9363, 9465, 9473, 9640,
 9680, 9746, 9764, 9819, 9861, 10037
`\endcsname` . 349, 647, 648, 2094, 2095,
 2096, 2097, 2098, 2103, 2104, 2105,
 2508, 3370, 3372, 3436, 3438, 8427
`\endgroup` 48, 50, 2024, 2524
`\endinput` 1864, 1945
`\endlist` 7545
`\endspfstepenv` ... 7680, 7684, 7736, 7740
endstex commands:
 `\endstex_annotate_env` ... 2285, 9265
`\ensuremath` 5465, 5471
environments:
 `assignment` 69, 111, 9768
 `blindfragment` 77, 131, 1585
 `copymodule` 137, 3329
 `exnote` 1, 8899
 `extstructure` 91, 148, 6888
 `extstructure*` 91
 `frame` 1, 8079
 `gnote` 1, 8947
 `hint` 1, 8848
 `interpretmodule` 137, 3394
 `mathstructure` 91, 148, 6853
 `mcb` 1, 9062
 `nassertion` 1
 `ndefinition` 1
 `nexample` 1
 `note` 1
 `nparagraph` 1, 1
 `nsproof` 1

problem	1	<code>\exp_args:Ne</code>	58,
sassertion	94, 149, 7228		63, 69, 77, 81, 157, 265, 266, 274,
scb	9208		791, 799, 803, 888, 931, 1408, 1766,
sdefinition	94, 149, 7223		1787, 1789, 1806, 1982, 2006, 2255,
sexample	94, 149, 7243		2664, 2782, 3056, 3064, 3065, 3115,
sfragment	77, 131, 1504		3551, 3763, 3832, 3909, 3998, 4063,
smodule	81, 134, 2230		4065, 4138, 4261, 4409, 4645, 4672,
solution	1, 8778		4676, 4755, 4766, 4837, 4876, 4894,
sparagraph	94, 149, 7246		5024, 5302, 5520, 5621, 5623, 5631,
spfblock	150, 7713		5637, 5872, 5889, 6012, 6361, 6727,
spfsketchenv	150, 7625		6960, 7020, 7086, 7132, 7213, 7232,
spfststepenv	150, 7710		7284, 7292, 7294, 7858, 8502, 8525,
sproblem	8581		9022, 9239, 9426, 9536, 9726, 9856
sproof	96, 150, 7422	<code>\exp_args:NNe</code>	89, 474, 477,
stex_annotate_env	120, 694		864, 874, 881, 1854, 1936, 2354,
stex_env_node	120, 694		2458, 2980, 3494, 4100, 4145, 4157,
subproblem	8695		4857, 4973, 4987, 5082, 5101, 5334,
subproof	150, 7547		5337, 5607, 5658, 5741, 5892, 5965,
testheading	70, 111		6208, 6304, 6315, 6982, 8419, 8689
titlefragment	131, 1563	<code>\exp_args:Nne</code>	304, 1034,
<code>\envname</code>	136		1299, 1441, 1553, 1767, 2065, 2086,
<code>\eq</code>	30		2543, 2814, 2913, 2942, 3157, 3721,
<code>\eqstep</code>	150, 7446, 7726		3754, 3822, 3997, 4226, 4610, 4896,
<code>\equal</code>	29		4913, 4961, 5028, 5030, 5083, 5371,
<code>\errmessage</code>	6370, 8509, 8532		5407, 5417, 6147, 6167, 6335, 6500,
<code>\escapechar</code>	88, 827, 9511		6530, 6875, 6975, 6978, 6981, 7097,
<code>\everyeof</code>	2653		7183, 7292, 7294, 7328, 7460, 7632,
<code>\examdata</code>	9970		7663, 7693, 8617, 8717, 9075, 9855
examdata	9905	<code>\exp_args:NNNe</code>	4147, 6608
<code>\examnumber</code>	9539, 9550, 9551, 9553	<code>\exp_args:NNne</code>	4234, 4382, 5313, 5319
<code>\excursion</code>	104, 8385	<code>\exp_args:NNNo</code>	
<code>\excursiongroup</code>	104, 8402		986, 2797, 3504, 4234, 4382
<code>\excursionref</code>	104, 8386, 8399	<code>\exp_args:NNno</code>	143
<code>exnote (env.)</code>	1, 8899	<code>\exp_args:Nnno</code>	3075, 3077
<code>\exnote</code>	8587, 8946, 9094, 9259	<code>\exp_args:NNNx</code>	986
exp commands:		<code>\exp_args:NNo</code>	
<code>\exp_after:wN</code>	158, 483, 486, 517, 520,		13, 42, 63, 83, 104, 107, 127,
	867, 870, 877, 975, 999, 1090, 1107,		131, 156, 222, 348, 735, 785, 989,
	1110, 1113, 1116, 1119, 1122, 1128,		1299, 1827, 2134, 2145, 2350, 2814,
	1196, 1209, 1212, 1218, 1225, 1250,		2846, 3501, 3507, 3511, 5110, 5111,
	1252, 1365, 1385, 1388, 1391, 1401,		5121, 5122, 5735, 6255, 6268, 6656
	1500, 1910, 2206, 2212, 2218, 2224,	<code>\exp_args:Nno</code>	335, 337, 372, 1862,
	2318, 2319, 2380, 2381, 2382, 2388,		1892, 2432, 3059, 3936, 4100, 4145,
	2392, 2396, 2397, 2457, 2655, 2674,		4157, 4289, 4433, 4455, 4633, 4637,
	2675, 2676, 2681, 2682, 2683, 3065,		4846, 8604, 8774, 9533, 9783, 9828
	3132, 3133, 3134, 3370, 3372, 3436,	<code>\exp_args:NNx</code>	
	3438, 3483, 3484, 3485, 3521, 3941,		90, 93, 426, 439, 8540, 8543,
	4294, 4438, 4961, 5034, 5035, 5040,		8546, 8549, 9495, 9498, 9501, 9504
	5198, 5208, 5210, 5253, 5386, 5393,	<code>\exp_args:No</code>	48, 257,
	5394, 5395, 5403, 5404, 5405, 5418,		308, 348, 570, 576, 642, 653, 770,
	5600, 5602, 5707, 5708, 5709, 5722,		899, 998, 1478, 1553, 1572, 1826,
	5928, 5984, 6008, 6388, 6401, 6749,		1834, 1835, 1861, 1872, 1875, 1880,
	6777, 6967, 6970, 7039, 7126, 7140		1882, 1888, 1889, 1890, 1899, 1912,
			1920, 1938, 1943, 1970, 1971, 2195,

2271, 2304, 2364, 2462, 2494, 2780, 2784, 2975, 2979, 3062, 3142, 3275, 3296, 3311, 3319, 3518, 3524, 3525, 3568, 3750, 3813, 3829, 3989, 3994, 4098, 4147, 4240, 4241, 4242, 4252, 4253, 4254, 4388, 4389, 4390, 4401, 4402, 4403, 4470, 4473, 4477, 4504, 4597, 4711, 4715, 4722, 4749, 4785, 4787, 4859, 4861, 4867, 4949, 4976, 4982, 4984, 4989, 4993, 5002, 5025, 5039, 5100, 5311, 5339, 5410, 5415, 5420, 5422, 5621, 5623, 5665, 5750, 6035, 6177, 6577, 6767, 6872, 6880, 6936, 6939, 6961, 7021, 7038, 7190, 7272, 7947, 8011, 8012, 8015, 8539, 8625, 8725, 8728, 9494, 9537, 9956	665, 1432, 2036, 2047, 2676, 2683, 4515, 4546, 5132, 5804, 5977, 5998, 8057, 8125, 8128, 8246, 8427, 8504, 8527, 8702, 8815, 8828, 9183, 9198, 9203, 9317, 9332, 9337, 9459, 9473, 9533, 9546, 9556, 10019, 10029, 10030
<code>\exp_args:Nx</code> 8080, 9114	<code>fiboxed</code> 100
<code>\exp_not:N</code> 94, 257, 308, 310, 898, 1049, 1050, 1055, 1056, 1087, 1096, 1097, 1102, 1202, 1203, 1371, 1372, 1772, 1856, 1938, 2402, 2653, 2673, 2681, 2943, 3558, 4468, 4469, 4472, 4476, 4480, 4537, 4613, 4616, 4710, 4786, 4860, 4866, 4886, 4919, 4948, 4951, 4975, 4981, 4990, 4995, 5004, 5039, 5262, 5268, 5381, 5659, 5664, 5742, 5746, 5748, 6176, 6583, 7958, 7963, 9190, 9192, 9324, 9326	file commands: <code>\file_if_exist:nTF</code> 599, 616, 966 <code>\fillinsol</code> 109, 8585, 9365 <code>\fn</code> 45 <code>\foo</code> 16, 45, 83 <code>\fooname</code> 16 <code>\footnote</code> 9202, 9336, 9443 <code>\footnotesize</code> 9857 <code>\foral</code> 38, 39 <code>\forall</code> 38, 7822
<code>\exp_not:n</code> 257, 308, 899, 1553, 1775, 1776, 1882, 1888, 1918, 1920, 1938, 2396, 2407, 2462, 3782, 3829, 3990, 4240, 4241, 4242, 4252, 4253, 4254, 4388, 4389, 4390, 4401, 4402, 4403, 4470, 4473, 4477, 4479, 4481, 4621, 4711, 4749, 4780, 4785, 4787, 4859, 4861, 4863, 4867, 4870, 4880, 4949, 4952, 4976, 4978, 4982, 4984, 4989, 4993, 5002, 5039, 5394, 5410, 5415, 5420, 5422, 5644, 5665, 5702, 5707, 5750, 6177, 6217, 6218, 6577, 6749, 6777, 9195, 9329, 9397	fp commands: <code>\fp_gadd:Nn</code> 8648, 8649 <code>\fp_new:N</code> 8591, 8592 <code>\fp_to_decimal:n</code> 8735, 8738
<code>\expandafter</code> 48, 648, 2094, 2095, 2096, 2097, 2098, 2508, 5804, 8055, 8056	<code>frame (env.)</code> 1, 8079 <code>\frameimage</code> 103, 8281 <code>frameimages</code> 100 <code>\frametitle</code> 8193 <code>\frontmatter</code> 201, 1702, 1703, 1704 <code>\fun</code> 38 <code>\funspace</code> 33
<code>\ExplSyntaxOff</code> 2532, 2580, 7865, 8553, 9508	
<code>\ExplSyntaxOn</code> 135, 2528, 2579, 7862, 8536, 9491	
<code>\extref</code> 80, 1975	
<code>extstructure (env.)</code> 91, 148, 6888	
<code>\extstructure</code> 6917, 6919	
<code>extstructure* (env.)</code> 91	
F	
<code>\fbox</code> 9425, 9441, 9454	
<code>\fi</code> 650,	
	G
	<code>\gdef</code> 1454, 1471, 1476, 8385, 9526, 9528, 9536, 9537, 9553
	<code>\given</code> 69, 111
	<code>\global</code> 1450, 2095, 7847, 7848, 8843, 8846, 9555, 9790, 9792
	<code>gnote (env.)</code> 1, 8947
	<code>\gnote</code> 8588, 8998, 9095, 9260
	<code>gnotes</code> 69, 105, 110
	group commands: <code>\group_begin:</code> 18, 78, 87, 600, 740, 826, 2434, 2550, 2614, 2623, 2631, 2860, 2890, 2940, 2963, 3681, 3712, 3786, 4221, 4370, 4487, 4598, 4602, 4878, 4903, 5021, 5129, 5144, 5362, 5405, 5533, 5578, 5653, 5701, 5717, 5780, 5850, 5867, 5884, 5898, 5910, 5924, 5999, 6052, 6075, 6130, 6162, 6815, 7127, 7256, 7311, 7491, 7536, 7692, 7767, 8771, 9510, 9825
	<code>\group_end:</code> 82, 85, 93, 606, 807, 830, 2448, 2559, 2627, 2634, 2637, 2865, 2896, 2942, 3016, 3689, 3726, 3791, 4231, 4378, 4548, 4561, 4587, 4599, 4603,

4633, 4794, 4850, 4887, 4908, 4914, 4987, 5040, 5158, 5195, 5206, 5222, 5233, 5241, 5274, 5365, 5386, 5418, 5423, 5541, 5578, 5658, 5709, 5741, 5784, 5995, 6017, 6054, 6075, 6133, 6172, 6821, 7136, 7258, 7322, 7497, 7545, 7707, 7850, 8775, 9513, 9829 <code>\group_insert_after:N</code> 340, 354	<code>\IfInputref</code> . . . 26, 27, 78, 132, <u>2040</u> , 8416 <code>\ifinputref</code> . . 26, 78, 132, 2030, <u>2039</u> , 2047 <code>\ifintest</code> 8480, 9473 <code>\ifmmode</code> 5804 <code>\ifnotes</code> 101, 7938, 8075 <code>\ifSGvar</code> 104, 8435 <code>\ifsolutions</code> 106, 8472, 8808, 8821, 9174, 9187, 9201, 9308, 9321, 9335, 9439 <code>\ifstexhtml</code> 26, 76, 120, 670, 1426, 8500, 8523, 8700, 9543, 9549 <code>\ifvmode</code> 4515, 4546, 5132, 5977, 5998 <code>\ifx</code> 2094, 8054, 8123, 8126, 10012, 10013, 10021 <code>\ignorespaces</code> 2867, 2871, 4231, 4378, 8248, 8759 <code>image</code> 112 <code>\importmodule</code> 34, 41, 46, 88– 90, 116, 120, 135, 137, 213, <u>2868</u> , 3199 <code>\includeassignment</code> 69, <u>9824</u> <code>\includegraphics</code> . . . 78, 133, 2162, 10005 <code>\includeproblem</code> 68, 109, <u>8763</u> <code>\indent</code> 4515, 4546, 5132, 5977, 5998 <code>\infprec</code> 85, 86, 147, 6255, <u>6793</u> , 7789 <code>\inline</code> 149 <code>\inline*</code> 95 <code>\inlineass</code> 43, 47, 49, 95, 149 <code>\inlinedef</code> 43, 95, 149 <code>\inlineex</code> 95, 149 <code>\input</code> . . . 25, 26, 75, 117, 135, 174, 44, 246, 669, 2057, 8537, 8541, 8544, 8547, 8550, 9492, 9496, 9499, 9502, 9505, 9862, 10014, 10017, 10023, 10027 <code>\inputassignment</code> 111 <code>\inputref</code> 26, 27, 77, 78, 97, 116, 132, <u>1993</u> , 8300, 8301, 8394, 8419, 8774, 9828 <code>\inputref*</code> 102, <u>8300</u> <code>\inputreffalse</code> 2035, 2039 <code>\inputreftrue</code> 2018, 2033 <code>\insertexamnumber</code> 9548, 9555, 9560 <code>\insertframenumber</code> 8379, 8381 <code>\insertshortauthor</code> 8372 <code>\insertshortdate</code> 8383 <code>\insertshorttitle</code> 8375 <code>\inset</code> 38 <code>int</code> commands: <code>\int_case:nn</code> 1586 <code>\int_case:nnTF</code> 1519, 1618, 1656, 1675 <code>\int_compare:nNnTF</code> 328, 1556, 1594, 2063, 2085, 3495, 4707, 4718, 4811, 5224, 5620, 5630, 5636, 5695, 6182, 6216, 6773, 6803, 7158, 7196, 7390, 9842
H <code>\halign</code> 9444 <code>\hbox</code> 1463, 2856, 3464, 3479, 3774, 3804, 3909, 4261, 4424, 6506, 6901, 7057, 7435, 7477, 7558, 7754, 8161, 8373, 8376, 9139, 9170, 9412 <code>hbox</code> commands: <code>\hbox_set:Nn</code> 6191, 9425 <code>\hbox_unpack:N</code> 1487, 1497, 3803, 9055, 9060 <code>\HCode</code> 660 <code>\hfil</code> 7760, 9444 <code>\hfill</code> 7760 <code>hint (env.)</code> <u>1</u> , <u>8848</u> <code>\hint</code> 8586, 8898, 9093, 9258 <code>hints</code> 69, 105, 110 <code>\hline</code> 9855, 9857, 9858, 9859, 9860 <code>\homeworkdata</code> 9976 <code>homeworkdata</code> <u>9905</u> <code>\href</code> 1912, 1989 <code>\hrule</code> 7754, 9139, 9170 <code>\hskip</code> 9425 <code>\hspace</code> 9177, 9311, 9457 <code>\HTML</code> 16, 24 <code>\huge</code> 9425, 9455, 9457 <code>hwexam</code> commands: <code>\l_hwexam_includeassignment_</code> <code>keys_tl</code> 9781, 9782, 9784, 9826 <code>\hwexam_kw_*</code> <u>9490</u> <code>\c_hwexam_multiple_bool</code> 9481 <code>\c_hwexam_qrcode_bool</code> . . . 9483, 9515 <code>\hwexamheader</code> 9862, 9901 <code>\hwexamminutes</code> 9864 <code>\hyperlink</code> 1987	I <code>\if</code> 2674, 2681 <code>if</code> commands: <code>\if_charcode:w</code> 9533 <code>\IfBooleanTF</code> . . 3377, 3443, 3666, 5290, 6124, 6159, 7564, 7674, 7731, 8282 <code>\ifcsname</code> 647, 8427 <code>\ifdefempty</code> 8418 <code>\IfFileExists</code> 6, 21, 1835, 1899, 2004, 2133, 2144, 2817, 2822, 2831

\int_compare_p:nNn	\intesttrue	8482
..... 7360, 7372, 7384, 7401, 7413	ior commands:	
\int_decr:N	\ior_close:N	623, 629, 831, 997
\int_eval:n	\ior_map_inline:Nn	985
351, 5316, 5322, 6802, 7008, 9790, 9856	\ior_new:N	971
\int_gincr:N	\ior_open:Nn	617, 625, 980
8733, 8791, 8854, 8902, 8952, 9836	\ior_open:NnTF	825
\int_gset:Nn	\ior_str_get:NN	828
\int_gzero:N	\ior_str_map_inline:Nn	619, 626
\int_incr:N 1521, 1525, 1557, 1588,	\g_tmpa_ior	617,
1591, 1595, 1620, 1624, 1660, 1666,	619, 623, 625, 626, 629, 825, 828, 831	
3898, 3899, 3900, 3901, 5596, 6686,	iow commands:	
6772, 7365, 7377, 7388, 7405, 7417,	\iow_close:N	612, 622, 1750, 9756
7997, 8340, 8344, 9012, 9703, 9843	\iow_new:N	585, 1748, 9753
\int_new:N	\iow_now:Nn	620,
1948, 3854, 4050, 5243, 5580, 6759,	627, 715, 1768, 1771, 7861, 8689, 9755	
6789, 8336, 8695, 8778, 8947, 9834	\iow_open:Nn	611, 618, 1749, 9754
\int_set:Nn 1635, 1636, 1637, 1638,	\g_tmpa_iow	618, 620, 622
1639, 1640, 1642, 3286, 3769, 3863,	\isassociative	41, 42
3867, 3871, 3875, 3879, 3883, 3887,	\iscommutative	42
3891, 3895, 4082, 4122, 4183, 4211,	\item	7540, 8748, 9022,
4337, 5538, 6659, 6767, 6774, 6796,	9041, 9069, 9234, 9628, 9647, 9724	
7358, 7370, 7382, 7399, 7411, 9928	\itshape	7764
\int_set_eq:NN		
\int_step_function:nN ...	J	
\int_step_inline:nm	\jobname	72, 235, 591,
..... 3847, 6247, 6254, 6274, 6594	611, 616, 617, 618, 625, 630, 1749, 9754	
\int_use:N	\join	54
1649, 1670, 3825, 4238, 4250, 4386,		
4399, 4457, 5266, 5302, 5643, 5661,	K	
5744, 6323, 6428, 6659, 6793, 6794,	keys commands:	
7185, 7990, 8341, 8345, 8748, 8793,	\l_keys_choice_tl	3640
8856, 8904, 8954, 9014, 9628, 9629,	\keys_define:nm	29,
9647, 9648, 9671, 9705, 9855, 9961	372, 7875, 7917, 8402, 8451, 9479, 9993	
\int_zero:N	\l_keys_key_str 6098, 6101, 6846, 6849	
.. 3857, 3858, 5587, 6641, 6770, 8338	\l_keys_key_tl ...	13, 735, 6099, 6847
\c_max_int	\keys_set:nm	118, 376, 8411
\l_tmpa_int	keyval commands:	
6641, 6644, 6659,	\keyval_parse:NNn	4730
6686, 6770, 6772, 6773, 6774, 6778,		
7358, 7361, 7364, 7365, 7370, 7373,	L	
7376, 7377, 7382, 7385, 7388, 7390,	\label	79, 1766, 1787, 1789, 8186
7391, 7393, 7394, 7399, 7402, 7405,	\labelsep	8131
7407, 7411, 7414, 7417, 7419, 7420	\labelwidth	8132
intarray commands:	\langle	7806
\intarray_gset:Nnn	\Large	7935,
..... 7393, 7407, 7420, 7426	8310, 9105, 9106, 9109, 9110, 9270,	
\intarray_gzero:N	9271, 9274, 9275, 9551, 9553, 9806	
\intarray_item:Nn	\lastslide	8328, 8343
7361, 7364,	\LaTeX	22
7373, 7376, 7385, 7394, 7402, 7414	\latex	23
\intarray_new:Nn	\ldots	7811, 7842, 7844, 7846
7355	\leaders	8312
\interpretmod	\leavevmode	8206
137, 3456, 3458, 3460		
interpretmodule (env.)		
137, 3394		
\interpretmodule		
3430, 3432		
\intestfalse		
8484		

<code>\leftmargin</code>	8133	<code>\mcb</code>	8583, 9090, 9112, 9255
<code>\let</code> 1703, 1704, 1713, 1714, 2019, 2095, 2616, 2624, 2632, 2654, 2659, 2910, 3804, 4526, 4547, 5855, 5878, 5895, 5990, 7032, 7847, 7848, 7863, 7864, 8004, 8319, 9211, 9215, 9218, 9222		<code>\mcc</code>	63, 107, 9096, <u>9133</u>
<code>\libinput</code>	19, 75, 132, <u>2050</u> , 2120	<code>\meaning</code>	1908, 2508, 3022, 3025, 3028, 4558, 5038, 5401, 5413, 5414, 6221, 6526, 6681
<code>\libusepackage</code> ...	75, 76, 132, <u>2061</u> , 8071	<code>\medskip</code>	8161, 8176, 8200
<code>\libusetHEME</code>	8070	<code>\meet</code>	54
<code>\libusetikzlibrary</code>	75, 113, 132, <u>2072</u>	<code>\message</code>	28, 1816, 2680, 7896, 7899, 8061, 9751
<code>\linewidth</code> ...	8831, 8838, 8891, 8896, 8939, 8944, 8991, 8996, 9684, 9697	<code>\mhframeimage</code>	103
<code>\list</code>	7536	<code>\mhgraphics</code>	78, 133, <u>2150</u> , 8294, 8296
<code>\LoadClass</code>	16, 7903, 7905	<code>\mhinput</code>	78, 132, <u>2027</u>
<code>\long</code>	8354, 8363	<code>\mhtikzinput</code>	113, 133, <u>2181</u>
<code>\lstinputlisting</code>	78, 133, 2178	<code>\min</code>	70, 111
<code>\lstinputmhlising</code>	78, 133, <u>2166</u>	<code>\min</code>	9057
M		<code>min</code>	69, 105, 110
<code>\macro</code>	115, 116, 118, 127, 136	<code>\mmlarg</code>	121, <u>697</u>
<code>\macroname</code>	30, 119	<code>\mmlint</code>	121, <u>697</u>
<code>\magma</code>	44	<code>\mname</code>	82, 91
<code>\maincomp</code>	30–32, 45, 85, 92, 98, 4528, 4547, 4645, 4646, 4876, 4884, 4889, 4894, 4897, 4906, 5240, 5789, 6176, 6185, 6195, 6196, 6217, 6523, 6583, 7026	mode commands:	
<code>\mainmatter</code>	1725, 1726, 1736, 1737	<code>\mode_if_math:TF</code> 3479, 3804, 3805, 4543, 4548, 9411
<code>\makeatletter</code>	2577, 8536, 9491, 9901	<code>\mode_if_vertical:TF</code>	1487, 1497, 3803
<code>\makeatother</code>	2578, 8553, 9508, 9901	<code>\mode_leave_vertical:</code>	3136
<code>\makebox</code>	9176, 9310	<code>\MSC</code>	7853
<code>\maketitle</code>	131, 1475, 1476, 1567, 1576, 8003, 8004	msg commands:	
<code>\marginnote</code>	8121, 9544	<code>\msg_error:nn</code> 2947, 3128, 5545, 7268, 7317, 8600
<code>\marginpar</code>	144, 6051, 8501, 8524, 8658, 8663, 8751, 8756, 9054, 9059, 9567, 9572, 9586, 9591, 9632, 9637, 9652, 9657	<code>\msg_error:nnn</code> 58, 469, 780, 2074, 2120, 2504, 2751, 2848, 3581, 3611, 4054, 4313, 5008, 5086, 6729, 7036, 7041, 9387
<code>\mathbb</code>	7828	<code>\msg_error:nnnn</code> ..	70, 2055, 2112, 3185, 3261, 3903, 4495, 5140, 5156, 5327, 5354, 5993, 6015, 6454, 7002
<code>\mathbin</code>	31, 7787, 7791, 7798, 7821	<code>\msg_fatal:nn</code> ..	916, 9944, 9950, 9953
<code>\mathclose</code>	31, 6822, 7789, 7796, 7800, 7806, 7808, 7816, 7821, 7825	<code>\msg_fatal:nnnn</code>	920, 2069, 2088
<code>\mathhub</code>	73, 124, 32, <u>822</u>	<code>\msg_none:nn</code>	117
<code>\mathop</code>	31, 7823	<code>\msg_redirect_module:nnn</code>	132
<code>\mathopen</code>	31, 6817, 7789, 7796, 7800, 7806, 7808, 7815, 7821, 7825	<code>\msg_redirect_name:nnn</code>	136
<code>\mathord</code>	31	<code>\msg_set:nnn</code> 114, 9378, 9911, 9915, 9919	
<code>\mathpunct</code>	31, 5570, 6598, 7815	<code>\msg_warning:nn</code>	844
<code>\mathrel</code>	30, 31, 7792	<code>\msg_warning:nnnn</code>	636
<code>\mathrm</code>	6583, 7815	<code>\mult</code>	35, 36, 86, 87
<code>\mathstructure (env.)</code>	91, 148, <u>6853</u>	<code>\multicolumn</code>	9856
<code>\mathstructure</code>	6862, 6863	<code>\multiple</code>	69, 110
<code>\mathtt</code>	7795, 7796, 7831, 7834, 7838	N	
<code>mcb (env.)</code>	1, <u>9062</u>	<code>nassertion (env.)</code>	1
		<code>\Nat</code>	32
		<code>ndefinition (env.)</code>	1
		<code>\neginfprec</code>	36, 85, 86, 147, 4559, 5218, 5229, 6225, 6228, 6237, 6240, 6253, <u>6793</u>

<code>\newcommand</code>	483, 517, 520, 1963, 2041, 2046, 2050, 2061, 2072, 2107, 2173, 2179, 2189, 2199, 7748, 8052, 8121, 8202, 8303, 8307, 8386, 8393, 8396, 8413, 8489, 8512, 8574, 9009, 9143, 9185, 9281, 9319, 9345, 9350, 9355, 9360, 9409, 9464, 9467, 9468, 9469, 9470, 9471, 9472, 9841, 9970, 9976, 9980, 10005, 10010, 10034	<code>\c_notesslides_sectocframes_bool</code>	7920, 8023
<code>\newcounter</code>	7906, 7907, 7908, 7909, 8036, 8048, 8573, 9779	<code>\c_notesslides_topsect_str</code>	7921, 7944, 7947, 8008, 8011, 8012, 8015
<code>\NewDocumentCommand</code>	notesslides internal commands:	
.....	119, 486, 1425, 1781, 1792, 1975, 1980, 1993, 2027, 2516, 5143, 5289, 5849, 5866, 5883, 5897, 5909, 5923, 5946, 5952, 5962, 5972, 7126, 7255, 7301, 7310, 7325, 7659, 7726, 7744, 8070, 8281, 8770, 9824	<code>\c_notesslides_class_str</code>
<code>\NewDocumentEnironment</code>	119	7873, 7876, 7882, 7903
<code>\NewDocumentEnvironment</code>	523, 1510, 1565, 1609, 7713, 8322, 8337	<code>_notesslides_define_chapter:</code>
<code>\newenvironment</code>	1722, 1733, 7625, 7710, 8254, 8256, 9884	<code>_notesslides_define_part:</code>
<code>\newif</code>	648, 672, 2039, 7938, 8472, 8480, 9891	<code>_notesslides_do_label:n</code>	7953, 7989
<code>\newlabel</code>	7863, 7864	<code>_notesslides_do_sectocframes:</code>
<code>\newlength</code>	8073, 8074, 8077	7952, 8024
<code>\newline</code>	8831, 8838, 8891, 8896, 8939, 8944, 8991, 8996, 9684, 9697	<code>_notesslides_do_yes_param:Nn</code>
<code>\newpage</code>	8183, 9471, 9903	8079, 8098, 8101, 8104, 8107, 8110, 8113
<code>\newsavebox</code>	8224	<code>\c_notesslides_dicopt_str</code> ...	7879
<code>nexample (env.)</code>	1	<code>\c_notesslides_document_str</code>
<code>\nextslide</code>	8324, 8339	8051, 8054
<code>\ninputref</code>	8301, 8303, 8307	<code>_notesslides_eat:</code> .	8257, 8261, 8264
<code>\nobreak</code>	7760	<code>_notesslides_excursion_args:n</code>
<code>\noexpand</code>	2674	8407, 8414
<code>\noindent</code>	1540, 7104, 7450, 7584, 7644, 8176, 8196, 8204, 8222, 8623, 8654, 8723, 8832, 8892, 8940, 8992, 9564, 9582, 9685	<code>\l_notesslides_excursion_id_str</code>	8403, 8409
<code>\nointerlineskip</code>	8314	<code>\l_notesslides_excursion_intro-</code> tl	8404, 8408, 8418, 8420
<code>\normalmarginpar</code>	9545	<code>\l_notesslides_excursion_-</code> mhrepos_str	8405, 8410, 8419
<code>\notation</code>	30, 31, 36, 83, 85, 90, 135, 145, 3198, 6114, 7787, 7790, 7791, 7792, 7805, 7807, 7820, 7822, 7823, 7828, 7831	<code>\l_notesslides_frame_allowdisplaybreaks_-</code> bool	8090, 8101
<code>note (env.)</code>	1	<code>\l_notesslides_frame_allowframebreaks_-</code> bool	8089, 8098
<code>notes</code>	69, 100, 105, 110	<code>_notesslides_frame_box_begin:</code>
<code>\notesfalse</code>	7950	8154, 8165, 8189
notesslides commands:		<code>_notesslides_frame_box_end:</code>
<code>\c_notesslides_notes_bool</code>	8159, 8175, 8191
...	7877, 7878, 7891, 7894, 7902, 7918, 7919, 7931, 7939, 8067, 8247, 8253, 8287, 8302, 8352, 8387, 8397	<code>\l_notesslides_frame_fragile_-</code> bool	8091, 8104
		<code>\l_notesslides_frame_label_str</code>
		8088, 8096, 8185, 8186
		<code>\l_notesslides_frame_shrink_-</code> bool	8092, 8107
		<code>\l_notesslides_frame_squeeze_-</code> bool	8093, 8110
		<code>\l_notesslides_frame_t_bool</code>
		8094, 8113
		<code>_notesslides_inputref:</code>
		8300, 8301, 8304
		<code>_notesslides_notes_env:nnnn</code>
		8252, 8272, 8273, 8274, 8275, 8276, 8277
		<code>\l_notesslides_num</code>
		7956, 7958, 7961,

7963, 7966, 7967, 7970, 7971, 7974, 7975, 7978, 7979, 7982, 7983, 7990	\PDF 16, 24
_notesslides_setup_itemize: 8117, 8188	\pdfbookmark 7990
\l_notesslides_slideshow_ counter_int 8336, 8338, 8340, 8341, 8344, 8345	peek commands:
\l_notesslides_tmp 8436, 8441	\peek_charcode:NTF 1350, 3530, 4552, 4565, 4625, 4627, 4654, 4660, 4803, 4814, 5166, 5173
\g_notesslides_variables_prop 8428, 8430, 8433, 8436	\peek_charcode_remove:NTF 4551, 4624, 4659, 4912, 5162, 5164, 5171, 5181, 5184, 5385, 5428
\notesslidesfont 8157, 8173, 8319	\phantom 9455
\notesslidesfooter 8161, 8176, 8317	\plus 34–37, 85–87
\notesslidestitleemph .. 8196, 8198, 8309	\precondition 8489
\notesttrue 7940	\prematuarestop 103, 8051
nparagraph (env.) 1, 1	\premise 96, 149, 7237, 7325
nsproof (env.) 1	prg commands:
\null 7760	\prg_new_conditional:Nnn 57, 62, 263, 272, 680, 2481, 2485, 2489, 2603, 3993, 3996, 4167, 5333, 5619, 5629, 5635, 6029, 6040, 6044
\number 69, 111	\prg_new_protected_conditional:Nnn 279
\numberline 7988	\prg_return_false: 60, 65, 264, 267, 273, 275, 295, 299, 682, 2483, 2487, 2491, 2604, 3994, 4005, 5351, 5355, 5625, 5626, 5627, 5632, 5633, 5638, 5639, 6030, 6045
\numberproblemsin 8573	\prg_return_true: 60, 65, 265, 267, 275, 299, 682, 2483, 2487, 2491, 2604, 3994, 4005, 5349, 5625, 5632, 5638, 6041
O	\printexcursion 104
\objective 8512	\printexcursions . 8385, 8394, 8415, 8423
\OMDoc 24	problem (env.) 1
\omdoc 16	problem commands:
\only 8341, 8345	\g_problem_id_counter 8778, 8791, 8793, 8854, 8856, 8902, 8904, 8952, 8954
\oplus 7797, 7798	\l_problem_inputproblem_keys_tl 8602, 8603, 8605, 8772
P	\problem_mcc_box_default_tl 9168, 9176, 9180, 9182
\PackageError 8437	\problem_mcc_box_tl 9069, 9073, 9105, 9109, 9137, 9173
\pagebreak 8671, 9822	\problem_scc_box_default_tl 9306, 9310, 9314, 9316
\pagenumbering ... 1709, 1719, 1730, 1741	\problem_scc_box_tl 9234, 9246, 9270, 9274, 9280, 9307
\par 47, 1538, 1542, 7424, 7450, 7489, 7495, 7535, 7544, 7580, 7584, 7626, 7631, 7640, 7705, 7760, 8161, 8176, 8196, 8204, 8209, 8217, 8222, 8229, 8239, 8654, 8667, 8670, 8701, 8831, 8838, 8891, 8896, 8939, 8944, 8991, 8996, 9042, 9045, 9102, 9223, 9267, 9564, 9575, 9578, 9582, 9612, 9620, 9684, 9697, 9740, 9743, 9805, 9810, 9820, 9822	\g_problem_total_min_fp .. 8592, 8649
\paragraph 26, 77	\g_problem_total_pts_fp .. 8591, 8648
\parent 117	\problemheader ... 8654, 8674, 9564, 9582
\parsep 7538, 8746, 9020, 9039, 9067, 9232, 9626, 9645, 9722	problems internal commands:
\part 26, 77, 8044, 8045	_problems_activate_macros: 8581, 8646, 8703
\partname 7958, 8041, 8042	
\parttitlename 7958	
\PassOptionsToClass 7881, 7882	
\PassOptionsToPackage 10, 7883, 7884, 7895, 7898, 7923, 7924, 7941, 8469, 9485	
\pause 8202	

\l__problems_anscls_int	__problems_record_problem:
..... 8947, 9012, 9014, 9703, 9705 8650, 8688
\c__problems_boxed_bool	__problems_sccline:n 9229,
..... 8465	9248, 9251, 9270, 9274, 9283, 9334
__problems_do_yes_param:Nn .. 9113	__problems_solution_start:n ...
__problems_exnote_start:n 8788, 8806, 8809
..... 8899, 8916, 8919	\c__problems_solutions_bool
\c__problems_false_tl ... 9376, 9386 8459, 8475
\l__problems_fillin_solution_tl .	\g__problems_subproblem_int
.. 9366, 9390, 9402, 9431, 9442, 9449 8642, 8695, 8733,
__problems_fillinsol:nn	8748, 9628, 9629, 9647, 9648, 9671
..... 9412, 9414, 9420, 9437	\c__problems_test_bool 8467, 8481,
__problems_gnote_start:n	8671, 9465, 9467, 9471, 9541, 9761
..... 8949, 8968, 8971	\c__problems_true_tl 9375, 9385, 9395
\c__problems_gnotes_bool	\ProcessKeysOptions
..... 8455, 8970, 8981, 9525, 9752 43, 7887, 7927, 8474, 9488, 9999
\l__problems_has_min_bool . 8596,	\ProcessOptions
8640, 8641, 8737, 8755, 9636, 9656 11
\l__problems_has_pts_bool	\prod
..... 8595, 8637,	prop commands:
8638, 8734, 8750, 9631, 9651, 9673	\prop_clear:N
__problems_hint_start:n 5013
..... 8851, 8868, 8871	\prop_const_from_keyval:Nn . 426, 439
\c__problems_hints_bool	\prop_gc_clear:N
..... 8457, 8870, 8881	2310, 2368, 2369, 2370, 2433, 5249
\l__problems_in_problem_bool ...	\prop_get:NnNTF
.. 8594, 8597, 8599, 8633, 8699, 8704	465, 5334, 8436
__problems_maybe_inline:n	\prop_gput:Nnn
..... 9074, 9100, 9209, 9238, 9265	... 2458, 2913, 3962, 5278, 5313,
__problems_maybe_newline:	5319, 5337, 6340, 7292, 8430, 9837
..... 9134, 9213, 9220	\prop_gset_from_keyval:Nn
__problems_mccline:n 9064, 2426, 2427, 5262
9083, 9086, 9105, 9109, 9145, 9200	\prop_if_exist:NTF ... 914, 1141, 5261
\c__problems_min_bool 8463, 8661,	\prop_if_in:NnTF
8754, 9058, 9570, 9589, 9635, 9655	... 222, 1170, 4970, 5045, 5059, 5081
\l__problems_min_tl	\prop_item:Nn
8635, 8639, 8649, 8662, 8663, 8690,	3158, 3997,
8706, 8738, 9571, 9572, 9590, 9591	4962, 5005, 5084, 6983, 7294, 8433
\c__problems_notes_bool	\prop_map_break:n
..... 8453, 8918, 8929	4012, 4015, 4120, 4325, 4328, 6362
__problems_oldpar:	\prop_map_function:NN
..... 9211, 9215, 9218, 9222	2400
__problems_parse_fillin_-	\prop_map_inline:Nn
arg:nnnn	2428, 2990, 3236, 3972, 4018, 4036,
.. 9370, 9371, 9372, 9383, 9401, 9423	4039, 4139, 4152, 4158, 4318, 4842,
\l__problems_path_seq	6351, 6359, 6471, 6474, 6956, 9849
.. 8614, 8615, 8709, 8710, 8714, 8715	\prop_new:N
\l__problems_prob_imports_tl . 8570	4194, 8428, 9831
\l__problems_prob_refnum_int . 8571	\prop_put:Nnn
\c__problems_pts_bool 8461, 8656,	4235, 4383,
8749, 9053, 9565, 9584, 9630, 9650	5047, 5061, 5090, 6978, 6989, 9844
\l__problems_pts_tl .. 8634, 8636,	\prop_remove:Nn
8648, 8657, 8658, 8690, 8705, 8735,	3624
9566, 9567, 9585, 9586, 9607, 9609	\prop_set_eq:NN
	1043
	\prop_to_keyval:N
	... 2380, 2381, 2382, 2388, 2392, 5263
	\protect
	7988, 8439
	\protected .. 1476, 5879, 5896, 5958, 5968
	\providecommand .. 2157, 2164, 8059, 10004
	\ProvidesExplClass
	5, 7870
	\ProvidesExplPackage
 20, 7914, 8448, 9476, 9990

\pts 9052
pts 69, 105, 110

Q

\qrcode 9544
\qgrid 9564, 9582, 9629, 9648, 9762
\qrjson
 9526, 9528, 9530, 9531, 9596, 9601,
 9608, 9616, 9619, 9661, 9663, 9668,
 9674, 9679, 9688, 9694, 9709, 9713
\qrschema 9539, 9544
\quad 7967, 7971, 7975,
 7979, 7983, 9410, 9416, 9813, 9816
quark commands:
 \q_no_value 1167
 \quark_new:N 2440, 2650
quark internal commands:
 \q_stex_smsmode_break
 2650, 2653, 2687
\quizdata 9980
quizdata 9905

R

\raisebox 9178, 9312
\rangle 7806
\realize 55, 56
\ref 79, 1819
\refstepcounter 8610, 9792
\relax 19, 20, 46, 604, 605,
 741, 742, 1704, 1714, 1933, 2005,
 2094, 2435, 2436, 2657, 2659, 2673,
 2674, 2681, 3804, 4515, 4546, 4547,
 5132, 5977, 5998, 8004, 8288, 8290
\renamedecl 56, 137, 3204, 3540
\renewcommand 8068, 8183, 8193, 9700, 9760
\renewenvironment
 .. 8122, 8180, 8203, 8225, 8248, 8250
\reqpts 70, 111
\requiremodule 88, 89, 3200
\RequirePackage
 4, 13, 19, 24, 25, 474, 477,
 7871, 7889, 7911, 7915, 7929, 7942,
 7943, 8279, 8449, 8487, 9477, 9489,
 9516, 9517, 9991, 10002, 10007, 10008
\resizebox 9854, 10016, 10022, 10026
\reversemarginpar 9544
\rhd 8124, 8127
\Rightarrow 7725
\rightmargin 7539, 8747, 9021,
 9040, 9068, 9233, 9627, 9646, 9723
\rule 8831, 8838, 8891, 8896,
 8939, 8944, 8991, 8996, 9684, 9697
rustex commands:
 \rustex_direct_HTML:n
 8210, 8218, 8230, 8240

\rustex_if:TF 8208, 8209,
 8216, 8217, 8228, 8229, 8238, 8239
\rustexBREAK 2581

S

sassertion (env.) 94, 149, 7228
scb (env.) 9208
\scb 8584, 9091, 9256, 9277
\scc 9261, 9278
\scriptsize 8121
\scriptstyle 8127
sdefinition (env.) 94, 149, 7223
\section 25, 26, 77, 131
\sectiontitleemph 7934, 7993
sectocframes 100
seq commands:
 \seq_clear:N 142, 180,
 456, 846, 849, 2020, 2122, 2617,
 2970, 2971, 2972, 2973, 2989, 3113,
 3210, 4025, 4740, 5014, 5015, 5023,
 5586, 5654, 5719, 5728, 6180, 6460,
 6593, 6743, 6889, 6924, 7049, 7084
 \seq_count:N .. 2063, 2085, 3495, 7196
 \seq_gclear:N 2610, 2611, 5250
 \seq_get_right:NN ... 8615, 8710, 8715
 \seq_gpush:Nn 64, 786
 \seq_gput_right:Nn .. 2575, 2595, 5279
 \seq_gset_eq:NN
 244, 252, 3009, 3010, 3011, 3012
 \seq_gset_split:Nnn 5268
 \seq_if_empty:NTF 178, 196, 264, 273,
 299, 915, 1162, 1235, 1276, 2054,
 2111, 4097, 7146, 7462, 7634, 7695
 \seq_if_empty_p:N 286, 287, 2130
 \seq_if_exist:NTF 5267
 \seq_if_in:NnTF 63, 785, 2134, 2145,
 2498, 2613, 2732, 2740, 2991, 3184,
 3234, 4034, 4854, 4973, 5101, 6469
 \seq_item:Nn 265, 266,
 274, 2064, 2086, 5335, 6290, 6296, 7197
 \seq_map_break: 929, 1061, 4012
 \seq_map_break:n 3098,
 3106, 3284, 3312, 3320, 4015, 4120
 \seq_map_function:NN 182, 186, 2057,
 2114, 3039, 4752, 7463, 7635, 7696
 \seq_map_indexed_function:NN . 7147
 \seq_map_inline:Nn 851, 927, 1029,
 1033, 1182, 2125, 3031, 3052, 3084,
 3173, 3253, 3257, 3314, 3322, 4017,
 4136, 5729, 6306, 6744, 6771, 6899,
 6941, 7055, 7212, 7231, 7432, 7555
 \seq_new:N
 24, 746, 2229, 2572, 2592, 2607
 \seq_pop:NN 179

<code>\seq_pop_left:NN</code>	9231, 9232, 9233, 9625, 9626, 9627, 9644, 9645, 9646, 9721, 9722, 9723
..... 163, 292, 293, 1033, 1183, 1185, 2138, 3497, 3499, 3506, 3510	<code>\setlicensing</code> 102
<code>\seq_pop_left:NNTF</code>	<code>\setmetatheory</code> 135, 2516, 2583
..... 988, 6265, 6267, 6275	<code>\setnotation</code> 31, 86, 145, 6425
<code>\seq_pop_right:NN</code>	<code>\setsectionlevel</code> 26, 77, 131, 1633, 7945, 7947, 8009, 8011
. 197, 210, 212, 217, 219, 221, 948, 1159, 1161, 1166, 1234, 1274, 1277, 2126, 2798, 4096, 4133, 4743, 6609	<code>\setSGvar</code> 104, 8429, 8439
<code>\seq_push:Nn</code> 185, 847, 2615, 2992, 3558	<code>\setslidelogo</code> 102
<code>\seq_put_left:Nn</code>	<code>\setsource</code> 102
164, 183, 2373, 4035, 5046, 5060, 6470	<code>sexample (env.)</code> 94, 149, 7243
<code>\seq_put_right:Nn</code> 200, 213, 223, 226, 235, 467, 858, 1171, 2135, 2139, 2146, 2313, 2500, 2980, 3001, 3116, 3133, 3235, 3484, 5082, 5642, 5659, 5693, 5700, 5725, 5731, 5735, 5742, 6248, 6255, 6276, 6278, 6595, 6746, 6748, 6894, 6929, 7052, 7087, 7177	<code>\sf</code> 8310
<code>\seq_reverse:N</code> 4744	<code>\sffamily</code> 8319
<code>\seq_set_eq:NN</code>	<code>sfragment (env.)</code> 77, 131, 1504
..... 188, 209, 216, 234, 280, 281, 1032, 1158, 1181, 5740, 6753, 6781	sfragment commands:
<code>\seq_set_split:Nnn</code>	<code>_sfragment_do_level:nn</code>
.... 143, 211, 218, 850, 939, 940, 987, 1030, 1160, 1180, 1233, 1273, 2123, 2124, 2797, 3494, 3504, 4095, 4132, 4742, 6264, 6268, 6304, 6608 1520, 1524, 1528, 1529, 1530, 1531, 1532, 1536, 1548, 7986
<code>\seq_use:Nn</code> 206, 213, 226, 472, 475, 478, 864, 921, 990, 1175, 1239, 2053, 2110, 2128, 2799, 3502, 3508, 3512, 4101, 4134, 5269, 5569, 6510, 6598, 6720, 6754	<code>_sfragment_end:</code> 1515, 1544, 1561, 1583
<code>\l_tmpa_seq</code> 456, 467, 472, 475, 478, 845, 847, 850, 851, 939, 943, 1030, 1031, 1033, 1160, 1161, 1162, 1166, 1171, 1175, 1180, 1182, 1273, 1274, 1276, 1277, 1330, 1335, 2797, 2798, 2799, 4132, 4133, 4134, 4740, 4742, 4743, 4744, 4752, 6304, 6306, 6593, 6595, 6598, 6608, 6609, 6743, 6746, 6748, 6753, 6781	<code>\shorttitle</code> 1571, 1572
<code>\l_tmpb_seq</code> 1032, 1033, 1034, 1181, 1183, 1185, 1186	<code>\skipfragment</code> 77, 131, 1617
<code>\seqmap</code> 91, 146, 285, 5631, 6732	<code>\slideframewidth</code> . 8077, 8078, 8167, 8288
<code>\setbox</code> ... 136, 1851, 1932, 7455, 7592, 8250, 8265, 8814, 8874, 8922, 8973	<code>\slideheight</code> 8074
<code>\setcounter</code> 9790, 9793	<code>slides</code> 100
<code>\setkeys</code> ... 2161, 2177, 2194, 8292, 10011	<code>\slidewidth</code>
<code>\setlength</code> 7537, 7538, 7539, 8073, 8074, 8078, 8131, 8132, 8133, 8745, 8746, 8747, 9019, 9020, 9021, 9038, 9039, 9040, 9066, 9067, 9068,	.. 8073, 8170, 8206, 8288, 8290, 8294
	<code>\smacro</code> 87, 88
	<code>\small</code> 7764
	<code>\smallskip</code> 7760, 8658, 8663, 8751, 8756, 8831, 8891, 8939, 8991, 9567, 9572, 9586, 9591, 9632, 9637, 9652, 9657, 9684, 9810
	<code>smodule (env.)</code> 81, 134, 2230
	<code>\smodule</code> 3201
	<code>\Sn</code> 19, 84, 5895
	<code>\sn</code> 19, 23, 84, 144, 1420, 5856
	<code>\Sn\Sns</code> 144, 5856
	<code>\Sns</code> 19, 84, 5896
	<code>\sns</code> 19, 84, 144, 5856
	<code>\solution</code> 61
	<code>solution (env.)</code> 1, 8778
	<code>\solution</code> 8582, 8841, 9092, 9257
	<code>solutions</code> 69, 105, 110
	<code>\solutionsfalse</code> 8478, 8846
	<code>\solutionstrue</code> 8476, 8843
	<code>\source</code> 116
	<code>sparagraph (env.)</code> 94, 149, 7246
	<code>spfblock (env.)</code> 150, 7713
	<code>\spfblock</code> 7448, 7719
	<code>\spfjust</code> 150, 7449, 7748
	<code>\spfsketch</code> 150, 7626
	<code>spfsketchenv (env.)</code> 150, 7625
	<code>\spfsketchenvautorefname</code> 7644
	<code>\spfstep</code> 150, 7443, 7723

<code>\spfstepautorefname</code>	7547, 7711	<code>\stex_annotate:nn</code>	120, 121,
<code>spfstepenv</code> (env.)	150, <u>7710</u>	694, 699, 702, 1488, 1498, 2042,	
<code>\spfstepenv</code>	7666, 7728	2043, 3136, 3138, 3479, 3909, 3930,	
<code>\spfstepenvautorefname</code>	7711	3933, 3940, 3945, 3947, 4283, 4286,	
<code>sproblem</code> (env.)	8581	4293, 4297, 4300, 4427, 4430, 4437,	
<code>\sproblemautorefname</code>	8680	4441, 4444, 4580, 4766, 4775, 4780,	
<code>sproof</code> (env.)	96, 150, <u>7422</u>	4999, 5520, 5553, 5668, 5822, 6506,	
<code>\sproofautorefname</code>	7512	6513, 6519, 6520, 6529, 6540, 6550,	
<code>\sproofend</code>	150, 7514, 7531, <u>7752</u>	7105, 7132, 7279, 7319, 7330, 7450,	
<code>\square</code>	7753, 9138, 9169	7479, 7482, 7572, 7584, 7745, 7749,	
<code>\sr</code>	19, 23, 83, 144, <u>5849</u>	8196, 8379, 8623, 8677, 8723, 9022,	
<code>\sref</code>	79, 80, 94, 1792, 8389	9030, 9054, 9059, 9147, 9151, 9159,	
<code>\sreflabel</code>	79, 81, <u>1763</u>	9285, 9289, 9297, 9391, 9426, 9726	
<code>\srefsetin</code>	79, 1963	<code>\stex_annotate_env</code>	2255, 9239
<code>\srefsym</code>	84, 1980	<code>_stex_annotate_force_break:n</code>	
<code>\srefsymuri</code>	84, 1982, 1985	121, <u>697</u> , 1541, 1603, 3137,
<code>\startsolutions</code>	106, <u>8842</u>	3478, 3927, 3945, 4280, 4298, 4425,	
<code>\stepcounter</code>	1619, 1623, 1627,	4442, 4779, 4915, 4993, 5525, 5552,	
	1628, 1629, 1630, 1631, 7987, 8182	5556, 5568, 5673, 6506, 7105, 7133,	
<code>\sTeX</code>	14, 16, 76	7280, 7320, 7450, 7453, 7474, 7591,	
<code>\stex</code>	16, 23, 76	8623, 8627, 8723, 8730, 9397, 9430	
stex commands:		<code>\stex_annotate_invisible:n</code>	
<code>\l_stex_aB_args_seq</code>	284,	121, 694, 695, 1613, 2263, 2549,
	5569, 5586, 5642, 5654, 5659, 5693,		2962, 3464, 3908, 5291, 6506, 7477
	5700, 5719, 5725, 5729, 5740, 5742,	<code>\stex_annotate_invisible:nn</code>	
	6720, 6744, 6753, 6754, 6771, 6781	121, <u>694</u> , 1463, 1658, 1664, 1670,
<code>\stex_activate_module:N</code>			2856, 2886, 3477, 3551, 3591, 3601,
	135, 2339, 2357, <u>2493</u> , 2493, 2754, 2761		4261, 4409, 5469, 5487, 5505, 6901,
<code>\stex_activate_module:n</code>			6943, 7057, 7305, 8505, 8528, 9734
	2328, <u>2493</u> , 2494, 2497, 2881, 5024,	<code>_stex_apply_patch_begin:n</code>	
	6907, 6949, 6951, 6961, 6976, 7060	525, 541, 3340, 3406
<code>_stex_activate_notations:</code>	2507, 6349	<code>_stex_apply_patch_end:n</code>	
<code>_stex_activate_symbols:</code>	2506, 3970	531, 556, 3345, 3411
<code>_stex_add_definiens:nn</code>		<code>\stex_archive_base:N</code>	
	124, <u>869</u> , 869, 1048
	<u>7277</u> , 7277, 7435, 7437, 7558, 7560	<code>\stex_archive_base:n</code>	
<code>\stex_add_morphism:nnnn</code>	137, 2883,	124, <u>869</u> , 873, 1087, 1096, 1202, 1371
	2912, 2912, 2917, 3035, 6904, 6946	<code>\stex_archive_id:N</code>	124, <u>866</u> ,
<code>\stex_add_notation:nnnnn</code>			866, 888, 1041, 1045, 1048, 1049,
	145, 3064, 6320, <u>6327</u> , 6334, 6347, 6427		1054, 1055, 1143, 1319, 1966, 2124
<code>\stex_add_symbol:nnnnnnN</code>	139, <u>3954</u>	<code>\stex_archive_path:N</code>	
<code>\stex_add_symbol:nnnnnnnN</code>	124, <u>876</u> , 876, 1069, 1179, 2123
	<code>\stex_archive_path:n</code>	
	140, 3088,	124, <u>876</u> , 880, 1079
	3093, 3754, 3822, 3954, 6875, 7183	<code>_stex_args_end:</code>	5253,
<code>\l_stex_all_module_seq</code>	5023		5277, 5280, 5985, 6009, 6020, 6024
<code>\l_stex_all_modules_seq</code>		<code>_stex_assign_do:n</code>	
	3151, 3464, 3466, 3473, 3525
	134, 2020, <u>2229</u> , 2313,	<code>\l_stex_assoc_args_count</code>	
	2373, 2498, 2500, 2617, 4017, 4136	3854, 3858, 3900, 3901
<code>\l_stex_allow_semantic_bool</code>		<code>\l_stex_brackets_dones_bool</code>	
	6648, 6792, 6798, 6799, 6816
	143, 4486,	<code>\stex_check_term:nn</code>	
	4507, 4508, 4510, 4577, 4847, 4879,	144, 2619, 3475, <u>6049</u> , 6050, 6058,
	4904, 4946, 5128, 5152, 5239, 5248,		
	5363, 5464, 5470, 5488, 5506, 5539,		
	5662, 5745, 5781, 5978, 6001, 7017		

6064, 6068, 6072, 6077, 6497, 7769	453, 458, 459, 1164, 1168, 1192,
\stex_check_terms: ... 144, 3753,	2257, 2791, 2793, 2801, 8619, 8719
3817, 4206, 4365, 6060, 6061, 6081	\l_stex_current_module_str
\c_stex_check_terms_bool 2555, 3584, 3593, 3603
..... 36, 6036, 6039, 7771	\l_stex_current_module_uri
\stex_close_module:	134, 1250, 1379, 2021, 2229, 2256,
... 134, 2283, 2375, 2375, 2556, 7849	2277, 2312, 2344, 2361, 2372, 2378,
\stex_css_link:n	2379, 2380, 2381, 2382, 2387, 2389,
121, 697	2393, 2398, 2400, 2476, 2477, 2482,
\stex_css_literal:n	2502, 2503, 2508, 2509, 2511, 2533,
..... 121, 81, 697, 803, 1679, 8141	2618, 2913, 3955, 3962, 3971, 3972,
\l_stex_current_*	3984, 3989, 6338, 6340, 6350, 6351,
141, 142	6359, 6956, 7178, 7190, 7284, 7847
\l_stex_current_archive	\l_stex_current_ns_uri
.... 124, 125, 887, 888, 897, 908,	2243
1029, 1068, 1069, 1141, 1143, 1178,	\l_stex_current_redo_tl
1179, 1317, 1319, 2119, 2123, 2124	4513, 4518, 4525, 4535, 4536, 4539,
\l_stex_current_args_tl	4711, 4749, 4785, 4941, 5135, 5405
.. 4530, 5252, 5253, 5256, 6573, 6588	\l_stex_current_return_tl
\l_stex_current_arity_str 4531, 4547, 5193, 5205, 5225, 5410
..... 4529, 4597, 4612,	\l_stex_current_section_level_-
4614, 5224, 5373, 5374, 5409, 6594	int
\l_stex_current_def_uri	131, 1483, 1519,
3125, 3127, 3130, 3134, 3136, 3142,	1521, 1525, 1539, 1556, 1557, 1586,
7168, 7190, 7191, 7197, 7198, 7265,	1588, 1591, 1594, 1595, 1602, 1618,
7267, 7270, 7272, 7314, 7316, 7319	1620, 1624, 1635, 1636, 1637, 1638,
\l_stex_current_display_tl	1639, 1640, 1642, 1646, 1649, 1656,
..... 141, 4490,	1660, 1666, 1670, 1675, 7990, 7997
5131, 5147, 5150, 5536, 5656, 5871,	\l_stex_current_section_level_-
5875, 5888, 5892, 5903, 5917, 5931,	str 131, 1483, 1490, 1500, 1559, 7998
5955, 5965, 6003, 6193, 6608, 7024	\l_stex_current_symbol_args_tl ..
\l_stex_current_doc_uri 141, 142, 262
7150	\l_stex_current_symbol_arity_int
\l_stex_current_document_uri 141, 142, 262
..... 117, 127, 136, 253, 254,	\l_stex_current_symbol_invoke_cs 141
259, 1128, 1178, 1252, 1272, 1291,	\l_stex_current_symbol_name_str .
1295, 1300, 1302, 1409, 2351, 7780 5981, 5982, 5983, 5984
\l_stex_current_domain_str ... 3619	\l_stex_current_symbol_return_tl
\l_stex_current_domain_uri 141, 142, 262
... 2919, 2932, 2936, 2938, 2943,	\l_stex_current_symbol_type_tl ..
2951, 2959, 2975, 2979, 3008, 3037, 141, 142, 262
3062, 3357, 3382, 3424, 3448, 3568	\l_stex_current_symbol_uri
\g_stex_current_file 141, 142,
.. 117, 244, 252, 257, 1071, 1081,	262, 283, 3682, 3695, 3696, 3701,
1331, 2348, 2814, 8614, 8709, 8714	3713, 3749, 3764, 3773, 3778, 3781,
\l_stex_current_full_tl 141,	3787, 3812, 3833, 4488, 4489, 4491,
4489, 4498, 4537, 4542, 4557, 4570,	5534, 5979, 5980, 5981, 6817, 6822
4582, 4710, 4793, 4919, 4951, 4995,	\l_stex_current_term_tl 4512, 4533,
5130, 5140, 5146, 5149, 5156, 5191,	4786, 4792, 4990, 5153, 5435, 5451,
5192, 5203, 5216, 5217, 5246, 5327,	5465, 5471, 5486, 5489, 5504, 5507,
5354, 5522, 5535, 5655, 5705, 5746,	5537, 5663, 5664, 5665, 5748, 5988
5747, 5824, 5827, 5870, 5887, 5902,	\stex_current_this: . 4526, 7016, 7032
5916, 5930, 5980, 5993, 6002, 6012,	\l_stex_current_this_tl
6013, 6015, 6192, 6528, 6539, 7019 6841, 6844, 7018, 7026
\l_stex_current_language_str ...	\l_stex_current_type_tl
..... 117, 119, 254, 258, 4532, 4579, 4652, 4707,

4715, 4718, 4722, 5024, 5025, 5100
`\stex_deactivate_macro:Nn`
 115, 67, 67, 2539, 2873,
 3195, 3196, 3197, 3198, 3199, 3200,
 3201, 3364, 3370, 3389, 3430, 3436,
 3456, 3471, 3546, 3615, 3676, 3707,
 3796, 5950, 5959, 5969, 5975, 6138,
 6862, 6917, 6967, 7275, 7309, 7324,
 7332, 7623, 7687, 7719, 7743, 7747,
 7751, 8841, 8898, 8946, 8998, 9051,
 9090, 9091, 9092, 9093, 9094, 9095,
 9112, 9207, 9255, 9256, 9257, 9258,
 9259, 9260, 9277, 9342, 9463, 9748
`\stex_debug:nn`
 116, 103, 103, 133, 137,
 166, 173, 237, 472, 547, 603, 832,
 837, 865, 907, 918, 965, 1002, 1036,
 1044, 1065, 1135, 1194, 1272, 1310,
 1323, 1338, 1347, 1455, 1803, 1828,
 1830, 1839, 1843, 1882, 1888, 1902,
 1905, 1908, 1931, 2022, 2053, 2110,
 2128, 2132, 2143, 2296, 2303, 2305,
 2333, 2336, 2338, 2341, 2353, 2361,
 2363, 2365, 2377, 2425, 2477, 2499,
 2508, 2513, 2517, 2519, 2620, 2673,
 2694, 2698, 2716, 2733, 2741, 2815,
 2830, 2832, 2854, 2933, 2951, 2993,
 3021, 3058, 3073, 3130, 3149, 3250,
 3306, 3474, 3493, 3496, 3500, 3520,
 3524, 3549, 3569, 3585, 3589, 3599,
 3620, 3622, 3695, 3955, 3971, 3984,
 4094, 4135, 4137, 4156, 4173, 4175,
 4212, 4334, 4542, 4558, 4651, 4895,
 5022, 5038, 5161, 5180, 5192, 5194,
 5197, 5203, 5216, 5238, 5246, 5361,
 5401, 5412, 5582, 5597, 5599, 5606,
 5609, 6062, 6066, 6070, 6074, 6221,
 6252, 6263, 6335, 6350, 6358, 6360,
 6363, 6499, 6526, 6627, 6640, 6681,
 6802, 6974, 7048, 7191, 7195, 7198,
 7200, 7270, 7291, 7327, 7433, 7556
`\c_stex_debug_clist` 30,
 42, 104, 107, 121, 125, 127, 131, 135
`\c_stex_default_metatheory`
 2019, 2616, 7848
`\l_stex_default_notation`
 ... 4572, 5206, 5211, 5231, 5689,
 6576, 6577, 6583, 6592, 6597, 6600
`_stex_definiens_impl:nn`
 3207, 7257, 7262
`\stex_do_default_notation:`
 ... 146, 4571, 5204, 5688, 6571, 6571
`\stex_do_default_notation_op:` ...
 146, 5228, 6571, 6572, 6581
`_stex_do_deprecation:n` 634, 634, 3836
`_stex_do_for_list:` ... 149, 3205,
7083, 7083, 7167, 7428, 7551, 7629
`_stex_do_id:` 639, 639,
 1513, 1580, 7112, 7130, 7441, 7569,
 7578, 7586, 7630, 7670, 8645, 9801
`\stex_do_up_to_module:n`
 134, 2471, 2471,
 2474, 2478, 3965, 6342, 6960, 7007
`\g_stex_document_kind_parameters_-`
`tl` 1437, 1443, 9959
`\g_stex_document_kind_str`
 14, 1436, 1438, 1442, 9948
`\stex_document_uri:n` 126
`\stex_document_uri_archive:N` ...
 126, 1106, 1106, 1295
`\stex_document_uri_element:N` ...
 127, 1118, 1118
`\stex_document_uri_from_archive_-`
`file:Nn`
 ... 127, 1133, 1133, 1827, 1995, 2029
`\stex_document_uri_language:N` ...
 127, 254, 1115, 1115, 1192
`\stex_document_uri_name:N`
 127, 1112, 1112, 1300, 1302
`\stex_document_uri_path:N`
 127, 1109, 1109, 1291
`\stex_document_uri_with_language:Nn`
 127, 1121, 1121, 2350
`_stex_eat_exclamation_point:` ...
 142, 3842, 5195, 5206, 5427, 5427, 5428
`_stex_end:` 414, 422
`\stex_end:` 7039, 7045
`\stex_every_module:n` 134,
2467, 2468, 2540, 2874, 3365, 3371,
 3390, 3431, 3437, 3457, 3677, 3708,
 3797, 6139, 6863, 6918, 6969, 7013
`\g_stex_every_module_tl`
 2294, 2467, 2469, 7785
`\l_stex_every_symbol_tl` 4467, 4469,
 4470, 4476, 4477, 4480, 4509, 4535
`\stex_execute_in_module:n`
 134, 2318, 2475, 2475,
 2480, 2880, 3780, 6906, 6935, 6948
`\l_stex_feature_name_str` .. 2954,
 3036, 3072, 3073, 3079, 3262, 3561
`\stex_file_in_smsmode:Nn`
 136, 2354, 2606, 2609, 2846
`\stex_file_resolve:Nn` 116, 147, 153,
 169, 176, 233, 243, 251, 852, 854, 1330
`\stex_file_set:Nn`
 116, 141, 141, 146, 160, 171, 845, 1134
`\stex_file_split_off_ext:NN`
 116, 208, 208

<code>\stex_file_split_off_lang:N</code> ...	<code>\l_stex_get_symbol_uri</code> .
116, <u>215</u> , 215, 2348, 8614, 8709, 8714	140, 1420,
<code>\stex_file_use:N</code> ...	1982, 3117, 3125, 3148, 3155, 3251,
116, 166, 173,	3256, 3260, 3276, 3297, 3474, 3477,
<u>205</u> , 205, 237, 825, 845, 847, 855,	3485, 3549, 3552, 3559, 3561, 3696,
859, 931, 965, 966, 968, 981, 1034,	3773, 4059, 4075, 4081, 4121, 4182,
1071, 1081, 1135, 1150, 1331, 1335,	4456, 5870, 5872, 5887, 5889, 5979,
1751, 2131, 2142, 2353, 2355, 2814	6002, 6004, 6122, 6126, 6144, 6145,
<code>\l_stex_fors_seq</code>	6321, 6432, 6435, 6443, 6444, 6445,
149, 3113,	6446, 6451, 6455, 7088, 7265, 7314,
3116, 7084, 7087, 7146, 7147, 7177,	8491, 8502, 8506, 8514, 8525, 8529
7196, 7197, 7212, 7231, 7432, 7462,	<code>\stex_get_var:n</code>
7463, 7555, 7634, 7635, 7695, 7696	140, 4309,
<code>\stex_get_env:Nn</code>	4309, 5900, 5912, 5926, 6154, 7303
.	<code>\l_stex_get_variable_str</code>
116, <u>86</u> , 87, 98, 119, 229,	. . .
231, 239, 241, 568, 574, 652, 823, 6033	4310, 4312, 4336, 5902, 5903,
<code>\stex_get_in_morphism:n</code> <i>137</i> , 3115,	5914, 5916, 5917, 5928, 5930, 5931,
3124, <u>3249</u> , 3249, 3462, 3518, 3541	6007, 6008, 6155, 6160, 6163, 7305
<code>\stex_get_mathstructure:n</code>	<code>\stex_has_definiens:N</code>
148, 2935, 6891, 6925, <u>7033</u> , 7033, 7047	3993
<code>\l_stex_get_structure_module_uri</code>	<code>\stex_has_definiens:n</code>
148, 2936, 6933, 6936, 7034, 7039, 7048	3996
<code>\l_stex_get_svariable_str</code>	<code>\stex_has_definiens:NTF</code> . . .
6006	<i>139</i> , <u>3993</u>
<code>\stex_get_symbol:n</code>	<code>\stex_has_definiens:nTF</code>
.
140, 1981, <u>4050</u> , 4052,
4454, 5852, 5869, 5886, 6000, 6116,	<code>\stex_has_definiens_p:N</code> . . .
6442, 7086, 7264, 7313, 8490, 8513	<i>139</i> , <u>3993</u>
<code>\stex_get_symbol:nTF</code>	<code>\c_stex_home_file</code> . .
. . .	<i>117</i> , <u>238</u> , 825, 845
140, 1419, <u>4050</u> , 4053, 4058, 7035	<code>\stex_html_literal:n</code>
<code>\l_stex_get_symbol_args_tl</code>
.	27, 35, 749, 757, 1646, 1649, 2006
140, 3287, 3826,	<code>\stex_html_node:nnn</code>
3846, 3848, 3849, 4083, 4123, 4184,
4239, 4251, 4338, 4387, 4400, 4458,	<i>120</i> , <i>121</i> , <u>694</u> , 1441, 1540
5256, 6387, 6389, 6526, 6588, 7186	<code>\stex_if_check_terms:</code> 6029, 6040, 6044
<code>\l_stex_get_symbol_arity_int</code> . . .	<code>\stex_if_check_terms:TF</code>
.
140, 3286, 3769, 3825, 3847,	<i>144</i> , 3698, 4206, 4215, 4365,
3857, 3863, 3867, 3871, 3875, 3879,	4368, <u>6027</u> , 6049, 6060, 6118, 6157
3883, 3887, 3891, 3895, 3898, 3899,	<code>\stex_if_check_terms_p:</code> . . .
3900, 3901, 3938, 4050, 4082, 4122,	<i>144</i> , <u>6027</u>
4183, 4211, 4238, 4250, 4291, 4337,	<code>\stex_if_do_html:</code>
4386, 4399, 4435, 4457, 6182, 6216,	680
6247, 6254, 6274, 6323, 6428, 7185	<code>\stex_if_do_html:TF</code>
<code>\l_stex_get_symbol_def_tl</code>
.	<i>120</i> , <u>680</u> , 689, 1462, 1486, 1496,
140, 3163,	1644, 2275, 2285, 2542, 2557, 2855,
3288, 4084, 4124, 4185, 4339, 4459	2885, 2956, 3014, 3131, 3463, 3476,
<code>\l_stex_get_symbol_invoke_cs</code> . . .	3550, 3700, 3765, 3776, 3819, 4209,
.	4217, 4260, 4364, 5550, 6120, 6900,
140, 3291,	6942, 7056, 7189, 7278, 7304, 7429,
4087, 4127, 4188, 4342, 4462, 7038	7434, 7440, 7451, 7507, 7519, 7527,
<code>\l_stex_get_symbol_macro_str</code> .	7552, 7557, 7565, 7582, 7585, 7589,
3285	7600, 7620, 7691, 8321, 8612, 8644,
<code>\l_stex_get_symbol_name_str</code> . .	8651, 8652, 8655, 8675, 8712, 8740,
3156	8742, 8796, 8805, 8818, 8823, 8859,
<code>\l_stex_get_symbol_return_tl</code> . . .	8867, 8878, 8883, 8907, 8915, 8926,
.	8931, 8958, 8967, 8978, 8983, 9017,
140, 3290, 4086,	9072, 9097, 9099, 9146, 9237, 9262,
4126, 4187, 4213, 4341, 4356, 4461	9264, 9284, 9410, 9416, 9725, 9795
<code>\l_stex_get_symbol_type_tl</code>	<code>\stex_if_do_html_p:</code>
.	<i>120</i> , <u>680</u>
140, 3289, 4085, 4125, 4186,	<code>\stex_if_file_absolute:N</code> . . .
4340, 4460, 6892, 6927, 7039, 7050	263, 272
	<code>\stex_if_file_absolute:NTF</code>

	<i>117</i> , <u>262</u> , 853

<code>\stex_if_file_absolute_p:N</code>	<code>\l_stex_in_this_bool</code>
<code>\stex_if_file_starts_with:NN</code> 117, 279 4947, 5433, 5443,
<code>\stex_if_file_starts_with:NNTF</code>	5449, 5459, 5483, 5497, 5501, 5515,
. 117, 279, 1031	5551, 5565, 5782, 7014, 7025, 7027
<code>\stex_if_html_backend</code> 518	<code>\l_stex_inparray_bool</code> 6804
<code>\stex_if_html_backend:TF</code>	<code>\stex_input_with_hooks:Nn</code> 117,
. 120, 171, 2, 529,	245, 245, 248, 2012, 2023, 2031, 2034
670, 673, 697, 710, 724, 1439, 1535,	<code>\stex_invoke_sequence:</code>
1599, 1612, 1654, 2000, 2040, 2253, 141, 4391, 4404, 4550, 4550, 5624
4515, 4546, 5132, 5431, 5447, 5461,	<code>\stex_invoke_structure:</code>
5481, 5499, 5518, 5548, 5821, 5977, 142, 4639, 4639, 6877, 7038
5998, 6028, 7103, 7116, 7890, 7930,	<code>\stex_invoke_symbol:</code>
8060, 8140, 8195, 8207, 8215, 8227,	141, 3830, 4243, 4255, 4541, 4541, 7187
8237, 8378, 9136, 9279, 9382, 9419	<code>_stex_invoke_symbol:NnnnnnN</code>
<code>\stex_if_html_backend_p:</code> 120, 670 141, 4455, 4485, 4503, 4506
<code>\stex_if_in_module:</code> 2481	<code>_stex_invoke_symbol:nnnnnnN</code>
<code>\stex_if_in_module:TF</code> 141, 3988, 4066, 4079, 4080,
135, 1249, 2288, 2475, 2481, 6426, 7283	4485, 4485, 4504, 5049, 5063, 6991
<code>\stex_if_in_module_p:</code> 135, 2481	<code>\stex_invoke_text_symbol:</code>
<code>\stex_if_module_exists:N</code> 2485 141, 3761, 4545, 4545
<code>\stex_if_module_exists:n</code> 2489	<code>_stex_invoke_variable:nnnnnn</code>
<code>\stex_if_module_exists:NNTF</code> 142, 5127
. 135, 1307, 2304,	<code>_stex_invoke_variable:nnnnnnN</code>
2337, 2364, 2485, 2749, 2758, 2847 4248, 4397, 5127, 5622
<code>\stex_if_module_exists:nTF</code>	<code>_stex_is_sequentialized:n</code>
. 135, 2485, 6893, 6928, 7051 5577, 5660, 5733, 5743
<code>\stex_if_module_exists_p:N</code> 135, 2485	<code>\stex_iterate_break:</code>
<code>\stex_if_module_exists_p:n</code> 135, 2485 140, 4007, 4008, 4011
<code>\stex_if_smsmode:</code> 2603	<code>\stex_iterate_break:n</code>
<code>\stex_if_smsmode:TF</code> 140, 3226, 3590,
. 136, 640, 1453, 1764,	3600, 3618, 4007, 4008, 4014, 4181
1783, 2276, 2284, 2601, 2813, 2859,	<code>\stex_iterate_morphisms:nn</code>
2889, 3143, 3332, 3347, 3355, 3380, 137, 3209, 3209, 3568, 3584
3398, 3414, 3422, 3446, 3672, 3703,	<code>\stex_iterate_notations:nn</code>
3785, 6129, 7096, 7115, 7131, 7137, 146, 3062, 5100, 6459, 6459
7211, 7230, 7244, 7282, 8740, 8742	<code>\stex_iterate_symbols:n</code>
<code>\stex_if_smsmode_p:</code> 136, 2601 140, 4007, 4007, 4174
<code>\stex_ignore_spaces_and_pars:</code>	<code>\stex_iterate_symbols:nn</code>
. 115, 45, 45, 140, 2979, 3621, 4024, 4024, 5025
48, 2532, 2534, 7137, 8668, 9576, 9613	<code>\l_stex_key_*</code> 139
<code>\l_stex_import_uri</code> 2518, 2519,	<code>\l_stex_key_answerclass_str</code>
2521, 2852, 2853, 2857, 2861, 2862, 8781, 8785, 8799, 8833, 8834
2878, 2879, 2881, 2884, 2887, 2892,	<code>\l_stex_key_archive_str</code>
2893, 2900, 2903, 2939, 2941, 2943 379, 382, 1826, 1889
<code>\stex_in_archive:nn</code>	<code>\l_stex_key_args_str</code>
. 124, 883, 883, 1826, 3632, 3637, 3646, 3686,
1889, 1970, 1994, 2028, 2051, 2079,	3717, 3850, 3859, 3864, 3868, 3872,
2108, 2152, 2168, 2183, 2195, 2780	3876, 3880, 3884, 3888, 3892, 3896,
<code>\g_stex_in_comp_bool</code>	3911, 4263, 4351, 4352, 4411, 7068
. 5544, 5815, 5820, 5831	<code>\l_stex_key_argtypes_clist</code>
<code>\stex_in_invisible_html_bool</code>	3651, 3655, 3944, 3946, 4296, 4299,
. 3928, 4281, 5430, 5463	4440, 4443, 6075, 6076, 6077, 7072
<code>\l_stex_in_meta_bool</code>	<code>\l_stex_key_assoc_str</code> 3635, 3640,
. 2315, 2324, 2327, 2329	3918, 3919, 4267, 4268, 4415, 4416

\l_stex_key_autogradable_bool ...	3743, 3744, 3746, 3750, 3756, 3788,
..... 8555, 8561, 8566, 8620, 8720	3809, 3810, 3813, 3824, 3836, 3910,
\l_stex_key_continues_tl . 7338, 7347	4201, 4202, 4212, 4219, 4222, 4235,
\l_stex_key_def_tl 3648, 3658, 3685,	4237, 4249, 4262, 4348, 4349, 4371,
3716, 3731, 3735, 3758, 3827, 3932,	4383, 4385, 4398, 4410, 6155, 6412,
3933, 4240, 4252, 4285, 4286, 4388,	6413, 6414, 6418, 6419, 6420, 7065,
4401, 4429, 4430, 6067, 6068, 7070	7074, 7101, 7163, 7164, 7170, 7178,
\stex_key_def_tl 144	7184, 7190, 7648, 7655, 7662, 7663,
\l_stex_key_deprecate_str	7699, 7700, 8559, 8565, 8613, 8615,
..... 405, 407, 635, 636	8618, 8708, 8710, 8713, 8715, 8718
\l_stex_key_due_tl	\l_stex_key_number_tl
..... 9771, 9775, 9815, 9816 9769, 9773, 9789, 9790
\l_stex_key_fallback_tl	\l_stex_key_op_tl
..... 1755, 1808, 1840, 1906	... 3770, 4358, 4360, 6088, 6094,
\l_stex_key_feedback_tl	6175, 6176, 6177, 6184, 6185, 6190,
..... 9003, 9006, 9029,	6196, 6325, 6417, 6418, 6421, 6430,
9032, 9045, 9046, 9121, 9126, 9158,	6434, 6436, 6446, 6448, 6537, 6541
9160, 9194, 9195, 9296, 9298, 9328,	\l_stex_key_post_tl
9329, 9716, 9733, 9736, 9743, 9744	... 1754, 1814, 1819, 5836, 5840,
\l_stex_key_file_str	5875, 5892, 5914, 5928, 5955, 5965
380, 383, 1795, 1827, 1834, 1890, 1971	\l_stex_key_pre_tl
\l_stex_key_for_clist 1753, 1814, 1819, 5835, 5839,
..... 149, 3114, 7067,	5875, 5892, 5914, 5928, 5955, 5965
7075, 7085, 7335, 7344, 7647, 7652	\l_stex_key_prec_str 3772,
\l_stex_key_for_str	6087, 6093, 6224, 6227, 6230, 6236,
\l_stex_key_from_tl 7336, 7345	6239, 6242, 6245, 6251, 6263, 6264
\l_stex_key_Ftext_tl	\l_stex_key_proofend_tl
.. 9124, 9130, 9155, 9192, 9293, 9326 7337, 7346, 7759, 7760
\l_stex_key_functions_tl . 7340, 7348	\l_stex_key_pts_str
\l_stex_key_given_tl 9002, 9005, 9024, 9025, 9042,
..... 9770, 9774, 9812, 9813	9043, 9717, 9728, 9729, 9740, 9741
\l_stex_key_hide_bool 7342,	\l_stex_key_pts_tl
7351, 7455, 7466, 7519, 7592, 7620	8557, 8563, 8621, 8630, 8634, 8721,
\l_stex_key_id_str	8732, 8735, 8751, 9632, 9652, 9675
..... 387, 389, 641, 642, 7149,	\l_stex_key_reorder_str 3634, 3638,
7150, 7205, 7207, 7566, 7575, 7667,	3921, 3922, 4273, 4274, 4421, 4422
8790, 8792, 8798, 8853, 8855, 8861,	\l_stex_key_return_tl
8901, 8903, 8909, 8951, 8953, 8960,	3649, 3654, 3829, 3935, 3938, 4213,
9011, 9013, 9023, 9041, 9603, 9670,	4242, 4254, 4288, 4291, 4356, 4390,
9690, 9702, 9704, 9715, 9724, 9727	4403, 4432, 4435, 6071, 6072, 7071
\l_stex_key_intent_args_clist ...	\stex_key_return_tl 144
..... 6090, 6096, 6288, 6297, 6299	\l_stex_key_role_str
\l_stex_key_intent_str 3633, 3641, 3748, 3924,
..... 3771, 6089, 6095, 6213, 6291	3925, 4270, 4271, 4418, 4419, 7181
\l_stex_key_macroname_str	\l_stex_key_root_tl
.. 7066, 7076, 7162, 7164, 7180, 7184 5837, 5841, 5845, 5847
\l_stex_key_method_tl 7341,	\l_stex_key_short_tl
7350, 7475, 7481, 7482, 7649, 7653	.. 1505, 1507, 1550, 1553, 1570, 1572
\l_stex_key_mhrepos_str	\l_stex_key_sig_str .. 2233, 2248,
.. 8560, 8568, 8764, 8766, 8774, 9828	2258, 2292, 2333, 2351, 2353, 2355
\l_stex_key_min_tl 8558,	\l_stex_key_style_clist
8564, 8635, 8738, 8756, 9637, 9657	399, 401, 504, 508, 542, 546, 7152,
\l_stex_key_name_str	7153, 7247, 9077, 9078, 9082, 9085,
3645, 3653, 3683, 3714, 3729, 3733,	9210, 9217, 9241, 9242, 9247, 9250

<code>\l_stex_key_T_bool</code>	8852, 8900, 8950, 9010, 9063, 9144,
... 9122, 9127, 9128, 9148, 9152,	9186, 9228, 9282, 9320, 9421, 9438,
9175, 9189, 9286, 9290, 9309, 9323	9701, 9783, 9787, 9827, 9885, 9947
<code>\l_stex_key_term_tl</code>	<code>\stex_kpsewhich:Nn</code> 115, 77, 77, 89, 99
7349, 7475, 7478, 7479, 7650, 7654	<code>\c_stex_language_abbrevs_prop</code> ...
<code>\l_stex_key_testspace_dim</code> 119, 426
..... 8779, 8782, 8784,	<code>\c_stex_languages_clist</code> . 31, 454, 457
8813, 9367, 9369, 9425, 9454, 9457	<code>\c_stex_languages_prop</code>
<code>\l_stex_key_title_str</code> 119, 222, 426, 465, 1170
<code>\l_stex_key_title_tl</code>	<code>\g_stex_last_feature_str</code> 2555
. 393, 395, 1877, 1882, 1919, 1920,	<code>\l_stex_macroname_str</code>
2269, 7100, 7106, 7107, 8623, 8624,	139, 2545, 2546, 3662, 3667, 3669,
8625, 8629, 8723, 8724, 8725, 8727,	3693, 3742, 3755, 3823, 3912, 3913,
8728, 8832, 8833, 8892, 8893, 8940,	4200, 4236, 4247, 4264, 4265, 4347,
8941, 8992, 8993, 9603, 9670, 9685,	4384, 4396, 4412, 4413, 6878, 7180
9686, 9690, 9796, 9797, 9807, 9808	<code>\c_stex_main_archive</code> 125, 1029, 1966
<code>\l_stex_key_Ttext_tl</code>	<code>\c_stex_main_document_uri</code>
.. 9123, 9129, 9153, 9190, 9291, 9324 127, 1178, 1829
<code>\l_stex_key_type_tl</code>	<code>\c_stex_main_file</code>
3647, 3657, 3684, 3715, 3730, 3734, 116, 228, 244, 1181, 1188
3929, 3930, 4241, 4253, 4282, 4283,	<code>_stex_map_args:N</code>
4426, 4427, 6063, 6064, 7069, 7078 145, 3939, 4292, 4436, 5258,
<code>\stex_key_type_tl</code>	6284, 6310, 6386, 6386, 6533, 6590
144	<code>_stex_map_notation_args:N</code>
<code>\l_stex_key_variant_str</code> 145, 6386, 6399, 6553
..... 3719, 4224, 4373,	<code>\c_stex_mathhub_<id>_archive</code> .. 124
6086, 6092, 6099, 6101, 6143, 6165,	<code>\c_stex_mathhub_file</code>
6211, 6322, 6412, 6418, 6508, 6541	<code>\c_stex_mathhub_files</code>
<code>\l_stex_key_wikidata_str</code> 124, 822, 915, 921, 927, 1029
..... 3650, 3656, 3915, 3916	<code>\c_stex_mathhub_main_archive</code> ...
<code>\l_stex_keys_cls_id</code> 6, 13, 36, 43, 44, 1039, 1040
46, 81, 728, 735, 758, 765, 766, 768, 803	<code>_stex_maybe_brackets:nn</code>
<code>\l_stex_keys_counter_id</code> 147, 4559,
..... 7, 10, 28, 37, 61, 62, 63, 64,	5218, 5229, 6630, 6646, 6797, 6797
729, 732, 750, 759, 783, 784, 785, 786	<code>\c_stex_metadata_bool</code> 37, 8500, 8523
<code>\l_stex_keys_counter_parent</code>	<code>\stex_metagroup_do_in:n</code>
..... 8, 11, 29, 48, 50, 51, 52, 53, 117, 118, 327,
54, 55, 56, 58, 730, 733, 751, 770,	327, 331, 348, 2472, 3132, 3483, 3557
772, 773, 774, 775, 776, 777, 778, 780	<code>\stex_metagroup_new:</code>
<code>\stex_keys_define:nnnn</code>	118, 323, 324, 2301, 2334, 2360, 2966
..... 118, 5, 359, 359,	<code>\l_stex_metatheory_uri</code> 2019, 2230,
378, 386, 392, 398, 404, 410, 727,	2237, 2239, 2259, 2260, 2295, 2296,
1504, 1752, 1762, 2232, 3631, 3644,	2319, 2521, 2523, 2616, 7847, 7848
3728, 5834, 5844, 5936, 5941, 6085,	<code>_stex_module_code_macro:N</code>
6106, 6108, 6840, 7064, 7334, 7646,	2223, 2307, 2367, 2379, 2398, 2432,
8087, 8556, 8763, 8780, 8849, 9001,	2476, 2486, 2503, 2508, 2509, 2533
9120, 9365, 9768, 9872, 9923, 9936	<code>_stex_module_code_macro:n</code>
<code>\stex_keys_set:nn</code> 2226, 2490, 6975
..... 118, 17, 374, 374, 739,	<code>\stex_module_setup:n</code>
1511, 1566, 1793, 1892, 2268, 3665, 134, 2273, 2287, 2287, 2551
3692, 3740, 3741, 4199, 4346, 5851,	<code>_stex_module_setup_top_nosig:n</code> .
5868, 5885, 5899, 5911, 5925, 5947, 2293, 2300, 7784
5953, 5963, 6115, 6153, 6865, 7093,	<code>\stex_module_setup_top_nosig:n</code> ..
7128, 7427, 7550, 7628, 7661, 8181, 134, 2300
8604, 8608, 8698, 8773, 8789, 8812,	

<code>\stex_module_uri:n</code>	127	<code>_stex_next_symbol:n</code>	
<code>\stex_module_uri_archive:N</code>		141, 4464, 4466, 4484, 4858, 4988, 5973	
.....	127, 1208, 1208, 2773	<code>\c_stex_no_archive</code>	125, 1020
<code>\stex_module_uri_as_qm:n</code>		<code>\c_stex_no_archive_str</code>	
.....	128, 1242, 1242, 4137, 4138	125, 1020, 1146, 1334, 2776
<code>\stex_module_uri_name:N</code>		<code>\c_stex_no_archive_uri</code> ...	1022, 1028
.....	128, 1217, 1217, 2256, 2775, 2862, 2893	<code>\c_stex_no_frontmatter_bool</code>	39, 1700
<code>\stex_module_uri_name:n</code>		<code>_stex_notation_add:</code>	
.....	128, 1217, 1221, 4145, 4148	...	145, 3699, 3775, 6119, 6319, 6319
<code>\stex_module_uri_path:N</code>		<code>\l_stex_notation_args_tl</code>	
.....	127, 1211, 1211, 2774	6181, 6214,
<code>\stex_module_uri_path:n</code>		6283, 6289, 6295, 6400, 6402, 6499	
.....	127, 1211, 1214, 4157	<code>_stex_notation_check:</code> ..	146, 3698,
<code>\stex_module_uri_split_name:NNN</code> ..		4215, 4368, 6118, 6157, 6496, 6496	
.....	128, 1224, 1224, 6933	<code>\l_stex_notation_cs</code>	
<code>\stex_module_uri_split_name:NNn</code>	146, 4558, 4561, 4572,
.....	128, 1224, 1228	4599, 4603, 4611, 4618, 4633, 4794,	
<code>\l_stex_morphism_assigns_seq</code> 2973,		5195, 5199, 5220, 5686, 6486, 6492	
3011, 3028, 3084, 3133, 3184, 3484		<code>_stex_notation_do_html:n</code>	
<code>\l_stex_morphism_morphisms_seq</code>	146, 3701, 3778, 4219, 6122, 6505, 6505
.....	2972, 3001, 3012	<code>\l_stex_notation_downprec</code>	
<code>\l_stex_morphism_renames_seq</code> 2971,		5538, 6789, 6796, 6802, 6803
3010, 3025, 3039, 3052, 3257, 3558		<code>\l_stex_notation_macrocode_cs</code> ...	
<code>\l_stex_morphism_symbols_prop</code>	145, 3721, 4226, 4375, 6147,
.....	3158, 3624	6167, 6209, 6221, 6324, 6412, 6415,	
<code>\l_stex_morphism_symbols_seq</code> ...		6429, 6433, 6444, 6499, 6500, 6531	
.....	2970, 2980, 3009,	<code>_stex_notation_make_args:</code>	
3022, 3031, 3173, 3253, 3314, 3322		146, 3722,
<code>_stex_morphisms_macro:N</code> ..	2211,	4227, 6148, 6168, 6501, 6552, 6552	
2308, 2368, 2380, 2389, 2426, 2913		<code>\stex_notation_parse:n</code>	
<code>_stex_morphisms_macro:n</code>	145, 3697, 3774, 4214,
..	2214, 2990, 3236, 4036, 6474, 6978	4359, 4362, 6117, 6156, 6174, 6174	
<code>\stex_new_document_element_uri:n</code>		<code>_stex_notation_set_default:n</code> ...	
.....	127, 1127, 1127, 1765	...	145, 6125, 6160, 6425, 6425, 6450
<code>\stex_new_module_uri:n</code>		<code>_stex_notations_macro:N</code>	
.....	128, 1248, 1248,	...	2217, 2310, 2370, 2382, 2400,
2302, 2335, 2362, 6872, 6936, 6961		2433, 2459, 6340, 6350, 6351, 6359	
<code>\stex_new_statement:nnn</code>		<code>_stex_notations_macro:n</code> ..	2220, 6471
.....	149, 7091, 7091, 7223, 7228, 7243, 7246	<code>\stex_par:</code>	
<code>\stex_new_stylable_cmd:nnnn</code>	9211, 9212, 9215, 9218, 9219, 9222
..	119, 482, 482, 2851, 2868, 3376,	<code>\stex_persist:n</code>	119,
3442, 3461, 3540, 3566, 3664, 3691,		176, 586, 601, 602, 711, 712, 714,	
3739, 4198, 4345, 6114, 6152, 7626		717, 720, 721, 1004, 1047, 1054, 2386	
<code>\stex_new_stylable_env:nnnnnnn</code> ..		<code>\c_stex_persist_force_bool</code> ..	35, 581
..	119, 516, 516, 2267, 3329, 3338,	<code>\c_stex_persist_mode_bool</code>	
3394, 3404, 6853, 6888, 6922, 7092,		33, 571, 582, 587
7501, 7523, 7549, 8598, 8697, 8804,		<code>_stex_persist_read_now:</code>	
8866, 8914, 8966, 9062, 9227, 9780		586, 1053, 7855
<code>\stex_new_symbol_uri:n</code>		<code>\c_stex_persist_write_mode_bool</code> ..	
.....	128, 1378, 1378, 3065, 3750, 3813, 7021	...	34, 577, 588, 594, 709, 1052, 2376
<code>\stex_new_symbol_uri:nn</code>		<code>\stex_pseudogroup:nn</code> 117, 118, 250,	
.....	128, 1381, 1381, 2981,	303, 303, 685, 884, 2501, 5247, 6827	
3989, 4121, 4182, 5032, 6991, 7190		<code>\stex_pseudogroup_restore:N</code>	
		118, 258,

259, 306, 306, 897, 2511, 6832, 6833
 \stex_pseudogroup_with:nn
 118, 313, 313, 4008, 4026,
 4079, 5818, 6461, 6718, 6737, 6762
 \c_stex_pwd_file
 116, 228, 855, 1031, 1032, 1751
 \stex_reactivate_macro:N
 115, 67, 73, 2540, 2874,
 2909, 3202, 3203, 3204, 3366, 3372,
 3391, 3432, 3438, 3458, 3677, 3708,
 3797, 6139, 6863, 6919, 6970, 7216,
 7217, 7218, 7219, 7220, 7221, 7235,
 7236, 7237, 7238, 7239, 7240, 7241,
 7442, 7443, 7444, 7445, 7446, 7447,
 7448, 7449, 8582, 8583, 8584, 8585,
 8586, 8587, 8588, 8976, 9096, 9261
 \stex_ref_new_doc_target:n
 642, 1763, 1763, 1781
 \stex_ref_new_id:n 7206
 \stex_ref_new_sym_target:n
 1782, 1782, 6012, 7213, 7232
 \stex_ref_new_symbol:n
 1977, 2429, 3763, 3832
 \l_stex_relative_import_bool ...
 1269, 1271, 1309, 2901
 \stex_renamedec1_do:nn
 3538, 3542, 3548
 \stex_require_module:N
 137, 2523, 2748, 2748, 2853, 2879, 2941
 \stex_require_module_noerr:N ...
 137, 2748, 2757, 2768
 \stex_require_module_noerr:n ...
 137, 2748, 2766, 2903
 \stex_return_args:nn
 3841, 3939, 4292, 4436
 \l_stex_return_notation_tl
 4524, 4694, 4702, 4866,
 4867, 4955, 4981, 4982, 5392, 5398
 \stex_set_language:n 2246
 \stex_sms_allow:N 136, 2562,
 2564, 2577, 2578, 2579, 2580, 2581
 \stex_sms_allow_env:n
 136, 2572, 2574, 2582, 3368, 3374,
 3434, 3440, 6861, 6916, 6966, 7120
 \stex_sms_allow_escape:N
 136, 2567, 2569, 2583, 2584,
 2875, 3393, 3460, 3470, 3545, 3614,
 3678, 3709, 3798, 6140, 7140, 7276
 \stex_sms_allow_import:Nn
 136, 2585, 2588, 2908
 \stex_sms_allow_import_* 136
 \stex_sms_allow_import_env:nn ...
 136, 2592, 2594, 2599
 \g_stex_sms_import_code
 136, 2600, 2612, 2629, 2902
 \stex_smsmode_do: 136, 2281,
 2525, 2536, 2654, 2656, 2659, 2659,
 2870, 3360, 3386, 3427, 3453, 3468,
 3543, 3673, 3704, 3793, 6135, 6855,
 6910, 6954, 7113, 7137, 7138, 7259
 \stex_source_path:n
 125, 1067, 1067, 1076, 1834,
 1890, 1971, 2004, 2012, 2022, 2023,
 2031, 2034, 2159, 2175, 2192, 2782
 \stex_source_path:nn
 ... 125, 1067, 1074, 2154, 2170, 2185
 \stex_sref_do_aux:n 7770
 \stex_str_if_ends_with:nn ... 115, 62
 \stex_str_if_ends_with:nnTF
 115, 62, 3575, 3598, 4138, 4145, 4157
 \stex_str_if_ends_with_p:nn
 115, 62, 4110, 4179
 \stex_str_if_starts_with:nn 115, 57
 \stex_str_if_starts_with:nnTF ...
 .. 115, 57, 413, 998, 1862, 4672, 4676
 \stex_str_if_starts_with_p:nn ...
 115, 57
 \l_stex_struct_this_tl
 4658, 4860, 4861, 4864, 4942, 4945,
 4954, 4975, 4976, 4979, 4993, 5783
 \stex_structural_feature_-
 module:nn ... 135, 2541, 2541, 6880
 \stex_structural_feature_module_-
 end: 135, 2541, 2554, 6857, 6912, 6959
 \stex_structural_feature_-
 morphism:nnnnn 137,
 2919, 2926, 3330, 3378, 3395, 3444
 \stex_structural_feature_-
 morphism_check_total:
 3172, 3397, 3413, 3451
 \stex_structural_feature_-
 morphism_end: . 137, 2919, 3007,
 3335, 3350, 3385, 3401, 3417, 3452
 \stex_structural_feature_-
 morphism_with_macros:nnnnn ..
 137, 2919, 2922, 3342, 3377, 3408, 3443
 \stex_style_apply: 119, 3, 482, 487,
 524, 530, 725, 2279, 2284, 2864,
 2895, 3333, 3339, 3344, 3348, 3358,
 3383, 3399, 3405, 3410, 3415, 3425,
 3449, 3688, 3725, 3790, 4230, 4377,
 6132, 6171, 7110, 7116, 7440, 7510,
 7527, 7565, 7582, 7585, 7605, 7639,
 8644, 8651, 8740, 8742, 8810, 8822,
 8872, 8882, 8920, 8930, 8963, 8982,
 9097, 9101, 9262, 9266, 9800, 9803

<code>\stex_suppress_html:n</code>	<code>\stex_titlefragment:</code>
..... 120, <u>684</u> , 684, 2642, 6204	8002
<code>\stex_symbol_macro:N</code>	<code>\stex_undefine:N</code>
... 2205, 2309, 2369, 2381, 2393,	. 115, <u>53</u> , 53, 56, 310, 320, 2021, 2618
2427, 2428, 3962, 3971, 3972, 6956	<code>\stex_uri_from_archive_file:Nn</code> . 127
<code>\stex_symbol_macro:n</code>	<code>\stex_uri_from_pair:Nn</code>
... 2208, 3998, 4018, 4039, 4139, 128, <u>1349</u> , 1349, 2239
4152, 4158, 6983, 6989, 7292, 7294	<code>\stex_uri_from_pair:Nnn</code>
<code>\stex_symbol_uri:n</code> 128, <u>1269</u> , 1270, 1287, 1358,
128	1362, 2518, 2852, 2878, 2900, 2939
<code>\stex_symbol_uri_archive:N</code>	<code>\stex_uri_resolve:Nn</code>
..... 128, <u>1384</u> , 1384	2243
<code>\stex_symbol_uri_module:N</code>	<code>\stex_uri_use:N</code>
..... 129, <u>1390</u> , 1390	7150
<code>\stex_symbol_uri_module:n</code>	<code>\stex_use_archive_uri:n</code>
129, <u>1390</u> , 1393, 3998, 7284, 7292, 7294 126, <u>1086</u> , 1086
<code>\stex_symbol_uri_name:N</code>	<code>\stex_use_document_uri:N</code>
129, <u>1400</u> , 1400, 3561, 4491, 5872, 126, <u>1089</u> , 1089, 1194,
5889, 5981, 6004, 6144, 8502, 8525	1272, 1409, 1766, 1767, 1769, 1774,
<code>\stex_symbol_uri_name:n</code>	1828, 1833, 2008, 2022, 2353, 2620
129, <u>1400</u> , 1403, 3056, 4000, 7293, 7294	<code>\stex_use_document_uri:n</code>
<code>\stex_symbol_uri_path:N</code> 126, <u>1089</u> , 1092
..... 128, <u>1387</u> , 1387	<code>\stex_use_module_uri:N</code>
<code>\stex_symdecl_do:</code> 127, <u>1195</u> , 1195, 1310,
..... 139, <u>247</u> , <u>320</u> , 3752,	1323, 1338, 1347, 2260, 2277, 2296,
3816, <u>3835</u> , 3835, 4205, 4354, 7182	2303, 2336, 2361, 2363, 2378, 2425,
<code>\stex_symdecl_html:</code>	2477, 2519, 2751, 2848, 2857, 2861,
... 139, 3766, 3819, <u>3907</u> , 3907, 7189	2887, 2892, 2951, 2959, 3357, 3382,
<code>\stex_term_arg:nnn</code>	3424, 3448, 3955, 3984, 6338, 7048
... 143, 5364, 5540, <u>5543</u> , 5549, 5565	<code>\stex_use_module_uri:n</code>
<code>\stex_term_arg:nnnnn</code>	127,
..... 143, <u>5532</u> , 5532, 5584,	<u>1195</u> , 1198, 2499, 2504, 2513, 2914,
5643, 5661, 5744, 6567, 6690, 6698	4766, 6902, 6944, 6974, 6978, 7058
<code>\stex_term_arg_aB:nnnnn</code>	<code>\stex_use_notation:nn</code>
... 143, <u>5567</u> , 5581, 6703, 6711, 6728	146
<code>\stex_term_do_aB_clist:</code>	<code>\stex_use_notation:nnTF</code>
..... 284, 5575,	146, 4570, 4793, 5191, 5685, <u>6484</u> , 6484
5591, 5645, 5674, 5703, 5751, 5754,	<code>\stex_use_op_notation:nnTF</code>
6718, 6719, 6738, 6742, 6763, 6768 146, 4557, 5217, <u>6484</u> , 6490
<code>\stex_term_oma:nn</code>	<code>\c_stex_use_sref_bool</code>
... 142, 5257, <u>5481</u> , 5482, 5497, 6642	27
<code>\stex_term_oma_or_omb:nn</code>	<code>\stex_use_symbol_uri:N</code>
..... 6642, 6647, 6695, 6708	128, <u>1364</u> , 1364, 1982, 3130, 3136,
<code>\stex_term_omb:nn</code>	3474, 3477, 3549, 3552, 3682, 3695,
142, 5285,	3701, 3713, 3764, 3778, 3781, 3787,
5286, <u>5499</u> , 5500, 5515, 6695, 6708	3833, 4489, 5870, 5887, 5980, 6002,
<code>\stex_term_oms:nn</code>	6122, 6145, 6432, 6435, 6443, 6444,
.... 142, 4517, 4519, <u>5431</u> , 5432,	6445, 6446, 6455, 7191, 7198, 7270,
5443, 5446, 5853, 5876, 5893, 6541	7319, 7463, 7635, 7696, 8506, 8529
<code>\stex_term_oms_or_omv:nn</code>	<code>\stex_use_symbol_uri:n</code>
..... 4517, 4519, 4547, 128, <u>1364</u> , 1367, 2460, 3185,
4560, 4712, 4750, 4789, 4845, 5134,	4496, 5103, 5104, 5106, 5107, 5114,
5136, 5219, 5230, 5240, 5446, 6631	5115, 5117, 5118, 6341, 6358, 6361,
<code>\stex_term_omv:nn</code>	6375, 6376, 6377, 6380, 6381, 6382,
..... 142, 5134, 5136, 5154,	7020, 7159, 7213, 7232, 7279, 7291
<u>5447</u> , 5448, 5459, 5905, 5919, 5933	<code>\stex_var_notation_macro:</code>
	... 145, 4216, 4369, 6158, <u>6411</u> , 6411
	<code>\stex_variable:nnnnnnnN</code> . 4196, 4335
	<code>\l_stex_variables_prop</code>
 140, <u>4194</u> , 4235, 4318, 4383

<code>\stex_with_file_hooks:Nnn</code>	<code>__stex_expr_arg:n</code>
..... 117, 246, <u>249</u> , 249, 2621	5254, 5289
stex internal commands:	<code>\l__stex_expr_arg_counter_int</code> ...
<code>\l__stex_annotate_do_output_bool</code> 5243, 5251, 5266, 5301, 5302
... 670, 675, 678, 681, 686, 690, 7772	<code>__stex_expr_arg_do:nnn</code>
<code>__stex_annotate_env_str</code> ... 652, 653 5303, 5309, 5325, 5360, 5367
<code>__stex_check_env_str</code> 6033, 6034, 6035	<code>__stex_expr_arg_inner:nn</code>
<code>__stex_comps_do_defref:nn</code> 5292, 5295, 5299, 5305
..... 5948, 5954, 5964, 5997	<code>\l__stex_expr_assigned_seq</code>
<code>__stex_comps_do_ref:nNn</code> .. 5853, 4854, 4973, 5015, 5082
5874, 5891, 5901, 5913, 5927, 5976	<code>__stex_expr_assoc_make_seq:nnn</code> .
<code>__stex_comps_slash:w</code> 5657, 5684, 5763
..... 5984, 6008, 6020, 6024	<code>__stex_expr_assoc_seq:nnnnnnn</code> ..
<code>__stex_comps_split_slash:n</code> 5601, 5652
..... 5856, 5872, 5889	<code>__stex_expr_check:n</code>
<code>__stex_comps_split_slash_i:nw</code> ..	5333
..... 5857, 5860, 5862	<code>__stex_expr_check:nTF</code> ... 5302, 5308
<code>__stex_comps_uppercase:n</code>	<code>__stex_expr_check_b:nn</code> .. 5258, 5283
..... 5881, 5892, 5928, 5965	<code>__stex_expr_check_comp:</code>
<code>__stex_debug:nn</code> 5434, 5443, 5450, 5459, 5484,
105, 108, 113	5497, 5502, 5515, 5543, 5557, 5565
<code>\l__stex_debug_cl</code>	<code>\l__stex_expr_clist</code> .. 4578, 4585,
121, 123, 126	4592, 4596, 4828, 4849, 4855, 4857
<code>__stex_debug_env_str</code> . 119, 120, 123	<code>\l__stex_expr_comp_cs</code>
<code>__stex_doc_check_topsect:</code> 4881, 4886, 4905, 4907
..... 1655, 1661, 1667, 1673	<code>\l__stex_expr_count_int</code>
<code>__stex_doc_do_section:n</code> 5580, 5587, 5596, 5643, 5661, 5744
..... 1512, 1518, 1522, 1526, 1579	<code>\l__stex_expr_cs</code>
<code>\g__stex_doc_ftml_link_text_tl</code> 4830, 4836, 4843, 5686, 5689, 5709
..... 1414, 1417, 1429	<code>__stex_expr_current_type:</code>
<code>__stex_doc_maketitle:</code> ... 1475, 1480 4784, 4920, 4996
<code>__stex_doc_orig_backmatter</code>	<code>\l__stex_expr_current_type_tl</code> ...
..... 1713, 1716, 1734 4709, 4746, 4787, 4792
<code>__stex_doc_orig_frontmatter</code> ...	<code>__stex_expr_custom:n</code> 5171, 5181, 5245
..... 1703, 1706, 1723	<code>\l__stex_expr_customs_prop</code>
<code>__stex_doc_set_title:n</code> .. 1452, 1469 5249, 5261, 5262,
<code>__stex_doc_skip_section:</code>	5263, 5278, 5313, 5319, 5334, 5337
..... 1600, 1606, 1610	<code>\l__stex_expr_customs_seq</code>
<code>__stex_doc_skip_section_i:</code> 5250, 5267, 5268, 5269, 5279, 5335
..... 1585, 1588, 1591, 1601, 1606	<code>__stex_expr_do_aB_clist:</code>
<code>__stex_doc_title_html:</code> 5567, 5575, 5645, 5703
..... 1457, 1461, 1472	<code>__stex_expr_do_ab_next:nn</code>
<code>\g__stex_doc_title_tl</code> 5257, 5259, 5285, 5286
..... 1434, 1449, 1456, 1463, 1468	<code>__stex_expr_do_all:w</code> ... 4627, 4636
<code>\l__stex_doc_titlefragment_bool</code> .	<code>__stex_expr_do_assign:nn</code> 4730, 4734
..... 1563, 1568, 1578, 1583	<code>__stex_expr_do_assign_list:n</code> ...
<code>__stex_doc_x_matter:</code> ... 1701, 1745 4705, 4727
<code>__stex_expr_aB_arg:nnnnn</code> 5589, 5595	<code>__stex_expr_do_decl:nnnnnnnnn</code> ..
<code>__stex_expr_aB_simple_arg:nnnnn</code> 5030, 5058
..... 5610, 5641	<code>__stex_expr_do_decl_nomacro:nnnnnnnnn</code>
<code>__stex_expr_add_prop_arg:nnw</code> 5028, 5044
..... 5253, 5277, 5280	<code>__stex_expr_do_first_arg:n</code>
<code>\l__stex_expr_after_tl</code> 4614, 4621
.. 5034, 5038, 5039, 5040, 5111, 5122	<code>__stex_expr_do_first_next:</code>
<code>__stex_expr_annotate:nnn</code> . 5436, 4616, 4623
5452, 5468, 5485, 5503, 5519, 5529	


```

\__stex_expr_do_headterm:nn ....
..... 5438, 5454, 5462, 5479, 5753
\l__stex_expr_do_one:w ... 4625, 4631
\__stex_expr_do_seqmap:nnnnnn ...
..... 5607, 5716
\__stex_expr_end: .....
.. 4673, 4677, 4693, 4698, 5603, 5616
\l__stex_expr_field_name_str ...
..... 4960, 4964, 4965, 4999
\l__stex_expr_fields_clist 4706,
4728, 4735, 4753, 4756, 5073, 5074
\l__stex_expr_first_args_tl ....
..... 4613, 4633, 4637
\__stex_expr_full_notation:n ...
..... 5187, 5190
\__stex_expr_get_field_name:n ...
..... 4959, 4971
\__stex_expr_get_index_notation:n
..... 4564, 4569, 4632
\__stex_expr_gobble:nnnnnnnn ...
..... 5603, 5616
\l__stex_expr_iarg_tl 5696, 5698, 5709
\__stex_expr_invoke:nnnnN 4493, 4514
\__stex_expr_invoke_field:n ....
..... 4935, 4969
\__stex_expr_invoke_maybe_
field:nn ..... 4924, 4928
\__stex_expr_invoke_this:n 4666, 4911
\__stex_expr_is_argsep:n ..... 5635
\__stex_expr_is_seqmap:n ..... 5629
\__stex_expr_is_seqmap:nTF ... 5605
\__stex_expr_is_varseq:n ..... 5619
\__stex_expr_is_varseq:nTF 5598, 5721
\__stex_expr_make_mod:n .. 4752, 4764
\__stex_expr_make_oml:n .. 4756, 4771
\__stex_expr_make_oml:nn . 4772, 4774
\__stex_expr_make_prop: .....
..... 4796, 4929, 5012
\__stex_expr_make_prop_assign: ..
..... 4797, 4932, 5072
\__stex_expr_make_prop_assign:nn
..... 5075, 5080
\__stex_expr_make_prop_assign_
replace:nnnn ..... 5083, 5089
\__stex_expr_make_type:n .....
..... 4715, 4720, 4722, 4738
\__stex_expr_math: ..... 4543, 5160
\__stex_expr_maybe_notation:w ...
..... 4661, 4663, 4791
\__stex_expr_maybe_return:n ....
..... 5198, 5209, 5368
\__stex_expr_merge:nw ... 4655, 4671
\l__stex_expr_more_nextsymbol_tl
..... 4972, 4974, 5002

\__stex_expr_notation:w .....
..... 4637, 5173, 5174, 5183, 5226
\l__stex_expr_old_seq .....
..... 5728, 5731, 5735, 5740
\__stex_expr_op_custom:n .....
..... 5164, 5181, 5237
\__stex_expr_op_notation:w .....
..... 5166, 5167, 5185, 5215
\__stex_expr_present: ... 4802, 4933
\__stex_expr_present:nn .....
..... 4806, 4812, 4817, 4824, 4827
\__stex_expr_present_entry:nn ...
..... 4832, 4838, 4853
\__stex_expr_present_i:w .....
..... 4798, 4804, 4810
\__stex_expr_present_ii:nw 4815, 4823
\l__stex_expr_prop ..... 4842,
4962, 4970, 5005, 5013, 5035, 5045,
5047, 5059, 5061, 5081, 5084, 5090
\__stex_expr_prop_do_decls: ....
..... 5017, 5020
\__stex_expr_prop_do_notations: .
..... 5037, 5099
\__stex_expr_range:w .... 4565, 4576
\l__stex_expr_redo_tl .....
.. 4989, 5016, 5039, 5093, 5110, 5121
\l__stex_expr_reset_tl .....
..... 4464, 4468, 4472, 4473, 4479
\__stex_expr_ret_cs: .....
..... 5372, 5377, 5408, 5413, 5418
\__stex_expr_return: .... 5387, 5390
\__stex_expr_return_arg:n 5374, 5380
\l__stex_expr_return_args_tl ...
.. 5370, 5381, 5386, 5395, 5415, 5420
\__stex_expr_return_next: 5375, 5384
\l__stex_expr_return_this_tl ...
..... 5369, 5386,
5391, 5395, 5401, 5404, 5414, 5422
\l__stex_expr_seq .....
..... 5014, 5046, 5060, 5101
\__stex_expr_seq_arg:n ... 4579, 4590
\__stex_expr_seq_first: .. 4553, 4609
\__stex_expr_seq_op:w ... 4552, 4556
\l__stex_expr_set_comp_tl .....
..... 4640, 4695, 4699, 4859, 4984
\__stex_expr_set_custom_comp:n ..
..... 4700, 4875, 4893
\__stex_expr_set_custom_i:nn ...
..... 4896, 4901
\__stex_expr_set_customcomp: ...
..... 4673, 4698
\__stex_expr_set_this:n .. 4930, 4939
\__stex_expr_set_thiscomp: 4640, 4644

```

```

\__stex_expr_set_thisnotation: ..
    ..... 4677, 4693
\__stex_expr_setup:nnnnn .....
    ..... 4516, 4523, 5133
\__stex_expr_struct_top:n . 4641,
    4650, 4674, 4678, 4681, 4684, 4686
\__stex_expr_struct_type:n 4657, 4704
\__stex_expr_text: ..... 4543, 5179
\__stex_expr_varseq_in_map:nnnnnnnn
    ..... 5723, 5762
\__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 ....
    ..... 2773, 2776, 2780, 2840
\__stex_import_check_file:nnn ...
    ..... 2792,
    2793, 2794, 2800, 2801, 2802, 2829
\l__stex_import_doc_uri .. 2839, 2846
\l__stex_import_file_str .....
    ..... 2760, 2790,
    2812, 2814, 2817, 2822, 2833, 2846
\__stex_import_import_module:nn .
    ..... 2869, 2877, 2910
\__stex_import_import_module_-
    presms:nn ..... 2899, 2910
\l__stex_import_language_str ...
    ..... 2791, 2795, 2803, 2843
\__stex_import_load_check:n ....
    ..... 2778, 2784, 2789
\__stex_import_load_file: .....
    ..... 2818, 2823, 2838
\__stex_import_load_module:NTF ..
    ..... 2750, 2759, 2771
\l__stex_import_name_str .....
    .. 2775, 2792, 2793, 2794, 2798, 2842
\l__stex_import_path_str .....
    .. 2774, 2777, 2782, 2797, 2799, 2841
\l__stex_import_pre_str .....
    ..... 2777, 2781, 2830, 2831, 2833
\l__stex_import_uri .....
    ..... 2767, 2768, 2772, 2847, 2848
\l__stex_inputs_gin_repo_str ...
    ..... 2187, 2190, 2195
\l__stex_inputs_id_seq .....
    ..... 2124, 2125, 2130, 2138
\l__stex_inputs_libinput_files_-
    seq ..... 2053, 2054, 2057,
    2063, 2064, 2085, 2086, 2110, 2111,
    2114, 2122, 2134, 2135, 2145, 2146
\l__stex_inputs_path_seq .....
    .. 2123, 2126, 2128, 2131, 2139, 2142

\l__stex_inputs_path_str .....
    2131, 2132, 2133, 2134, 2135, 2138,
    2139, 2142, 2143, 2144, 2145, 2146
\__stex_inputs_ref:n 1996, 2003, 2016
\c__stex_inputs_ref_post_str ...
    ..... 2002, 2009
\c__stex_inputs_ref_pre_str ....
    ..... 2001, 2007
\l__stex_inputs_tmp_str .. 2064, 2066
\__stex_inputs_up_archive:nn ...
    .. 2052, 2062, 2084, 2109, 2118, 2118
\l__stex_inputs_uri .. 1995, 2008,
    2012, 2022, 2023, 2029, 2031, 2034
\__stex_inputs_usetikzlibrary:n .
    ..... 2077, 2079, 2083
\__stex_inputs_usetikzlibrary_-
    i:nn ..... 2086, 2092
\__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: ..... 975, 999
\l__stex_mathhub_base_str .....
    ..... 983, 993, 998, 999, 1001, 1005
\l__stex_mathhub_bool .....
    .... 941, 942, 944, 947, 956, 958, 967
\__stex_mathhub_check_manifest: .
    ..... 946, 954
\__stex_mathhub_check_manifest:n
    ..... 955, 957, 959, 964
\l__stex_mathhub_cs .....
    .... 885, 888, 890, 894, 898, 899, 904
\__stex_mathhub_do_manifest:nn ..
    ..... 919, 926, 1010
\l__stex_mathhub_file_str . 981, 1001
\__stex_mathhub_find_manifest:n 931
\__stex_mathhub_find_manifest:nn
    ..... 928, 937, 1034
\l__stex_mathhub_id_str .....
    ..... 982, 992, 1001, 1005
\l__stex_mathhub_key .. 988, 989, 991
\c__stex_mathhub_manifest_iior ...
    ..... 971, 980, 985, 997
\l__stex_mathhub_manifest_str ...
    ..... 929, 932, 938, 968, 980, 1035
\__stex_mathhub_parse_manifest:n
    ..... 933, 979, 1038
\__stex_mathhub_replace_https:w .
    ..... 975, 999
\__stex_mathhub_require:n .. 906, 913
\__stex_mathhub_restore:nn .....
    ..... 1005, 1009, 1048

```

<code>\l__stex_mathhub_seq</code> . . . 940, 943, 948, 965, 966, 968, 981, 987, 988, 990	<code>__stex_morphisms_add_symbol:nnnnnnnnN</code> 3059, 3075, 3077, 3082
<code>__stex_mathhub_set_current:n</code> 893, 905	<code>__stex_morphisms_add_symbol_-</code> <code>i:nnnnnnnnN</code> 3089, 3092
<code>\l__stex_mathhub_str</code> 823, 824, 828, 829, 834, 840, 843, 850, 852, 853, 854, 859	<code>\l__stex_morphisms_ass_tl</code> 3492, 3508, 3512, 3523, 3524, 3525
<code>\l__stex_mathhub_tl</code> 948	<code>__stex_morphisms_begin_copy:Nnnn</code> 3330, 3342, 3353
<code>\l__stex_mathhub_val</code> .. 990, 992, 993	<code>__stex_morphisms_begin_interpret:Nnnn</code> 3395, 3408, 3420
<code>__stex_modules_export:n</code> . 2529, 2531	<code>__stex_morphisms_break:</code> . 3150, 3154
<code>__stex_modules_html_annots:</code> 2254, 2275	<code>__stex_morphisms_check_name:nnnn</code> 3270, 3273
<code>__stex_modules_load_meta:</code> 2297, 2315, 2317	<code>\l__stex_morphisms_continue_bool</code> 3211, 3214, 3227, 3246
<code>__stex_modules_load_meta_i:n</code> 2319, 2323	<code>__stex_morphisms_cs:nnnn</code> 3212, 3237
<code>__stex_modules_load_sig:</code> 2342, 2347	<code>__stex_morphisms_definiens_-</code> <code>impl:nn</code> 3122, 3207
<code>__stex_modules_macro_short:nnn</code> 2202, 2206, 2209, 2212, 2215, 2218, 2221, 2224, 2227	<code>__stex_morphisms_do:n</code> 2975, 2988, 3003
<code>__stex_modules_nested:n</code> 2288, 2359, 2359	<code>__stex_morphisms_do_decls:</code> 2974, 2978
<code>__stex_modules_persist_module:</code> 2376, 2385, 7768	<code>__stex_morphisms_do_elaboration:</code> 3017, 3020
<code>__stex_modules_persist_not_-</code> <code>i:nn</code> 2401, 2406	<code>__stex_morphisms_do_for_list:</code> 3112, 3205
<code>__stex_modules_restore_module:nnnn</code> 2387, 2423	<code>__stex_morphisms_do_morph:nnnn</code> 2994, 2999
<code>__stex_modules_restore_not:n</code> 2437, 2442	<code>__stex_morphisms_do_morph_-</code> <code>assign:nnn</code> 3573, 3576, 3617
<code>__stex_modules_restore_not_i:n</code> 2443, 2446, 2465	<code>__stex_morphisms_do_parsed_-</code> <code>assign:</code> 3514, 3517
<code>__stex_modules_restore_not_-</code> <code>ii:nnnnn</code> 2450, 2454	<code>__stex_morphisms_do_parsed_-</code> <code>newname:</code> 3521, 3529
<code>\l__stex_modules_sig</code> 2349, 2353, 2354	<code>__stex_morphisms_do_parsed_-</code> <code>newname:w</code> 3531, 3533, 3537
<code>\l__stex_modules_sigfile</code> 2348, 2353, 2355	<code>\l__stex_morphisms_dones_seq</code> 2989, 2991, 2992
<code>__stex_modules_stringify_-</code> <code>uri:nnn</code> 2411, 2424	<code>__stex_morphisms_elab:nn</code> 3032, 3050
<code>__stex_modules_stringify_-</code> <code>uri:nnnn</code> 2416, 2461	<code>__stex_morphisms_elab_check_-</code> <code>assign:nnnnn</code> 3085, 3096
<code>\l__stex_modules_tl</code> . 2455, 2457, 2462	<code>__stex_morphisms_elab_check_-</code> <code>rename:nnn</code> 3053, 3104
<code>__stex_modules_top:n</code> . . . 2288, 2291	<code>__stex_morphisms_elab_i:nnn</code> 3056, 3071
<code>__stex_modules_top_sig:n</code> 2293, 2332, 2332	<code>__stex_morphisms_end:</code> . . . 3521, 3537
<code>\l__stex_modules_uri</code> . 2302, 2303, 2304, 2307, 2308, 2309, 2310, 2312, 2313, 2335, 2336, 2337, 2339, 2344, 2357, 2362, 2363, 2364, 2367, 2368, 2369, 2370, 2372, 2373, 2424, 2425, 2426, 2427, 2428, 2432, 2433, 2459	<code>\l__stex_morphisms_feature_str</code> 2953, 3038
<code>__stex_morphisms_add_definiens:nn</code> 3142, 3147, 3206	<code>__stex_morphisms_get_check:nn</code> 3254, 3269, 3315, 3323
	<code>\l__stex_morphisms_get_str</code> 3252, 3275, 3296, 3306, 3311, 3313, 3319, 3321

<code>__stex_morphisms_gobble:nnnnnnN</code>	<code>__stex_notations_activate_-</code>
..... 3299, 3303	not:nn 6344, 6356
<code>__stex_morphisms_iterate:nn</code> ...	<code>__stex_notations_add:nnnnn</code>
..... 3217, 3221, 3230, 3232 6675, 6680, 6685
<code>__stex_morphisms_macro:nnnnnnN</code>	<code>__stex_notations_add_last:nnnnn</code>
..... 3279, 3295 6668, 6679
<code>\l__stex_morphisms_mods_seq</code>	<code>__stex_notations_add_missing_-</code>
..... 3210, 3234, 3235	args:nn 6310, 6313
<code>__stex_morphisms_morphism:nnnnn</code>	<code>__stex_notations_add_next:nnnnnn</code>
..... 2924, 2928, 2931 6670, 6674
<code>\l__stex_morphisms_morphism_dom_-</code>	<code>\l__stex_notations_after_tl</code>
str ... 3567, 3580, 3592, 3602, 3619 6655, 6682
<code>\l__stex_morphisms_name_str</code>	<code>__stex_notations_args_end:</code>
..... 3490, 3499, 3500, 3501, 6389, 6392, 6395,
3505, 3506, 3507, 3518, 3520, 3524	6402, 6405, 6408, 6663, 6666, 6676
<code>\l__stex_morphisms_newname_str</code> ..	<code>\l__stex_notations_args_tl</code>
.. 3491, 3510, 3511, 3519, 3520, 3521 6574, 6576, 6589
<code>\l__stex_morphisms_next_tl</code>	<code>__stex_notations_augment_arg:nn</code>
..... 3497, 3502, 3504 6590, 6603
<code>__stex_morphisms_parse_assign:n</code>	<code>\l__stex_notations_bind_bool</code> ...
..... 3379, 3445, 3489 6109, 6111
<code>__stex_morphisms_reactivate:</code> ...	<code>__stex_notations_check_aB_-</code>
..... 2965, 3194	arg:Nn 6717, 6726, 6736, 6761
<code>__stex_morphisms_rename:n</code> 3039, 3042	<code>\l__stex_notations_clist_count_-</code>
<code>__stex_morphisms_rename:nn</code>	int 6759, 6767, 6773
..... 3043, 3046	<code>\l__stex_notations_code_tl</code>
<code>__stex_morphisms_rename_all:</code> . 3167 6613, 6628, 6643, 6657,
<code>__stex_morphisms_renamed_-</code>	6658, 6681, 6689, 6697, 6702, 6710
check:nn 3258, 3305	<code>__stex_notations_complex:nnnnnn</code>
<code>__stex_morphisms_renamed_-</code> 6622, 6639
check:nnnnnn 3307, 3310	<code>__stex_notations_const_precs:</code> ..
<code>\l__stex_morphisms_seq</code> 6183, 6223
..... 3494, 3495, 3497, 3499,	<code>\l__stex_notations_cs</code>
3502, 3504, 3506, 3508, 3510, 3512 6629, 6634, 6644, 6653
<code>__stex_morphisms_set:nnnnnnN</code> ...	<code>__stex_notations_default_args:</code> .
..... 3277, 3283, 3298 6575, 6586
<code>__stex_morphisms_set_definiens_-</code>	<code>__stex_notations_do_argname:nn</code> .
macros: 3150, 3154 6284, 6287
<code>__stex_morphisms_set_definiens_-</code>	<code>__stex_notations_do_argnames:</code> ..
macros_i:nnnnnnn 3157, 3162 6259, 6282
<code>__stex_morphisms_setup:</code> . 2964, 2969	<code>__stex_notations_do_missing_-</code>
<code>__stex_morphisms_split_qm:w</code> . 3190	args:n 6189, 6302
<code>\l__stex_morphisms_tmp</code> 3051,	<code>__stex_notations_fun_precs:</code> ...
3055, 3058, 3059, 3065, 3072, 3107 6188, 6235
<code>\l__stex_morphisms_tmp_b</code>	<code>__stex_notations_it_not_-</code>
..... 3083, 3087, 3099	check:nnnn 6475, 6479
<code>\l__stex_morphisms_todo_tl</code>	<code>__stex_notations_it_not_i:n</code> ...
..... 3216, 3220, 3233, 3246 6464, 6468, 6481
<code>__stex_morphisms_total_check:nn</code>	<code>\l__stex_notations_left_bracket_-</code>
..... 3174, 3178	str 6790, 6818, 6828, 6832
<code>__stex_morphisms_total_check:nnnnnnN</code>	<code>__stex_notations_macro:nn</code>
..... 3179, 3182	6327, 6375, 6376, 6377, 6412, 6413,
<code>\l__stex_morphisms_with_macros_-</code>	6414, 6432, 6443, 6444, 6485, 6486
bool ... 2920, 2923, 2927, 3013, 3074	

<code>__stex_notations_make_arg:nnnn</code> .	<code>__stex_others_old_newlabel:</code> . . .
. 6553, 6556 7858, 7863
<code>__stex_notations_make_arg_-</code>	<code>__stex_path:</code> 148, 158
<code>html:nn</code> 6533, 6549	<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
. 6582, 6587, 6607	<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
. 6388, 6392, 6395	<code>\l__stex_path_can_seq</code>
<code>__stex_notations_map_args_ii:w</code> 180, 183, 188, 196, 197, 200
. 6401, 6405, 6408	<code>\l__stex_path_can_str</code>
<code>__stex_notations_map_cs:</code> 179, 181, 185, 197
. 6518, 6520,	<code>__stex_path_canonicalize:N</code>
6739, 6741, 6749, 6764, 6766, 6777 161, 172, 177
<code>\l__stex_notations_missing_str</code> . .	<code>__stex_path_dodots:n</code> . 182, 186, 192
. 6303, 6304, 6305, 6307, 6315	<code>\l__stex_path_return_tl</code>
<code>\l__stex_notations_missing_tl</code> 282, 288, 295, 298, 300
. 6218, 6309, 6316	<code>\l__stex_path_seq</code> 211, 212,
<code>\l__stex_notations_mods_seq</code>	213, 218, 219, 221, 223, 226, 251, 252
. 6460, 6469, 6470	<code>\l__stex_path_str</code> 149, 150, 151, 154,
<code>\l__stex_notations_name_str</code>	156, 157, 158, 160, 163, 170, 171,
. 6583, 6609	210, 211, 212, 217, 218, 219, 221,
<code>__stex_notations_not_cs:nnnnn</code> . .	222, 223, 229, 231, 233, 239, 241, 243
. 6461, 6462, 6472	<code>\l__stex_path_win_drive</code>
<code>__stex_notations_op_macro:nn</code> 150, 155, 162, 164
6330, 6380, 6381, 6382, 6418, 6419,	<code>__stex_path_win_take:w</code> . . . 148, 158
6420, 6435, 6445, 6446, 6491, 6492	<code>__stex_persist_env_str</code>
<code>\l__stex_notations_opprec_tl</code> 6212, 568, 569, 570, 574, 575, 576
6225, 6228, 6230, 6237, 6240, 6242,	<code>__stex_persist_load_file:n</code>
6246, 6253, 6266, 6272, 6278, 6509 591, 598, 630
<code>__stex_notations_parse_args:nnnnw</code>	<code>__stex_persist_read_and_write:</code> .
. 6663, 6666, 6676 589, 615
<code>__stex_notations_parse_precs:</code> . .	<code>\c__stex_persist_sms_iow</code>
. 6257, 6262 585, 611, 612, 627, 715
<code>\l__stex_notations_pre_tl</code>	<code>__stex_persist_write_only:</code>
. 6642, 6657, 6694, 6707 594, 610, 624, 631
<code>\l__stex_notations_precs_seq</code> . . .	<code>__stex_proof_add_counter:</code> 7398, 7581
. 6180, 6248,	<code>__stex_proof_begin_proof:nn</code> . . .
6255, 6276, 6278, 6290, 6296, 6510 7424, 7502, 7524
<code>__stex_notations_process:nnnnnn</code>	<code>\l__stex_proof_counter_intarray</code> .
. 6612, 6618 7355, 7361,
<code>\l__stex_notations_right_-</code>	7364, 7373, 7376, 7385, 7393, 7394,
<code>bracket_str</code> . 6791, 6823, 6829, 6833	7402, 7407, 7414, 7420, 7425, 7426
<code>\l__stex_notations_seq</code>	<code>__stex_proof_end_comment:</code> 7494,
. 6264, 6265, 6267, 6268, 6275	7518, 7611, 7619, 7660, 7721, 7727
<code>__stex_notations_set_macro:nnnnn</code>	<code>__stex_proof_end_list:</code>
. 6343, 6352, 6365, 6374	. . 7508, 7543, 7602, 7606, 7677, 7734
<code>__stex_notations_simple:nnnnn</code> . .	<code>__stex_proof_html:</code> . 7470, 7473, 7703
. 6620, 6626	<code>__stex_proof_html_env:n</code>
<code>\l__stex_notations_str</code> 7430, 7459, 7553
. . 6265, 6266, 6267, 6269, 6275, 6276	<code>\l__stex_proof_in_spfblock_bool</code> .
<code>__stex_notations_styledefs:</code> 7422,
. 6131, 6142	7490, 7496, 7503, 7525, 7571, 7574,
<code>__stex_others_newlabel:n</code> 7856, 7864	7595, 7602, 7606, 7673, 7714, 7730

__stex_proof_inc_counter:	\l__stex_refs_uri
. 7381, 7597, 7606, 7723, 7724	1827, 1828,
\l__stex_proof_inc_counter_bool 7354	1829, 1833, 1842, 1843, 1862, 1871,
__stex_proof_insert_number: . . .	1872, 1875, 1880, 1912, 1931, 1943
. 7357, 7580, 7723, 7724	__stex_seqs_add: 4366, 4381
__stex_proof_make_step_macro:Nnnnn	\l__stex_seqs_args_tl . . . 4436, 4438
. 7658, 7723, 7724, 7725	\l__stex_seqs_cs 4434, 4438
__stex_proof_number_as_string:N	__stex_seqs_html: 4364, 4408
. 7368, 7567, 7576, 7668	__stex_seqs_macro: 4367, 4395
__stex_proof_proof_box_tl 7337, 7752	\l__stex_seqs_range_clist
__stex_proof_remove_counter: 4357, 4389, 4402
. 7410, 7596	\g__stex_smsmode_allowed_escape_-
__stex_proof_start_comment: 7488,	tl 2567, 2570, 2697
7505, 7593, 7615, 7681, 7716, 7737	\g__stex_smsmode_allowed_import_-
__stex_proof_start_list:n	env_seq 2592, 2595, 2720, 2723
. 7504, 7534, 7580, 7677, 7734	\g__stex_smsmode_allowed_import_-
__stex_proof_step_html:nn 7675,	tl 2585, 2589, 2715
7677, 7683, 7690, 7732, 7734, 7739	\g__stex_smsmode_allowed_tl
__stex_refs_check_i:nnnn 1852, 1860 2562, 2565, 2693
__stex_refs_check_in:nnnn 1934, 1942	\g__stex_smsmode_allowedenvs_seq
\l__stex_refs_default_archive 2572, 2575, 2702, 2705
. 1961, 1966, 1968, 1970	\g__stex_smsmode_bool 2601, 2604, 2644
\l__stex_refs_default_file 1874,	__stex_smsmode_check_begin:Nn . . .
1878, 1879, 1882, 1960, 1964, 1971 2702, 2720, 2731
\l__stex_refs_default_title	__stex_smsmode_check_cs:N 2666, 2672
. 1877, 1959, 1973	__stex_smsmode_check_cs:Nnn
\l__stex_refs_file 1834, 1835, 1853, 2665, 2679
1878, 1890, 1899, 1902, 1931, 1935	__stex_smsmode_check_end:Nn
__stex_refs_find: 1836, 1850 2705, 2723, 2739
__stex_refs_find_in: . . . 1900, 1930	\l__stex_smsmode_cycles_seq
\l__stex_refs_in 1825, 1870, 1888, 1892 2607, 2613, 2615
__stex_refs_in: 1883, 1893, 1897	__stex_smsmode_do:w
__stex_refs_in_text:nn . . 1910, 1916	. . . 2661, 2663, 2665, 2668, 2676,
__stex_refs_in_text:w . . . 1917, 1924	2695, 2707, 2725, 2736, 2742, 2744
\c__stex_refs_iow	__stex_smsmode_do_aux:N
. 1748, 1749, 1750, 1771 2665, 2675, 2686
\c__stex_refs_iow_str . . . 1751, 1879	__stex_smsmode_do_aux_curr:N
\l__stex_refs_key 1824, 1861 2624, 2632, 2688
__stex_refs_local:n 1796, 1802, 1831	__stex_smsmode_do_aux_imports:N
__stex_refs_local_ref:n 2624, 2714
. 1806, 1812, 1872, 1875, 1880	__stex_smsmode_do_aux_normal:N
__stex_refs_maybe_in: . . . 1844, 1869 2632, 2692
\l__stex_refs_new_tl	\g__stex_smsmode_import_setup_tl
. 1765, 1766, 1767, 1769, 1774 2586, 2590, 2596, 2625
__stex_refs_remote:nn . . . 1798, 1823	\l__stex_smsmode_importmodules_-
__stex_refs_stop: 1917, 1924	seq 2610
\l__stex_refs_str 7207	__stex_smsmode_in_smsmode:n
\l__stex_refs_tmp 1794, 2622, 2641
1838, 1842, 1856, 1863, 1898, 1904,	\l__stex_smsmode_sigmodules_seq 2611
1908, 1909, 1910, 1912, 1938, 1944	__stex_smsmode_smsmode_do:
\l__stex_refs_unnamed_counter_- 2654, 2661
int 1948	__stex_smsmode_start_smsmode:n
__stex_refs_uppercase:n . 1925, 1928 2626, 2633, 2652
	__stex_statements_add_definiens:n
 7285, 7290

```

\__stex_statements_comma_sep_-
  uri:nn ..... 7147, 7157
\__stex_statements_definiens_-
  inner:nnnnnnnN ..... 7294, 7298
\__stex_statements_force_id: ...
  ..... 7176, 7204, 7224, 7248
\__stex_statements_html_keyvals:nn
  ..... 7098, 7132, 7143
\__stex_statements_make_macro:nnn
  ..... 7122, 7125
\__stex_statements_setup:n 7161,
  7161, 7225, 7229, 7244, 7249, 7252
\__stex_statements_setup_def: ...
  ..... 7210, 7210, 7226, 7250
\__stex_statements_setup_named:n
  ..... 7171, 7175
\__stex_statements_setup_noname:
  ..... 7170, 7194
\__stex_structures_begin:nn ....
  ..... 6854, 6864, 6898, 6939
\__stex_structures_check_-
  def:nnnnnnnn ..... 6957, 7000
\__stex_structures_do_externals:
  ..... 6858, 6884, 6913, 6963
\l__stex_structures_exstruct_-
  name_str ... 6936, 6939, 6961, 7010
\__stex_structures_extend_-
  i:nnnnnnnN ..... 6982, 6996
\__stex_structures_extend_-
  ii:nnnn ..... 6981, 6988
\__stex_structures_extend_-
  structure:nnnn ..... 6936, 6973
\l__stex_structures_extname_-
  count ..... 7008, 7010, 7013
\__stex_structures_get:w . 7039, 7045
\l__stex_structures_imports_seq .
  ..... 6889, 6894, 6899,
  6924, 6929, 6941, 7049, 7052, 7055
\l__stex_structures_last_name_-
  str ..... 6933, 6936
\l__stex_structures_name_str ...
  ..... 6842, 6847, 6849, 6866,
  6867, 6872, 6876, 6881, 7021, 7024
\__stex_structures_new_extstruct_-
  name: ..... 6923, 7006
\l__stex_structures_parent_uri .
  ..... 6933, 6936
\l__stex_structures_replace_-
  this_tl ..... 6885
\l__stex_structures_struct_uri .
  ..... 6871, 6876
\__stex_styles_apply_patch:n 488, 503
\g__stex_styles_counters_seq ...
  ..... 24, 63, 64, 746, 785, 786

\__stex_styles_css_patch:nnnn ...
  ..... 16, 97, 521, 738, 819
\l__stex_styles_first_str .....
  ..... 60, 62, 81, 782, 784, 803
\__stex_styles_patch:nnn ... 484, 495
\__stex_styles_patch:nnnn .....
  ..... 4, 88, 518, 726, 810
\__stex_styles_patch_html: .....
  ..... 34, 74, 77, 756, 796, 799
\__stex_styles_patch_html_c: ...
  ..... 26, 66, 69, 748, 788, 791
\__stex_styles_patch_i:nnn .....
  ..... 21, 42, 743, 764
\__stex_syms_activate_i:nnnnnnnn
  ..... 3966, 3979, 3983
\__stex_syms_add_decl: ... 3818, 3821
\l__stex_syms_args_tl ... 3939, 3941
\l__stex_syms_cs .....
  .. 3937, 3941, 4061, 4063, 4065, 4089
\__stex_syms_def_style: .. 3703, 3711
\__stex_syms_do_args: ... 3838, 3845
\__stex_syms_get_from_one_-
  string:n ..... 4098, 4172
\__stex_syms_get_symbol_from_cs:
  ..... 4067, 4078
\__stex_syms_get_symbol_from_-
  modules:nn ..... 4100, 4131
\__stex_syms_get_symbol_from_-
  string:n ... 4068, 4070, 4073, 4093
\__stex_syms_has_definiens:nnnnnnnN
  ..... 3997, 4004
\__stex_syms_it_decl_check:nnnn .
  ..... 4037, 4045
\__stex_syms_it_decl_i:n .....
  ..... 4029, 4033, 4047
\__stex_syms_maybe_activate:nnnnnnnn
  ..... 3973, 3977
\l__stex_syms_mods_seq .....
  ..... 4025, 4034, 4035
\l__stex_syms_name ..... 4096, 4102
\l__stex_syms_name_str .....
  ..... 4133, 4145, 4147
\__stex_syms_parse_arity: 3837, 3856
\l__stex_syms_path_str .....
  ..... 4134, 4144, 4151, 4156, 4157
\l__stex_syms_seq .....
  ..... 4095, 4096, 4097, 4101
\__stex_syms_set_textsymdecl_-
  macro:nnn ..... 3781, 3800
\__stex_syms_style: ..... 3672, 3680
\__stex_syms_sym_cs:nnnnnnnnN ...
  .. 4008, 4009, 4019, 4026, 4027, 4040
\__stex_syms_sym_from_str_i:nnnn
  ..... 4106, 4140, 4153, 4159

```

_stex_syms_sym_i_finish:nnnnnnN	\l_stex_vars_bind_bool .. 4195, 4276
..... 4112, 4119	_stex_vars_check_var:nnnnnnnnN
_stex_syms_sym_i_gobble:nnnnnn 4319, 4323
..... 4114, 4117	\l_stex_vars_cs 4290, 4294
_stex_syms_top:n 3671, 3694, <u>3808</u> , 3808	_stex_vars_get_var:n ... 4311, 4317
_stex_syms_uri_match:n 4167	_stex_vars_html: 4209, 4259
_stex_uris_absolute:N 1281, 1312, 1316	_stex_vars_macro: 4208, 4246
\l_stex_uris_archive 1142, 1146, 1149	_stex_vars_set_vars:nnnnnnN ... 4325, 4328, 4333
_stex_uris_as_qm:nnn ... 1243, 1245	stex_annotate_env (env.) <u>120</u> , <u>694</u>
_stex_uris_doc:nnnnn 1090, 1093, 1095	stex_env_node (env.) <u>120</u> , <u>694</u>
_stex_uris_doc_from_archive_-	\stexcommentfont <u>151</u> , 7491, <u>7763</u>
file:NN 1136, 1139, 1186, 1188	\stexdoctitle <u>130</u> , <u>1434</u> , 1537, 1549, 1575, 2271, 8194, 8370, 8625, 8725, 8728, 9797
_stex_uris_end: ... 1354, 1357, 1361	\STEXexport <u>41</u> , <u>42</u> , <u>81</u> , <u>82</u> , <u>135</u> , <u>2527</u> , 2584
\l_stex_uris_file .. 1134, 1135, 1136	\STEXftmlink <u>130</u> , <u>1407</u>
_stex_uris_first_three:nnnn ... 1391, 1394, 1397	\stexhtmlfalse 677
\l_stex_uris_lang_str 1152, 1161, 1163, 1164, 1166, 1167, 1168, 1170, 1171, 1172	\stexhtmltrue 674
_stex_uris_maybe_relative:N ... 1281, 1289	\TeXInternalNewSRefLabel 1767, 1769, 1955
_stex_uris_module:nnn 1196, 1199, 1201	\TeXInternalNotation <u>146</u> , 6210, 6576, <u>6611</u>
\l_stex_uris_name_str 1151, 1159, 1160, 1163, 1175, 1274, 1299, 1305, 1321, 1336, 1345	\TeXInternalSRefLabel 1772, 1852, 1934, 1953
_stex_uris_new_mod:nnnnnn 1252, 1256	\STEXinvisible ... <u>76</u> , <u>121</u> , <u>694</u> , 6316, 7435, 7437, 7558, 7560, 7791, 7792
_stex_uris_new_nested_mod:nnnn 1250, 1264	\STEXlinkftml <u>130</u> , <u>1407</u>
_stex_uris_pair_in_archive:Nn . 1284, 1341	\STEXRestoreNotsEnd 2402, 2440, 2447
_stex_uris_pair_no_archive:N .. 1325, 1329	\STEXsetlinkftml <u>130</u> , <u>1407</u>
_stex_uris_parent:NNnnn 1225, 1229, 1232	\stexstyle⟨environmentname⟩ <u>119</u>
_stex_uris_parse_i:Nw .. 1351, 1357	\stexstyle⟨macroname⟩ <u>119</u>
_stex_uris_parse_ii:Nw . 1353, 1361	\stexstyleassertion <u>99</u>
\l_stex_uris_path_seq 1150, 1158, 1159	\stexstyleassign <u>99</u>
\l_stex_uris_path_str 1275, 1277, 1280, 1320, 1331, 1344	\stexstyleassignMorphism <u>99</u>
_stex_uris_split_file:N 1140, 1157	\stexstylecopymod <u>99</u>
_stex_uris_symbol:nnnn 1365, 1368, 1370	\stexstylecopymodule <u>99</u>
_stex_uris_with_elem:nnnnnn ... 1128, 1130	\stexstyledefinition <u>99</u>
_stex_uris_with_language:nnnnnn 1122, 1124	\stexstyleexample <u>99</u>
_stex_vars_add: 4207, 4233	\stexstyleextstructure <u>99</u>
\l_stex_vars_args_tl ... 4292, 4294	\stexstylegnote 9683
	\stexstyleimportmodule <u>99</u>
	\stexstyleinterpretmod <u>99</u>
	\stexstyleinterpretmodule <u>99</u>
	\stexstylemathstructure <u>99</u>
	\stexstylemcb 9104, 9108
	\stexstyleMMTinclud <u>99</u>
	\stexstylemodule <u>99</u>
	\stexstylenotation <u>99</u>
	\stexstyleparagraph <u>99</u> , <u>119</u>
	\stexstyleproblem 9563, 9581
	\stexstyleproof <u>99</u>
	\stexstylerealization <u>99</u>
	\stexstylerealize <u>99</u>
	\stexstylerenamedec <u>99</u>
	\stexstylerequiremodule <u>99</u>

<code>\stexstylescb</code>	9269, 9273	8713, 8790, 8833, 8853, 8901, 8951,
<code>\stexstylespfsketch</code>	99	8992, 9011, 9024, 9042, 9685, 9702,
<code>\stexstylesubproblem</code>	9623, 9642	9728, 9740, 9743, 9949, 9952, 9963
<code>\stexstylesubproof</code>	99	<code>\str_if_empty:nTF</code>
<code>\stexstylesymdecl</code>	99	... 89, 142, 193, 496, 811, 3554, 3978
<code>\stexstylesymdef</code>	99	<code>\str_if_eq:NNTF</code>
<code>\stexstyletextsymdecl</code>	99	294, 1879, 2776
<code>\stexstyleusemodule</code>	99	<code>\str_if_eq:nnTF</code> ..
<code>\stexstylevardef</code>	99	58, 63, 90, 157,
<code>\stexstylevarnotation</code>	99	194, 195, 266, 462, 570, 576, 653,
<code>\stexstylevarseq</code>	99	1003, 1258, 1299, 1861, 1943, 2814,
<code>\stopsolutions</code>	106, 8842	3063, 3215, 3275, 3296, 3311, 3319,
str commands:		3572, 3588, 4147, 4168, 4324, 4327,
<code>\c_ampersand_str</code> .	1098, 1100, 1101,	4597, 6035, 6227, 6239, 6251, 6361,
1103, 1204, 1206, 1373, 1375, 1376		7284, 8012, 8015, 8080, 8258, 9114
<code>\c_backslash_str</code>	156, 1817	<code>\str_if_eq:p:nn</code> 4108, 4109, 4177, 4178
<code>\c_colon_str</code>	987	<code>\str_if_in:NnTF</code>
<code>\c_dollar_str</code>	6297	5982, 6006, 6315
<code>\c_hash_str</code>	6304, 6315, 6336	<code>\str_item:Nn</code>
<code>\c_left_brace_str</code>		157, 3850
.....	9602, 9669, 9689, 9714	<code>\str_item:nn</code>
<code>\c_percent_str</code>	89, 90, 239	266
<code>\c_right_brace_str</code>		<code>\str_lowercase:n</code>
.....	9619, 9679, 9695, 9718	9114
<code>\str_case:Nn</code>	991	<code>\str_map_break:</code>
<code>\str_case:nn</code>	5284, 6687	3861
<code>\str_case:nnTF</code> 48, 770, 1634, 3860,		<code>\str_map_break:n</code> .
5311, 5339, 6557, 7954, 8492, 8515		3862, 3866, 3870,
<code>\str_clear:N</code> .	6, 7, 8, 91, 155, 379,	3874, 3878, 3882, 3886, 3890, 3894
380, 387, 405, 728, 729, 730, 834,		<code>\str_map_function:nN</code>
938, 1275, 2190, 2233, 2790, 3490,		9536
3491, 3567, 3632, 3633, 3634, 3635,		<code>\str_map_inline:Nn</code>
3645, 3646, 3650, 3667, 3729, 3771,		3859
3772, 4310, 6086, 6087, 6088, 6089,		<code>\str_new:N</code>
6305, 6842, 7065, 7066, 7068, 7369,		...
7648, 8088, 8409, 8410, 8559, 8560,		453, 1436, 1960, 1961, 3662, 9518
8764, 8781, 9002, 9925, 9926, 9927		<code>\str_put_left:Nn</code>
<code>\str_const:Nn</code>	1020, 7873	62, 784
<code>\str_count:n</code>	59, 64	<code>\str_put_right:Nn</code>
<code>\str_gput_right:Nn</code>	9526	7376
<code>\str_gset_eq:NN</code>	829	<code>\str_range:Nnn</code>
<code>\str_if_empty:NnTF</code>		2066
..	43, 61, 120, 162, 181, 458, 569,	<code>\str_range:nnn</code>
575, 635, 641, 765, 783, 822, 824,		59, 64
843, 929, 932, 1035, 1280, 1302,		<code>\str_replace_all:Nnn</code>
1795, 2258, 2292, 2545, 2760, 2812,		156
3505, 3519, 3580, 3743, 3809, 3912,		<code>\str_set:Nn</code> 13, 14, 44, 46, 50, 51, 52,
3915, 3918, 3921, 3924, 4144, 4151,		53, 54, 55, 56, 60, 83, 94, 149, 150,
4201, 4264, 4267, 4270, 4273, 4312,		151, 154, 170, 254, 459, 461, 735,
4348, 4351, 4412, 4415, 4418, 4421,		766, 768, 772, 773, 774, 775, 776,
4964, 6034, 6098, 6224, 6236, 6245,		777, 778, 782, 864, 968, 973, 981,
6291, 6846, 6866, 7149, 7162, 7163,		982, 983, 989, 990, 999, 1142, 1172,
7170, 7205, 7566, 7575, 7662, 7667,		1175, 1179, 1192, 1438, 1484, 1751,
7699, 7944, 8008, 8185, 8613, 8708,		1824, 1833, 1834, 1863, 1890, 1964,
		1966, 1968, 1971, 1973, 2001, 2002,
		2064, 2131, 2142, 2153, 2158, 2169,
		2174, 2184, 2187, 2191, 2277, 2278,
		2781, 2795, 2799, 2803, 2833, 2861,
		2862, 2892, 2893, 2949, 2953, 3155,
		3156, 3252, 3285, 3313, 3321, 3501,
		3507, 3511, 3619, 3640, 3669, 3693,
		3742, 3744, 3746, 3748, 3810, 3864,
		3868, 3872, 3876, 3880, 3884, 3888,
		3892, 3896, 4134, 4200, 4202, 4336,
		4347, 4349, 4352, 4529, 4960, 4965,
		5981, 5983, 6007, 6099, 6144, 6303,
		6847, 6867, 6878, 7008, 7010, 7013,
		7181, 7568, 7577, 7669, 7998, 8792,
		8855, 8903, 8953, 9013, 9704, 9948

<code>\str_set_eq:NN</code>	840, 992, 993, 1146, 1163, 1164, 1168, 1842, 1878, 2777, 2791, 2954, 3696, 3719, 4224, 4373, 6101, 6143, 6155, 6165, 6849, 7024, 7101, 7164, 7180, 7207, 7547, 7711	<code>\testsmallspace</code>	. 68, 68, 68, 110, 110, 110
<code>\str_uppercase:n</code>	1500, 8080	<code>\testsmallspace</code> 9468
<code>\l_tmpa_str</code> 461, 465, 466, 467, 469, 973, 975, 1179, 1180, 2949, 2951, 2954, 2958	<code>\testspace</code> 68, 110
<code>\string</code> 1830, 9512, 9553	<code>\testspace</code> 8813, <u>9467</u>
<code>\subparagraph</code> 26, 77	<code>\TeX</code> 22, 23
<code>subproblem (env.)</code> 8695	<code>\tex</code> 23
<code>subproof (env.)</code> 150, <u>7547</u>	T _E X and L ^A T _E X 2 _ε commands:	
<code>\subproof</code> 7442, 7623	<code>\@</code> 2096, 2099, 2103
<code>\subproofautorefname</code> 7547	<code>\@addtoreset</code> 8575
<code>\subsection</code> 25, 26, 77, 131	<code>\@arabic</code> 8381
<code>\subsubsection</code> 26, 77	<code>\@author</code> 8360
<code>\svar</code> 90, 142, 3842, <u>5143</u>	<code>\@auxout</code> 1768, 7861, 8689
<code>\symbol</code> 84	<code>\@bonuspointsfalse</code> 9893
<code>\symdecl</code>	... 22, 29, 30, 34, 35, 40, 82– 84, 94, 99, 135, 139, 141, 247, 3195, <u>3662</u> , 7802, 7804, 7819, 7827, 7830	<code>\@bonuspointstrue</code> 9898
<code>\symdef</code> 30–32, 36, 83, 90, 135, 139, 141, 247, 3197, <u>3691</u> , 7786, 7789, 7795, 7797, 7799, 7801, 7811, 7812, 7813, 7814, 7825, 7834, 7835, 7838, 7841, 7843, 7845	<code>\@currentHref</code>	. 1775, 7568, 7577, 7669
<code>\Symname</code> 84, 95, 144, <u>5856</u>	<code>\@currentlabel</code> 7567, 7568, 7576, 7577, 7668, 7669, 8184
<code>\symname</code> 18, 19, 23, 54, 84, 85, 95, 97, 144, <u>5856</u>	<code>\@currentlabelname</code> 1776
<code>\symref</code> 18, 19, 23, 41, 83, 84, 88, 95, 97, 144, <u>5849</u>	<code>\@currenvir</code> 8054, 8055
<code>\symrefemph</code> 97, 98, 143, <u>5810</u>	<code>\@dblarg</code> 8353, 8362
<code>\symuse</code> 41, 84, 141, <u>4453</u> , 4747, 4754, 4765, 4848, 4914, 4991, 4992, 5749, 6733	<code>\@gobble</code> 48
sys commands:		<code>\@ifnextchar</code> 47
<code>\sys_get_shell:nnN</code> 80	<code>\@ifstar</code> 8301
<code>\sys_if_platform_windows:TF</code> 86, 147, 228, 238, 262	<code>\@mainmatterfalse</code> 1708, 1718
T		<code>\@mainmattertrue</code> 1729, 1740
		<code>\@notprerr</code> 1645
<code>\target</code> 116	<code>\@onlypreamble</code> 1645
<code>\test</code> 69, 110	<code>\@rustexfalse</code> 649
<code>test</code> 105	<code>\@title</code> 1477, 1478, 8369
<code>\testbigspace</code> 9470	<code>\activate@excursion</code> 8393, 8398
<code>\testemptypage</code> 68, 110	<code>\bbl@loaded</code> 8539, 9494
<code>\testemptypage</code> <u>9464</u> , 9760	<code>\beamer@shortauthor</code>	. 8356, 8358, 8373
<code>testheading (env.)</code> 70, 111	<code>\beamer@shorttitle</code>	. 8365, 8367, 8376
<code>\testheading</code> <u>9862</u>	<code>\c@chapter</code> 8035
<code>\testmedspace</code> 9469	<code>\c@framenumber</code> 8381
<code>\testnewpage</code> 68, 110	<code>\c@part</code> 8047
<code>\testnewpage</code> <u>9471</u> , <u>9472</u>	<code>\compemph@uri</code>	98, 143, <u>5766</u> , 5990, 6524
<code>\testnewpageInProblem</code> 9472	<code>\correction@table</code> <u>9841</u>
		<code>\defemph@uri</code> 98, 143, <u>5799</u> , 6013
		<code>\define@key</code> 2151, 2167, 2182
		<code>\Gin@eheight</code>	10013, 10016, 10021, 10026
		<code>\Gin@ewidth</code> 8292, 8293, 10012, 10022, 10026
		<code>\Gin@exclamation</code>	. 10012, 10013, 10021
		<code>\Gin@mhrepos</code> 2153, 2158, 2162, 2184, 2191, 2196
		<code>\hwexam@bonuspts</code> 9895
		<code>\hwexam@checktime</code> 9889
		<code>\hwexam@duration</code>	9865, 9868, 9874, 9879
		<code>\hwexam@kw@due</code> 9816
		<code>\hwexam@kw@forgrading</code> 9857
		<code>\hwexam@kw@given</code> 9813
		<code>\hwexam@kw@grade</code> 9858
		<code>\hwexam@kw@probs</code> 9858

<code>\hwexam@kw@pts</code>	9859	<code>\textbf</code>	18, 5804, 7935, 8501, 8524, 8677, 8680, 9445, 9806, 9811
<code>\hwexam@kw@reached</code>	9860	<code>\textcolor</code>	9441
<code>\hwexam@kw@sum</code>	9858	<code>\textsymdecl</code>	22, 83, 139, 141, 3196, 3728
<code>\hwexam@kw@testemptypage</code>	9465, 9764	<code>\texttt</code>	9425, 9441
<code>\hwexam@min</code>	9866, 9873, 9878, 9889	<code>\textwarning</code>	103
<code>\hwexam@minutes@kw</code>	9866	<code>\textwidth</code>	8161, 8290, 8312, 9854
<code>\hwexam@reqpts</code>	9875, 9880, 9892, 9896	<code>\the</code>	334, 335, 337, 339, 341, 353, 355, 2096, 2097, 2098
<code>\hwexam@tools</code>	9876, 9881	<code>\theassignment</code>	9794, 9806
<code>\hwexam@totalmin</code>	9888, 9889	<code>\thechapter</code>	1524, 1590, 1623, 1663, 1674, 7961, 7966, 7970, 7974, 7978, 7982
<code>\hwexam@totalpts</code>	9887, 9896	<code>\theframenumber</code>	8184
<code>\if@bonuspoints</code>	9891	<code>\thepage</code>	9762
<code>\if@rustex</code>	657	<code>\theparagraph</code>	7978, 7982
<code>\itemize@inner</code>	8120, 8126, 8135	<code>\thepart</code>	1520, 1587, 1619, 1657, 7956
<code>\itemize@label</code>	8124, 8127, 8130	<code>\theplainsproblem</code>	8579, 8580
<code>\itemize@level</code>	8118, 8123, 8126, 8135	<code>\thesection</code>	7966, 7970, 7974, 7978, 7982, 8580
<code>\itemize@outer</code>	8119, 8123	<code>\thesproblem</code>	8576, 8580, 8680, 8690, 9564, 9582, 9604, 9629, 9648, 9671, 9794
<code>\lst@mhrepos</code>	2169, 2174, 2178	<code>\thesubparagraph</code>	7982
<code>\ltx@ifpackageloaded</code>	2150, 2166, 2181, 7753, 8538, 9138, 9169, 9493	<code>\thesubsection</code>	7970, 7974, 7978, 7982
<code>\mdf@patchamsthm</code>	8157	<code>\thesubsubsection</code>	7974, 7978, 7982
<code>\metakeys@show@keys</code>	8121	<code>\this</code>	49, 50, 91, 92, 148, 263, 4526, 4864, 4948, 4949, 4954, 4979, 7014
<code>\notesslides@slidelabel</code>	8160, 8176	<code>\thisarchive</code>	2891
<code>\ns@author</code>	8353, 8354	<code>\thisargs</code>	3686, 3717
<code>\ns@title</code>	8362, 8363	<code>\thiscopyname</code>	3356, 3381, 3423, 3447
<code>\pgf@temp</code>	2093, 2094, 2095	<code>\thisdeclname</code>	3683, 3714, 3788, 6144
<code>\pgfkeys@spdef</code>	2093	<code>\thisdecluri</code>	3682, 3713, 3787, 6145
<code>\pgfutil@empty</code>	2095	<code>\thisdefiniens</code>	3685, 3716
<code>\pgfutil@InputIfFileExists</code>	2102	<code>\thisfor</code>	7102
<code>\prematurestop@endsofragment</code>	8053, 8056, 8062	<code>\thismodulename</code>	2278, 2862, 2893
<code>\problem@kw@*</code>	8535	<code>\thismoduleuri</code>	2277, 2861, 2892, 3357, 3382, 3424, 3448
<code>\problem@kw@correct</code>	9153, 9291	<code>\thisname</code>	7101
<code>\problem@kw@feedback</code>	9046, 9744	<code>\thisnotation</code>	3720, 4225, 4374, 6146, 6166
<code>\problem@kw@grading</code>	8992, 9685	<code>\thisnotationvariant</code>	3719, 4224, 4373, 6143, 6165
<code>\problem@kw@hint</code>	8892	<code>\thisstyle</code>	505, 508, 509, 510, 543, 546, 547, 548, 549, 557, 560, 561, 2863, 2894, 3687, 3718, 3789, 4223, 4372, 6164
<code>\problem@kw@minutes</code>	8663, 8756, 9059, 9572, 9591, 9637, 9657	<code>\thistitle</code>	98, 2269, 2270, 2271, 7100, 8629, 8676, 8677, 8681, 8682
<code>\problem@kw@note</code>	8940	<code>\thistype</code>	3684, 3715
<code>\problem@kw@points</code>	9043, 9567, 9586, 9632, 9652, 9741	<code>\thisvarname</code>	4222, 4371, 6163
<code>\problem@kw@pts</code>	8658, 8751, 9054	<code>\throwaway</code>	136
<code>\problem@kw@solution</code>	8832	<code>\tikzinput</code>	113, 133, 2196, 10005, 10010, 10036
<code>\problem@kw@wrong</code>	9155, 9293		
<code>\problem@restore</code>	8690, 8694, 9835		
<code>\stex@backend</code>	120, 647		
<code>\symrefemph@uri</code>	97, 98, 143, 5810, 5853, 5876, 5893, 8502, 8525		
<code>\varemph@uri</code>	98, 143, 5790, 5901, 5915, 5929		
tex commands:			
<code>\tex_undefined:D</code>	54		
<code>\texname</code>	23		
<code>\text</code>	19		

tikzinput commands: \c_tikzinput_image_bool 38, 9994, 10001 \tiny ... 144, 6051, 8161, 8176, 8501, 8524 \title 69, 111 \title 1569, 8362 titlefragment (env.) 131, 1563 tl commands: \tl_clear:N 282, 393, 505, 543, 1237, 1505, 1753, 1754, 1755, 1794, 1898, 2237, 2612, 2863, 2894, 2932, 3051, 3083, 3233, 3251, 3492, 3647, 3648, 3649, 3687, 3718, 3730, 3731, 3770, 3789, 3846, 4059, 4223, 4360, 4372, 4524, 4533, 4942, 4972, 5016, 5153, 5225, 5370, 5534, 5535, 5536, 5537, 5835, 5836, 5837, 5845, 5988, 6164, 6181, 6192, 6193, 6283, 6309, 6448, 6528, 6539, 6574, 6769, 6841, 7034, 7069, 7070, 7071, 7168, 7335, 7336, 7338, 7339, 7340, 7341, 7649, 7650, 8408, 8557, 9003, 9121, 9123, 9124, 9366, 9769, 9770, 9771, 9846, 9847, 9848, 9873, 9874, 9875, 9876 \tl_const:Nn 1022, 6793, 6794, 9375, 9376 \tl_count:N 5316, 5322 \tl_count:n 5620, 5630, 5636 \tl_gclear:N 2307, 2367 \tl_gput_left:Nn 365, 367, 368 \tl_gput_right:Nn 335, 2469, 2476, 2533, 2565, 2570, 2589, 2590, 2596, 2902, 8394 \tl_gset:Nn 257, 337, 360, 361, 1001, 1414, 1417, 1449, 1456, 1956, 2432, 4468, 4472, 6196, 7774, 7857, 8735, 8738, 9838, 9839 \tl_gset_eq:NN 81, 1049, 1050, 1055, 1056, 2555, 3008 \tl_head:N 4065, 4645, 4876, 4894 \tl_head:n 5621, 5631, 5637, 6727 \tl_if_empty:NTF 298, 557, 1420, 1468, 1477, 1550, 1570, 1808, 1838, 1840, 1870, 1874, 1904, 1906, 1919, 2259, 2270, 2295, 2938, 3055, 3087, 3127, 3256, 3260, 3523, 3758, 3827, 3929, 3932, 3935, 4075, 4282, 4285, 4288, 4296, 4358, 4426, 4429, 4432, 4440, 5193, 5205, 5252, 5336, 5392, 5435, 5451, 5486, 5504, 5663, 5824, 6063, 6067, 6071, 6076, 6175, 6184, 6190, 6387, 6400, 6417, 6434, 6537, 6573, 7018, 7106, 7267, 7316, 7476, 7478, 7481, 7759, 8293, 8415, 8491, 8514, 8621, 8624, 8630, 8676, 8681, 8721, 8724, 8727, 8732, 8832, 8892, 8940, 9029, 9045, 9153, 9155, 9158, 9190, 9192, 9194, 9202, 9291, 9293, 9296, 9324, 9326, 9328, 9336, 9442, 9733, 9789, 9796, 9807, 9812, 9815, 9865, 9892 \tl_if_empty:nTF 265, 274, 362, 886, 1069, 1071, 1075, 1079, 1081, 1097, 1102, 1131, 1203, 1261, 1279, 1372, 1429, 1918, 1965, 2076, 2236, 2664, 2914, 2934, 2946, 3000, 3123, 3183, 3560, 3964, 4005, 4046, 4063, 4680, 4683, 4729, 4739, 4829, 4831, 4915, 4931, 4940, 4993, 5027, 5091, 5112, 5145, 5280, 5300, 5583, 5861, 6021, 6379, 6394, 6407, 6480, 6514, 6619, 6667, 7001, 7122, 7263, 7312, 7326, 8355, 8364, 9422, 9446 \tl_if_empty_p:N 288 \tl_if_eq:NNTF .. 943, 1829, 4645, 4876, 4894, 7038, 9385, 9386, 9395 \tl_if_eq:NnTF 8441, 8636, 8639, 8657, 8662, 9566, 9571, 9585, 9590, 9607 \tl_if_eq:nnTF .. 2447, 3097, 3105, 4591, 5631, 5637, 5692, 5730, 6745 \tl_if_exist:NTF 307, 334, 347, 509, 548, 560, 656, 887, 1011, 1064, 1068, 1078, 1178, 1317, 1492, 1502, 1804, 2119, 2482, 2486, 2490, 2503, 6817, 6822, 7773, 7987, 9550 \tl_if_in:NnTF 2693, 2697, 2715 \tl_item:nn 5623 \tl_map_inline:nn 314, 318 \tl_new:N 1434, 1437, 1959, 2230, 2467, 2562, 2567, 2585, 2586, 2600, 2919, 4464, 4512, 4513 \tl_put_left:Nn 4536, 5404, 6657, 6658, 8603, 9782 \tl_put_right:Nn 1053, 3216, 3220, 3848, 3849, 4518, 4535, 4941, 5039, 5093, 5110, 5111, 5121, 5122, 5135, 5381, 6289, 6295, 6307, 6316, 6592, 6597, 6600, 6689, 6697, 6702, 6710, 6776, 6975, 9390, 9402, 9850, 9851, 9852 \tl_set:Nn 90, 91, 93, 94, 295, 308, 492, 497, 499, 536, 537, 812, 813, 815, 816, 1012, 1148, 1236, 1239, 1290, 1293, 1318, 1333, 1342, 1559, 1706, 1716, 1765, 1825, 1856, 1909, 1938, 1944, 2302, 2335, 2349, 2362, 2424,

2455, 2502, 2767, 2773, 2774, 2775, 2839, 2891, 2943, 3072, 3099, 3107, 3148, 3163, 3276, 3287, 3288, 3289, 3290, 3291, 3297, 3356, 3357, 3381, 3382, 3423, 3424, 3447, 3448, 3502, 3508, 3512, 3682, 3713, 3749, 3787, 3812, 3939, 4081, 4083, 4084, 4085, 4086, 4087, 4121, 4123, 4124, 4125, 4126, 4127, 4182, 4184, 4185, 4186, 4187, 4188, 4247, 4292, 4338, 4339, 4340, 4341, 4342, 4396, 4436, 4467, 4469, 4476, 4488, 4489, 4490, 4509, 4525, 4530, 4531, 4532, 4537, 4613, 4640, 4658, 4694, 4695, 4699, 4702, 4709, 4710, 4746, 4786, 4860, 4866, 4919, 4945, 4948, 4951, 4974, 4975, 4981, 4990, 4995, 5034, 5035, 5039, 5092, 5094, 5102, 5104, 5107, 5113, 5115, 5118, 5130, 5131, 5146, 5147, 5149, 5150, 5316, 5322, 5335, 5341, 5344, 5347, 5369, 5391, 5655, 5656, 5664, 5696, 5698, 5705, 5746, 5748, 5779, 5870, 5871, 5887, 5888, 5902, 5903, 5916, 5917, 5930, 5931, 5980, 6002, 6003, 6145, 6176, 6185, 6209, 6225, 6228, 6237, 6240, 6246, 6253, 6272, 6375, 6377, 6380, 6382, 6576, 6583, 6589, 6628, 6642, 6643, 6655, 6694, 6707, 6719, 6742, 6768, 6790, 6791, 6828, 6829, 6871, 6885, 6990, 7019, 7039, 7190, 7197, 7475, 7752, 7758, 7780, 7956, 7957, 7961, 7962, 7966, 7967, 7970, 7971, 7974, 7975, 7978, 7979, 7982, 7983, 8558, 8630, 8674, 8705, 8706, 8732, 8772, 9073, 9137, 9168, 9173, 9188, 9246, 9280, 9306, 9307, 9322, 9384, 9826, 9832, 9833, 9887, 9888, 9889, 9895, 9959	1101, 1103, 1202, 1204, 1206, 1261, 1267, 1273, 1343, 1371, 1373, 1375, 1376, 1379, 1455, 1784, 1785, 1786, 1787, 1789, 1956, 1986, 2074, 2412, 2413, 2414, 2417, 2418, 2419, 2420, 2460, 2461, 2477, 2575, 2595, 2680, 2694, 2698, 2716, 2732, 2740, 2993, 3115, 3149, 3306, 3474, 3494, 3524, 3957, 3958, 3959, 3960, 3985, 4156, 4652, 4672, 4676, 4837, 4895, 4973, 5046, 5060, 5082, 5101, 5361, 5582, 5597, 6269, 6337, 6363, 6627, 6640, 6729, 6767, 7086, 7775, 7776, 7781, 7782, 7857, 7858, 8539, 9393, 9403, 9425, 9441, 9455, 9457, 9494, 9536
<code>\tl_trim_spaces:N</code> 84	
<code>\l_tmpa_tl</code> 80, 81, 83, 1033, 1183, 1185, 1293, 1307, 1308, 2126, 2431, 2432, 4743, 5102, 5110, 5111, 5113, 5121, 5122, 5303, 5309, 5314, 5316, 5320, 5322, 5325, 5334, 5336, 5341, 5344, 5347, 5750, 6299, 6733, 6769, 6776, 6783, 7475, 7476, 9188, 9202, 9322, 9336, 9384, 9385, 9386, 9395, 9846, 9850, 9859	
<code>\l_tmpp_tl</code> 1290, 1298, 1302, 5303, 5309, 5311, 5325, 5335, 5339, 9847, 9851, 9859	
tmpr commands:	
<code>\l_tmpr_tl</code> 9848, 9852, 9860	
<code>\to</code> 7820, 7821	
<code>\today</code> 8383	
<code>\TODO</code> 4730, 8439	
token commands:	
<code>\c_math_subscript_token</code> 42, 82, 5779, 6559, 6560, 6563, 6564, 6568, 7823, 7825	
<code>\topsep</code> 7537, 8745, 9019, 9038, 9066, 9231, 9625, 9644, 9721	
<code>\trueFfalseT</code> 9360	
<code>\trueTfalseF</code> 9355	
<code>\type</code> 69, 111	
	U
<code>\univ</code> 54	
<code>\unless</code> 8054	
<code>\uppercase</code> 149, 1928, 5881	
<code>\url</code> 1408	
use commands:	
<code>\use:N</code> 375, 506, 510, 512, 544, 549, 551, 558, 561, 563, 574, 881, 1002, 1551, 1553, 1806, 1853, 1935, 2509, 3760, 7452, 7460,	

7509, 7527, 7580, 7590, 7601, 7603, 7632, 7641, 7693, 7706, 7988, 7994	V
\use:n 329, 2065, 2150, 2166, 2181, 3936, 4289, 4433, 4455, 4846, 4961, 5371, 5407	\varbind . . 40, 90, 95, 149, 7220, 7235, 7301 \vardef .. 32, 40, 90, 140, 4195, 7328, 7663 \varemp 98, 143, 5790 \Varname 144, 5897 \varname 144, 5897 \varnotation 90, 145, 6152 \varref 144, 5897 \varseq 37, 90, 141, 4345 \vbox 136, 1851, 1932, 7455, 7592, 7754, 8155, 8250, 8265, 8814, 8874, 8922, 8973, 9139, 9170
\use:nn 93, 304, 474, 477, 874, 881, 1553, 1854, 1936, 2458, 2942, 3059, 3157, 3721, 3754, 3822, 3997, 4226, 4610, 4633, 4637, 4913, 4987, 5083, 5417, 5607, 5658, 5741, 5892, 5965, 6147, 6167, 6500, 6530, 6875, 6981, 6982, 7183, 7294, 7328, 7663, 8419, 8774, 9533, 9828	vbox commands: \vbox_set:Nn 2643 \vdash 7835 \vfill 7992, 9465, 9473, 9764 \vM 45 \vMs 46 \vop 38, 40 \vphantom 9457 \vrule 7754, 8312, 9139, 9170 \vskip 7754, 8311, 8313, 9139, 9170 \vspace 9467
\use_i:nn 3059, 4961, 9851 \use_i:nnn 867, 1209 \use_i:nnnn 1385 \use_i:nnnnn 1107 \use_ii:nn 3047, 3065, 5004 \use_ii:nnn 870, 874, 1212, 1215 \use_ii:nnnn 1388 \use_ii:nnnnn 1110 \use_iii:nnn 877, 881, 1218, 1222 \use_iii:nnnnn 1113 \use_iv:nnnn 1401, 1404 \use_iv:nnnnn 1116 \use_none:nn 6992 \use_v:nnnnn 1119	W \wff 19 \withbrackets 87, 147, 6826, 6837
\usebox 8244 \usemodule . 16, 21, 22, 25, 34, 41, 88, 89, 91, 116, 120, 137, 416, 423, 2851 \usepackage 7, 22, 2065, 8068 \useSGvar 104, 8432 \usestructure 91, 148, 7046 \usetheme 8068 \usetikzlibrary 75, 132, 2073, 10004	X \xdef ... 9512, 9564, 9582, 9629, 9648, 9762 \xspace 131, 1492, 1502, 3804, 3805, 4547, 4548
	Y \yesFnoT 9350 \yesTnoF 9345 \yield 150, 7447, 7744