

The sTeX3 Package Collection *

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

2022-06-30

Abstract

sTeX is a collection of L^AT_EX packages that allow to markup documents semantically without leaving the document format.

Running ‘pdflatex’ over sTeX-annotated documents formats them into normal-looking PDF. But sTeX also comes with a conversion pipeline into semantically annotated HTML5, which can host semantic added-value services that make the documents active (i.e. interactive and user-adaptive) and essentially turning L^AT_EX into a document format for (mathematical) knowledge management (MKM).

sTeX augments L^AT_EX with

- *semantic macros* that denote and distinguish between mathematical concepts, operators, etc. independent of their notational presentation,
- a powerful *module system* that allows for authoring and importing individual fragments containing document text and/or semantic macros, independent of – and without hard coding – directory paths relative to the current document, and
- a mechanism for exporting sTeX documents to (modular) XHTML, preserving all the semantic information for semantically informed knowledge management services.

This is the full documentation of sTeX. It consists of four parts:

- **Part I** is a general manual for the sTeX package and associated software. It is primarily directed at end-users who want to use sTeX to author semantically enriched documents.
- **Part II** documents the macros provided by the sTeX package. It is primarily directed at package authors who want to build on sTeX, but can also serve as a reference manual for end-users.
- **Part III** documents additional packages that build on sTeX, primarily its module system. These are not part of the sTeX package itself, but useful additions enabled by sTeX package functionality.
- **Part IV** is the detailed documentation of the sTeX package implementation.

*Version 3.1 (last revised 2022-06-30)

Contents

I	Manual	1
1	What is sTeX?	2
2	Quickstart	3
2.1	Setup	3
2.1.1	Minimal Setup for the PDF-only Workflow	3
2.1.2	GIT-based Setup for the sTeX Development Version	3
2.1.3	sTeX Archives (Manual Setup)	4
2.1.4	The sTeX IDE	4
2.1.5	Manual Setup for Active Documents and Knowledge Management Services	4
2.2	A First sTeX Document	5
2.2.1	OMDoc/xhtml Conversion	8
2.2.2	MMT/OMDoc Conversion	9
3	Creating sTeX Content	10
3.1	How Knowledge is Organized in sTeX	10
3.2	sTeX Archives	11
3.2.1	The Local MathHub-Directory	11
3.2.2	The Structure of sTeX Archives	12
3.2.3	MANIFEST.MF-Files	12
3.2.4	Using Files in sTeX Archives Directly	13
3.3	Module, Symbol and Notation Declarations	14
3.3.1	The smodule-Environment	14
3.3.2	Declaring New Symbols and Notations	16
3.3.3	Operator Notations	19
3.3.3	Argument Modes	20
3.3.3	Mode-b Arguments	20
3.3.3	Mode-a Arguments	21
3.3.3	Mode-B Arguments	22
3.3.4	Type and Definiens Components	23
3.3.5	Precedences and Automated Bracketing	24
3.3.6	Variables	26
3.3.7	Variable Sequences	27
3.4	Module Inheritance and Structures	29
3.4.1	Multilinguality and Translations	29
3.4.2	Simple Inheritance and Namespaces	30
3.4.3	The mathstructure Environment	31
3.4.4	The copymodule Environment	34
3.4.5	The interpretmodule Environment	36
3.5	Primitive Symbols (The sTeX Metatheory)	36
4	Using sTeX Symbols	37
4.1	\symref and its variants	37
4.2	Marking Up Text and On-the-Fly Notations	38
4.3	Referencing Symbols and Statements	40

5	<code>sTeX</code> Statements	41
5.1	Definitions, Theorems, Examples, Paragraphs	41
5.2	Proofs	44
5.3	Highlighting and Presentation Customizations	47
6	Additional Packages	49
6.1	Tikzinput: Treating TIKZ code as images	49
6.2	Modular Document Structuring	50
6.2.1	Introduction	50
6.2.2	Package Options	50
6.2.3	Document Fragments	50
6.2.4	Ending Documents Prematurely	52
6.2.5	Global Document Variables	52
6.3	Slides and Course Notes	52
6.3.1	Introduction	52
6.3.2	Package Options	53
6.3.3	Notes and Slides	53
6.3.4	Customizing Header and Footer Lines	54
6.3.5	Frame Images	55
6.3.6	Excursions	56
6.4	Representing Problems and Solutions	57
6.4.1	Introduction	57
6.4.2	Problems and Solutions	57
6.4.3	Multiple Choice Blocks	59
6.4.4	Including Problems	60
6.5	Homeworks, Quizzes and Exams	60
6.5.1	Introduction	60
6.5.2	Package Options	61
6.5.3	Assignments	61
6.5.4	Including Assignments	61
6.5.5	Typesetting Exams	61
II	Documentation	63
7	<code>sTeX</code>-Basics	64
7.1	Macros and Environments	64
7.1.1	HTML Annotations	64
7.1.2	Babel Languages	65
7.1.3	Auxiliary Methods	65
8	<code>sTeX</code>-MathHub	66
8.1	Macros and Environments	66
8.1.1	Files, Paths, URIs	66
8.1.2	MathHub Archives	67
8.1.3	Using Content in Archives	68

9	sTeX-References	69
9.1	Macros and Environments	69
9.1.1	Setting Reference Targets	69
9.1.2	Using References	70
10	sTeX-Modules	71
10.1	Macros and Environments	71
10.1.1	The <code>smodule</code> environment	73
11	sTeX-Module Inheritance	75
11.1	Macros and Environments	75
11.1.1	SMS Mode	75
11.1.2	Imports and Inheritance	76
12	sTeX-Symbols	78
12.1	Macros and Environments	78
13	sTeX-Terms	80
13.1	Macros and Environments	80
14	sTeX-Structural Features	82
14.1	Macros and Environments	82
14.1.1	Structures	82
15	sTeX-Statements	83
15.1	Macros and Environments	83
16	sTeX-Proofs: Structural Markup for Proofs	84
17	sTeX-Metatheory	85
17.1	Symbols	85
III	Extensions	86
18	Tikzinput: Treating TIKZ code as images	87
18.1	Macros and Environments	87
19	document-structure: Semantic Markup for Open Mathematical Documents in L^AT_EX	88
20	NotesSlides – Slides and Course Notes	89
21	problem.sty: An Infrastructure for formatting Problems	90
22	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	91
IV	Implementation	92

23	STeX-Basics Implementation	93
23.1	The STeXDocument Class	93
23.2	Preliminaries	94
23.3	Messages and logging	94
23.4	HTML Annotations	95
23.5	Babel Languages	97
23.6	Persistence	98
23.7	Auxiliary Methods	99
24	STeX-MathHub Implementation	102
24.1	Generic Path Handling	102
24.2	PWD and kpsewhich	104
24.3	File Hooks and Tracking	105
24.4	MathHub Repositories	106
24.5	Using Content in Archives	111
25	STeX-References Implementation	115
25.1	Document URIs and URLs	115
25.2	Setting Reference Targets	117
25.3	Using References	119
26	STeX-Modules Implementation	122
26.1	The smodule environment	126
26.2	Invoking modules	132
27	STeX-Module Inheritance Implementation	134
27.1	SMS Mode	134
27.2	Inheritance	138
28	STeX-Symbols Implementation	144
28.1	Symbol Declarations	144
28.2	Notations	152
28.3	Variables	160
29	STeX-Terms Implementation	168
29.1	Symbol Invocations	168
29.2	Terms	175
29.3	Notation Components	179
29.4	Variables	181
29.5	Sequences	184
30	STeX-Structural Features Implementation	185
30.1	Imports with modification	186
30.2	The feature environment	194
30.3	Structure	194
31	STeX-Statements Implementation	205
31.1	Definitions	205
31.2	Assertions	211
31.3	Examples	214
31.4	Logical Paragraphs	217

32 The Implementation	222
32.1 Proofs	222
32.2 Justifications	233
33 \TeX-Others Implementation	234
34 \TeX-Metatheory Implementation	236
35 Tikzinput Implementation	239
36 document-structure.sty Implementation	242
36.1 Package Options	242
36.2 Document Structure	243
36.3 Front and Backmatter	247
36.4 Global Variables	249
37 NotesSlides – Implementation	250
37.1 Class and Package Options	250
37.2 Notes and Slides	252
37.3 Header and Footer Lines	256
37.4 Frame Images	258
37.5 Sectioning	259
37.6 Excursions	262
38 The Implementation	264
38.1 Package Options	264
38.2 Problems and Solutions	265
38.3 Multiple Choice Blocks	272
38.4 Including Problems	273
38.5 Reporting Metadata	274
39 Implementation: The hwexam Package	277
39.1 Package Options	277
39.2 Assignments	278
39.3 Including Assignments	281
39.4 Typesetting Exams	282
39.5 Leftovers	284
40 References	285

Part I

Manual



Boxes like this one contain implementation details that are mostly relevant for more advanced use cases, might be useful to know when debugging, or might be good to know to better understand how something works. They can easily be skipped on a first read.



Boxes like this one explain how some \LaTeX concept relates to the MMT/OMDoc system, philosophy or language; see [MMT; Koh06] for introductions.

Chapter 1

What is sTeX?

Formal systems for mathematics (such as interactive theorem provers) have the potential to significantly increase both the accessibility of published knowledge, as well as the confidence in its veracity, by rendering the precise semantics of statements machine actionable. This allows for a plurality of added-value services, from semantic search up to verification and automated theorem proving. Unfortunately, their usefulness is hidden behind severe barriers to accessibility; primarily related to their surface languages reminiscent of programming languages and very unlike informal standards of presentation.

sTeX minimizes this gap between informal and formal mathematics by integrating formal methods into established and widespread authoring workflows, primarily L^AT_EX, via non-intrusive semantic annotations of arbitrary informal document fragments. That way formal knowledge management services become available for informal documents, accessible via an IDE for authors and via generated *active* documents for readers, while remaining fully compatible with existing authoring workflows and publishing systems.

Additionally, an extensible library of reusable document fragments is being developed, that serve as reference targets for global disambiguation, intermediaries for content exchange between systems and other services.

Every component of the system is designed modularly and extensibly, and thus lay the groundwork for a potential full integration of interactive theorem proving systems into established informal document authoring workflows.

The general sTeX workflow combines functionalities provided by several pieces of software:

- The sTeX package collection to use semantic annotations in L^AT_EX documents,
- RuS_{TeX} [RT] to convert `tex` sources to (semantically enriched) `xhtml`,
- The MMT system [MMT], that extracts semantic information from the thus generated `xhtml` and provides semantically informed added value services. Notably, MMT integrates the RuS_{TeX} system already.

Chapter 2

Quickstart

2.1 Setup

There are two ways of using $\text{\texttt{sTeX}}$: as a

1. way of writing $\text{\texttt{LATeX}}$ more modularly (object-oriented Math) for creating PDF documents or
2. foundation for authoring active documents in HTML5 instrumented with knowledge management services.

Both are legitimate and useful. The first requires a significantly smaller tool-chain, so we describe it first. The second requires a much more substantial (and experimental) toolchain of knowledge management systems. Both workflows profit from an integrated development environment (IDE), which (also) automates setup as far as possible (see [subsection 2.1.4](#)).

2.1.1 Minimal Setup for the PDF-only Workflow

In the best of all worlds, there is no setup, as you already have a new version of $\text{\texttt{TeXLive}}$ on your system as a $\text{\texttt{LATeX}}$ enthusiast. If not now is the time to install it; see [\[TL\]](#). You can usually update $\text{\texttt{TeXLive}}$ via a package manager or the $\text{\texttt{TeXLive}}$ manager **tlmgr**.

Alternatively, you can install $\text{\texttt{sTeX}}$ from CTAN, the Comprehensive $\text{\texttt{TeX}}$ Archive Network; see [\[ST\]](#) for details.

2.1.2 GIT-based Setup for the $\text{\texttt{sTeX}}$ Development Version

If you want use the latest and greatest $\text{\texttt{sTeX}}$ packages that have not even been released to CTAN, then you can directly clone them from the $\text{\texttt{sTeX}}$ development repository [\[sTeX\]](#) by the following command-line instructions:

```
cd <stexdir>
git clone https://github.com/slatex/sTeX.git
```

and keep it updated by pulling updates via `git pull` in the cloned $\text{\texttt{sTeX}}$ directory. Then update your `TEXINPUTS` environment variable, e.g. by placing the following line in your `.bashrc`:

```
export TEXINPUTS="$(TEXINPUTS):<sTeXDIR>//:"
```

2.1.3 $\text{\texttt{sTeX}}$ Archives (Manual Setup)

Writing semantically annotated $\text{\texttt{sTeX}}$ becomes much easier, if we can use well-designed libraries of already annotated content. $\text{\texttt{sTeX}}$ provides such libraries as $\text{\texttt{sTeX}}$ archives – i.e. GIT repositories at <https://gl.mathhub.info> – most prominently the SMGLoM libraries at <https://gl.mathhub.info/smgloom>.

To do so, we set up a **local MathHub** by creating a MathHub directory `<mhdir>`. Every $\text{\texttt{sTeX}}$ archive as an **archive path** `<apath>` and a name `<archive>`. We can clone the $\text{\texttt{sTeX}}$ archive by the following command-line instructions:

```
cd <mhdir>/<apath>
git clone https://gl.mathhub.info/smgloom/<archive>.git
```

Note that $\text{\texttt{sTeX}}$ archives often depend on other archives, thus you should be prepared to clone these as well – e.g. if `pdflatex` reports missing files. To make sure that $\text{\texttt{sTeX}}$ too knows where to find its archives, we need to set a global system variable `MATHHUB`, that points to your local MathHub-directory (see [section 3.2](#)).

```
export MATHHUB="<mhdir>"
```

2.1.4 The $\text{\texttt{sTeX}}$ IDE

We are currently working on an $\text{\texttt{sTeX}}$ IDE as an $\text{\texttt{sTeX}}$ plugin for VScode; see [\[S1a\]](#). It will feature a setup procedure that automates the setup described above (and below). For additional functionality see the (now obsolete) plugin for $\text{\texttt{sTeX}}$ 1 [\[SLS; Stb\]](#).

2.1.5 Manual Setup for Active Documents and Knowledge Management Services

Foregoing on the $\text{\texttt{sTeX}}$ IDE, we will need several additional (on top of the minimal setup above) pieces of software; namely:

- **The Mmt System** available [here](#). We recommend following the setup routine documented [here](#).

Following the setup routine (Step 3) will entail designating a MathHub-directory on your local file system, where the MMT system will look for $\text{\texttt{sTeX}}$ /MMT content archives.

- **$\text{\texttt{sTeX}}$ Archives** If we only care about $\text{\texttt{LATEX}}$ and generating `pdfs`, we do not technically need MMT at all; however, we still need the `MATHHUB` system variable to be set. Furthermore, MMT can make downloading content archives we might want to use significantly easier, since it makes sure that all dependencies of (often highly interrelated) $\text{\texttt{sTeX}}$ archives are cloned as well.

Once set up, we can run `mmt` in a shell and download an archive along with all of its dependencies like this: `lmh install <name-of-repository>`, or a whole *group* of archives; for example, `lmh install smgloom` will download all `smgloom` archives.

- **$\text{\texttt{RUSTeX}}$** The MMT system will also set up $\text{\texttt{RUSTeX}}$ for you, which is used to generate (semantically annotated) `xhtml` from `tex` sources. In lieu of using MMT, you can also download and use $\text{\texttt{RUSTeX}}$ directly [here](#).

2.2 A First \TeX Document

Having set everything up, we can write a first \TeX document. As an example, we will use the `smglom/calculus` and `smglom/arithmetics` archives, which should be present in the designated MathHub-folder, and write a small fragment defining the *geometric series*:

```

1 \documentclass{article}
2 \usepackage{stex,xcolor,stexthm}
3
4 \begin{document}
5 \begin{smodule}{GeometricSeries}
6   \importmodule[smglom/calculus]{series}
7   \importmodule[smglom/arithmetics]{realarith}
8
9   \symdef{geometricSeries}[name=geometric-series]{\comp{S}}
10
11   \begin{sdefinition}[for=geometricSeries]
12     The \definame{geometricSeries} is the \symname{?series}
13     \[\defeq{\geometricSeries}{\definiens{
14       \infinitesum{\svar{n}}{1}{
15         \realdive[frac]{1}{
16           \realpower{2}{\svar{n}}
17         }
18       }}
19     \].\]
20   \end{sdefinition}
21
22   \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
23     The \symname{geometricSeries} \symname{converges} towards $1$.
24   \end{sassertion}
25 \end{smodule}
26 \end{document}

```

Compiling this document with `pdflatex` should yield the output

Definition 0.1. The **geometric series** is the **series**

$$S := \sum_{n=1}^{\infty} \frac{1}{2^n}.$$

Theorem 0.2. The **geometric series converges** towards 1.

Move your cursor over the various highlighted parts of the document – depending on your pdf viewer, this should yield some interesting (but possibly for now cryptic) information.

Remark 2.2.1:

Note that all of the highlighting, tooltips, coloring and the environment headers come from `stexthm` – by default, the amount of additional packages loaded is kept to a minimum and all the presentations can be customized, see [section 5.3](#).

Let’s investigate this document in detail to understand the respective parts of the \TeX markup infrastructure:

```
smodule (env.) \begin{smodule}{GeometricSeries}
...
\end{smodule}
```

First, we open a new *module* called `GeometricSeries`. The main purpose of the `smodule` environment is to group the contents and associate it with a *globally unique* identifier (URI), which is computed from the name `GeometricSeries` and the document context.

(Depending on your pdf viewer), the URI should pop up in a tooltip if you hover over the word [geometric series](#).

```
\importmodule \importmodule[smglom/calculus]{series}
\importmodule \importmodule[smglom/arithmetics]{realarith}
```

Next, we *import* two modules – `series` from the \TeX archive `smglom/calculus`, and `realarith` from the \TeX archive `smglom/arithmetics`. If we investigate these archives, we find the files `series.en.tex` and `realarith.en.tex` (respectively) in their respective source-folders, which contain the statements `\begin{smodule}{series}` and `\begin{smodule}{realarith}` (respectively).

The `\importmodule`-statements make all \TeX symbols and associated semantic macros (e.g. `\infinitesum`, `\realdive`, `\realpower`) in the imported module available to the current module `GeometricSeries`. The module `GeometricSeries` “exports” all of these symbols to all modules imports it via an `\importmodule{GeometricSeries}` instruction. Additionally it exports the local symbol `\geometricSeries`.

```
\usemodule
```

If we only want to *use* the content of some module `Foo`, e.g. in remarks or examples, but none of the symbols in our current module actually *depend* on the content of `Foo`, we can use `\usemodule` instead – like `\importmodule`, this will make the module content available, but will *not* export it to other modules.

```
\symdef \symdef{GeometricSeries}[name=geometric-series]{\comp{S}}
```

Next, we introduce a new *symbol* with name `geometric-series` and assign it the semantic macro `\geometricSeries`. `\symdef` also immediately assigns this symbol a *notation*, namely `S`.

```
\comp
```

The macro `\comp` marks the `S` in the notation as a *notational component*, as opposed to e.g. arguments to `\geometricSeries`. It is the notational components that get highlighted and associated with the corresponding symbol (i.e. in this case `geometricSeries`). Since `\geometricSeries` takes no arguments, we can wrap the whole notation in a `\comp`.

```

\begin{sdefinition}[for=geometricSeries]
...
\end{sdefinition}
\begin{sassertion}[name=geometricSeriesConverges,type=theorem]
...
\end{sassertion}

```

What follows are two $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ -*statements* (e.g. definitions, theorems, examples, proofs, ...). These are semantically marked-up variants of the usual environments, which take additional optional arguments (e.g. `for=`, `type=`, `name=`). Since many $\text{\texttt{L}}\text{\texttt{A}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ templates predefine environments like `definition` or `theorem` with different syntax, we use `sdefinition`, `sassertion`, `sexample` etc. instead. You can customize these environments to e.g. simply wrap around some predefined `theorem`-environment. That way, we can still use `sassertion` to provide semantic information, while being fully compatible with (and using the document presentation of) predefined environments.

In our case, the `stexthm`-package patches e.g. `\begin{sassertion}[type=theorem]` to use a `theorem`-environment defined (as usual) using the `amsthm` package.

`\symname` ... is the `\symname{?series}`

The `\symname`-command prints the name of a symbol, highlights it (based on customizable settings) and associates the text printed with the corresponding symbol.

Note that the argument of `\symref` can be an imported symbol (here the `series` symbol is imported from the `series` module). $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ tries to determine the full symbol URI from the argument. If there are name clashes in or with the imported symbols, the name of the exporting module can be prepended to the symbol name before the `?` character.

If you hover over the word `series` in the pdf output, you should see a tooltip showing the full URI of the symbol used.

`\symref` The `\symname`-command is a special case of the more general `\symref`-command, which allows customizing the precise text associated with a symbol. `\symref` takes two arguments: the first is the symbol name (or macro name), and the second a variant verbalization of the symbol, e.g. an inflection variant, a different language or a synonym. In our example `\symname{?series}` abbreviates `\symref{?series}{series}`.

`\define` The `\define{geometricSeries}` ...

`\definiendum` The `sdefinition`-environment provides two additional macros, `\define` and `\definiendum` which behave similarly to `\symname` and `\symref`, but explicitly mark the symbols as *being defined* in this environment, to allow for special highlighting.

```

\[\defeq{\geometricSeries}{\definiens{
  \infinitesum{\svar{n}}{1}{
    \realdivide[frac]{1}{
      \realpower{2}{\svar{n}}
    }
  }}
}\].\]

```

The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) `series` and `realarithmetics`, such as `\defeq`, `\infinitesum`,

etc. In math mode, using a semantic macro inserts its (default) definition. A semantic macro can have several notations – in that case, we can explicitly choose a specific notation by providing its identifier as an optional argument; e.g. `\realdivide[frac]{a}{b}` will use the explicit notation named `frac` of the semantic macro `\realdivide`, which yields $\frac{a}{b}$ instead of a/b .

`\svar` The `\svar{n}` command marks up the `n` as a variable with name `n` and notation `n`.

`\definiens` The `sdefinition`-environment additionally provides the `\definiens`-command, which allows for explicitly marking up its argument as the *definiens* of the symbol currently being defined.

2.2.1 OMDoc/xhtml Conversion

So, if we run `pdflatex` on our document, then $\text{\texttt{STEX}}$ yields pretty colors and tooltips¹. But $\text{\texttt{STEX}}$ becomes a lot more powerful if we additionally convert our document to `xhtml` while preserving all the $\text{\texttt{STEX}}$ markup in the result.

TODO VSCode Plugin

Using `RuSTEX [RT]`, we can convert the document to `xhtml` using the command `rustex -i /path/to/file.tex -o /path/to/outfile.xhtml`. Investigating the resulting file, we notice additional semantic information resulting from our usage of semantic macros, `\symref` etc. Below is the (abbreviated) snippet inside our `\definiens` block:

```
<mrow resource="" property="stex:definiens">
  <mrow resource="...?series?infinitesum" property="stex:OMBIND">
    <munderover displaystyle="true">
      <mo resource="...?series?infinitesum" property="stex:comp"> $\Sigma$ </mo>
      <mrow>
        <mrow resource="1" property="stex:arg">
          <mi resource="var://n" property="stex:OMV">n</mi>
        </mrow>
        <mo resource="...?series?infinitesum" property="stex:comp">=</mo>
        <mi resource="2" property="stex:arg">1</mi>
      </mrow>
      <mi resource="...?series?infinitesum" property="stex:comp"> $\infty$ </mi>
    </munderover>
    <mrow resource="3" property="stex:arg">
      <mfrac resource="...?realarith?division#frac#" property="stex:OMA">
        <mi resource="1" property="stex:arg">1</mi>
        <mrow resource="2" property="stex:arg">
          <msup resource="...realarith?exponentiation" property="stex:OMA">
            <mi resource="1" property="stex:arg">2</mi>
            <mrow resource="2" property="stex:arg">
              <mi resource="var://n" property="stex:OMV">n</mi>
            </mrow>
          </msup>
        </mrow>
      </mfrac>
    </mrow>
  </mrow>
</mrow>
```

¹...and hyperlinks for symbols, and indices, and allows reusing document fragments modularly, and...

...containing all the semantic information. The MMT system can extract from this the following OPENMATH snippet:

```
<OMBIND>
  <OMID name="...?series?infinitesum"/>
  <OMV name="n"/>
  <OMLIT name="1"/>
  <OMA>
    <OMS name="...?realarith?division"/>
    <OMLIT name="1"/>
    <OMA>
      <OMS name="...realarith?exponentiation"/>
      <OMLIT name="2"/>
      <OMV name="n"/>
    </OMA>
  </OMA>
</OMBIND>
```

...giving us the full semantics of the snippet, allowing for a plurality of knowledge management services – in particular when serving the `xhtml`.

Remark 2.2.2:

Note that the `html` when opened in a browser will look slightly different than the `pdf` when it comes to highlighting semantic content – that is because naturally `html` allows for much more powerful features than `pdf` does. Consequently, the `html` is intended to be served by a system like MMT, which can pick up on the semantic information and offer much more powerful highlighting, linking and similar features, and being customizable by *readers* rather than being prescribed by an author.

Additionally, not all browsers (most notably Chrome) support MATHML natively, and might require additional external JavaScript libraries such as MathJax to render mathematical formulas properly.

2.2.2 Mmt/OMDoc Conversion

Another way to convert our document to *actual* MMT/OMDOC is to put it in an `TEX` archive (see [section 3.2](#)) and have MMT take care of everything.

Assuming the above file is `source/demo.tex` in an `TEX` archive `MyTest`, you can run MMT and do `build MyTest stex-omdoc demo.tex` to convert the document to both `xhtml` (which you will find in `xhtml/demo.xhtml` in the archive) and formal MMT/OMDOC, which you can subsequently view in the MMT browser (see <https://uniformal.github.io/doc/applications/server.html#the-mmt-web-site> for details).

Chapter 3

Creating sTeX Content

We can use sTeX by simply including the package with `\usepackage{stex}`, or – primarily for individual fragments to be included in other documents – by using the sTeX document class with `\documentclass{stex}` which combines the standalone document class with the stex package.

Both the stex package and document class offer the following options:

lang (*(⟨language⟩*)*) Languages to load with the babel package.

mathhub (*(⟨directory⟩)*) MathHub folder to search for repositories – this is not necessary if the MATHHUB system variable is set.

writesms (*(⟨boolean⟩)*) with this package option, sTeX will write the contents of all external modules imported via `\importmodule` or `\usemodule` into a file `\jobname.sms` (analogously to the table of contents `.toc`-file).

usesms (*(⟨boolean⟩)*) subsequently tells sTeX to read the generated sms-file at the beginning of the document. This allows for e.g. collaborating on documents without all authors having to have all used archives and modules available – one author can load the modules with **writesms**, and the rest can use the modules with **usesms**. Furthermore, the sms file can be submitted alongside a `tex`-file, effectively making it “standalone”.

image (*(⟨boolean⟩)*) passed on to tikzinput.

debug (*(⟨log-prefix⟩*)*) Logs debugging information with the given prefixes to the terminal, or all if **all** is given. Largely irrelevant for the majority of users.

3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

1. sTeX **archives** (see [section 3.2](#)) contain individual `.tex`-files.
2. These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.

3. Modules contain $\text{\S}\text{\TeX}$ **symbol declarations**, introduced via $\text{\textbackslash symdecl}\{\text{symbolname}\}$, $\text{\textbackslash symdef}\{\text{symbolname}\}$ and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro* $\text{\textbackslash symbolname}$ generated by symbol declarations.
4. $\text{\S}\text{\TeX}$ **expressions** finally are built up from usages of semantic macros.

- $\text{\S}\text{\TeX}$ archives are simultaneously MMT archives, and the same directory structure is consequently used.
 - $\text{\S}\text{\TeX}$ modules correspond to OMDOC/MMT *theories*. $\text{\textbackslash importmodules}$ (and similar constructions) induce MMT **includes** and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDOC sense [RK13].
 - Symbol declarations induce OMDOC/MMT *constants*, with optional (formal) *type* and *definiens* components.
 - Finally, $\text{\S}\text{\TeX}$ expressions are converted to OMDOC/MMT terms, which use the abstract syntax (and XML encoding) of OPENMATH [Bus+04].

3.2 $\text{\S}\text{\TeX}$ Archives

3.2.1 The Local MathHub-Directory

$\text{\textbackslash usemodule}$, $\text{\textbackslash importmodule}$, $\text{\textbackslash inputref}$ etc. allow for including content modularly without having to specify absolute paths, which would differ between users and machines. Instead, $\text{\S}\text{\TeX}$ uses *archives* that determine the global namespaces for symbols and statements and make it possible for $\text{\S}\text{\TeX}$ to find content referenced via such URIs.

All $\text{\S}\text{\TeX}$ archives need to exist in the local MathHub-directory. $\text{\S}\text{\TeX}$ knows where this folder is via one of four means:

1. If the $\text{\S}\text{\TeX}$ package is loaded with the option $\text{mathhub}=\text{/path/to/mathhub}$, then $\text{\S}\text{\TeX}$ will consider /path/to/mathhub as the local MathHub-directory.
2. If the mathhub package option is *not* set, but the macro $\text{\textbackslash mathhub}$ exists when the $\text{\S}\text{\TeX}$ -package is loaded, then this macro is assumed to point to the local MathHub-directory; i.e. $\text{\textbackslash def}\text{\textbackslash mathhub}\{\text{/path/to/mathhub}\}\text{\textbackslash usepackage}\{\text{stex}\}$ will set the MathHub-directory as path/to/mathhub .
3. Otherwise, $\text{\S}\text{\TeX}$ 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. Finally, if all else fails, $\text{\S}\text{\TeX}$ will look for a file $\text{\textasciitilde}/\text{.stex/mathhub.path}$. If this file exists, $\text{\S}\text{\TeX}$ 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.

3.2.2 The Structure of \TeX Archives

An \TeX archive `group/name` is stored in the directory `/path/to/mathhub/group/name`; e.g. assuming your local MathHub-directory is set as `/user/foo/MathHub`, then in order for the `smglom/calculus`-archive to be found by the \TeX system, it needs to be in `/user/foo/MathHub/smgglom/calculus`.

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 where \TeX will look for files included via `\libinput`.

Additionally a *group* of archives `group/name` may have an additional archive `group/meta-inf`. If this `meta-inf`-archive has a `/lib`-subdirectory, it too will be searched by `\libinput` from all tex files in any archive in the `group/*-group`.

We recommend the following additional directory structure in the `source`-folder of an \TeX archive:

- `/source/mod/` – individual \TeX modules, containing symbol declarations, notations, and `\begin{spargraph}[type=symdoc,for=...]` environments for “encyclopaedic” symbol documentations
- `/source/def/` – definitions
- `/source/ex/` – examples
- `/source/thm/` – theorems, lemmata and proofs; preferably proofs in separate files to allow for multiple proofs for the same statement
- `/source/snip/` – individual text snippets such as remarks, explanations etc.
- `/source/frag/` – individual document fragments, ideally only `\inputrefing` snippets, definitions, examples etc. in some desirable order
- `/source/tikz/` – tikz images, as individual `.tex`-files
- `/source/PIC/` – image files.

3.2.3 MANIFEST.MF-Files

The `MANIFEST.MF` in the `META-INF`-directory consists of key-value-pairs, informing \TeX (and associated software) of various properties of an archive. For example, the `MANIFEST.MF` of the `smglom/calculus`-archive looks like this:

```
id: smglom/calculus
source-base: http://mathhub.info/smgglom/calculus
narration-base: http://mathhub.info/smgglom/calculus
dependencies: smglom/arithmetic,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
```

teaser: Terminology for the mathematical study of change. description: desc.html

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

id: The name of the archive, including its group (e.g. `smglom/calculus`),

source-base or

ns: The namespace from which all symbol and module URIs in this repository are formed, see (TODO),

narration-base: The namespace from which all document URIs in this repository are formed, see (TODO),

url-base: The URL that is formed as a basis for *external references*, see (TODO),

dependencies: All archives that this archive depends on. \TeX ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

3.2.4 Using Files in \TeX Archives Directly

Several macros provided by \TeX allow for directly including files in repositories. These are:

`\mhinput` `\mhinput`[Some/Archive]{some/file} directly inputs the file `some/file` in the `source-` folder of `Some/Archive`.

`\inputref` `\inputref`[Some/Archive]{some/file} behaves like `\mhinput`, but wraps the input in a `\begingroup ... \endgroup`. When converting to `xhtml`, the file is not input at all, and instead an `html`-annotation is inserted that references the file, e.g. for lazy loading.

In the majority of practical cases `\inputref` is likely to be preferred over `\mhinput` because it leads to less duplication in the generated `xhtml`.

`\ifinput` Both `\mhinput` and `\inputref` set `\ifinput` to “true” during input. This allows for selectively including e.g. bibliographies only if the current file is not being currently included in a larger document.

`\addmhbibresource` `\addmhbibresource`[Some/Archive]{some/file} searches for a file like `\mhinput` does, but calls `\addbibresource` to the result and looks for the file in the archive root directory directly, rather than the `source` directory. Typical invocations are

- `\addmhbibresource{lib/refs.bib}`, which specifies a bibliography in the `lib` folder in the local archive or
- `\addmhbibresource[HW/meta-inf]{lib/refs.bib}` in another.

`\libinput` `\libinput{some/file}` searches for a file `some/file` 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 include all found files in reverse order; e.g. `\libinput{preamble}` in a `.tex`-file in `smglom/calculus` will *first* input `.../smglom/meta-inf/lib/preamble.tex` and then `.../smglom/calculus/lib/preamble.tex`.

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

`\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` is found.

Remark 3.2.1:

A good practice is to have individual \TeX fragments follow basically this document frame:

```
1 \documentclass{stex}
2 \libinput{preamble}
3 \begin{document}
4   ...
5   \ifinputref \else \libinput{postamble} \fi
6 \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 `postamble.tex` can e.g. print the bibliography, index etc.

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

`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` ($\langle token\ list \rangle$) to display in customizations.
`type` ($\langle string \rangle *$) for use in customizations.
`deprecate` ($\langle module \rangle$) if set, will throw a warning when loaded, urging to use $\langle module \rangle$ instead.
`id` ($\langle string \rangle$) for cross-referencing.
`ns` ($\langle URI \rangle$) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed using `\stex_modules_current_namespace:`.
`lang` ($\langle language \rangle$) if not set, computed from the current file name (e.g. `foo.en.tex`).
`sig` ($\langle language \rangle$) 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.
`creators` ($\langle string \rangle *$) names of the creators.
`contributors` ($\langle string \rangle *$) names of contributors.
`srccite` ($\langle string \rangle$) a source citation for the content of this module.

\hookrightarrow An \LaTeX module corresponds to an MMT/OMDOC *theory*. As such it
 \hookrightarrow gets assigned a module URI (*universal resource identifier*) of the form
 \hookrightarrow `<namespace>?<module-name>`.

By default, opening a module will produce no output whatsoever, e.g.:

Example 1

Input:

```

1 \begin{smodule}[title={This is Some Module}]{SomeModule}
2   Hello World
3 \end{smodule}
  
```

Output:

Hello World

`\stexpatchmodule` We can customize this behavior either for all modules or only for modules with a specific `type` using the command `\stexpatchmodule[optional-type]{begin-code}{end-code}`. Some optional parameters are then available in `\smodule*`-macros, specifically `\smoduletitle`, `\smoduletype` and `\smoduleid`.

For example:

Example 2

Input:

```

1 \stexpatchmodule[display]
2   {\textbf{Module (\smoduletitle})}\par}
3   {\par\noindent\textbf{End of Module (\smoduletitle)}}}
4
5 \begin{smodule}[type=display,title={Some New Module}]{SomeModule2}
6   Hello World
7 \end{smodule}

```

Output:

```

Module (Some New Module)
  Hello World
End of Module (Some New Module)

```

3.3.2 Declaring New Symbols and Notations

Inside an `smodule` environment, we can declare new \TeX symbols.

`\symdecl` The most basic command for doing so is using `\symdecl{symbolname}`. This introduces a new symbol with name `symbolname`, arity 0 and semantic macro `\symbolname`.

The starred variant `\symdecl*{symbolname}` will declare a symbol, but not introduce a semantic macro. If we don't want to supply a notation (for example to introduce concepts like “abelian”, which is not something that has a notation), the starred variant is likely to be what we want.

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

Without a semantic macro or a notation, the only meaningful way to reference a symbol is via `\symref`, `\symname` etc.

Example 3

Input:

```

1 \symdecl*{foo}
2 Given a \symname{foo}, we can...

```

Output:

```

Given a foo, we can...

```

Obviously, most semantic macros should take actual *arguments*, implying that the symbol we introduce is an *operator* or *function*. We can let `\symdecl` know the *arity* (i.e. number of arguments) of a symbol like this:

Example 4

Input:

```

1 \symdecl{binarysymbol}[args=2]
2 \symref{binarysymbol}{this} is a symbol taking two arguments.

```

Output:

this is a symbol taking two arguments.

So far we have gained exactly ... nothing by adding the arity information: we cannot do anything with the arguments in the text.

We will now see what we can gain with more machinery.

\notation We probably want to supply a notation as well, in which case we can finally actually use the semantic macro in math mode. We can do so using the **\notation** command, like this:

Example 5

Input:

```

1 \notation{binarysymbol}{\text{First: }#1\text{; Second: }#2}
2 $\binarysymbol{a}{b}$

```

Output:

First: *a*; Second: *b*

\hookrightarrow Applications of semantic macros, such as $\binarysymbol{a}{b}$ are translated to
 \rightarrow MMT/OMDOC as OMA-terms with head $\langle \text{OMS name} = "...?binarysymbol"/>$.
 \rightsquigarrow Semantic macros with no arguments correspond to OMS directly.

\comp For many semantic services e.g. semantic highlighting or **wikification** (linking user-visible notation components to the definition of the respective symbol they come from), we need to specify the notation components. Unfortunately, there is currently no way the \LaTeX engine can infer this by itself, so we have to specify it manually in the notation specification. We can do so with the **\comp** command.

We can introduce a new notation **highlight** for \binarysymbol that fixes this flaw, which we can subsequently use with $\binarysymbol[\text{highlight}]$:

Example 6

Input:

```

1 \notation{binarysymbol}[highlight]
2   {\comp{\text{First: }}#1\comp{\text{; Second: }}#2}
3 $\binarysymbol[highlight]{a}{b}$

```

Output:

First: a ; Second: b



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 arguments 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 `\addition{a}{b}` taking two arguments 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 `\addition`.

Similarly, a semantic macro can not conceptually be part of the notation of `\addition`, since a semantic macro represents a *distinct mathematical concept* with *its own semantics*, whereas 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), and might want to give those another thought - quite likely, the macro 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.

`\symdef`

In the vast majority of cases where a symbol declaration should come with a semantic macro, we will want to supply a notation immediately. For that reason, the `\symdef` command combines the functionality of both `\symdecl` and `\notation` with the optional arguments of both:

Example 7

Input:

```

1 \symdef{newbinarysymbol}[hl,args=2]
2   {\comp{\text{1.: }}#1\comp{\text{; 2.: }}#2}
3 $\newbinarysymbol{a}{b}$

```

Output:

```

1.: a; 2.: b

```

We just declared a new symbol `newbinarysymbol` with `args=2` and immediately provided it with a notation with identifier `hl`. Since `hl` is the *first* (and so far, only) notation supplied for `newbinarysymbol`, using `\newbinarysymbol` without optional argument defaults to this notation.

But one man’s meat is another man’s poison: it is very subjective what the “default notation” of an operator should be. Different communities have different practices. For instance, the complex unit is written as *i* in Mathematics and as *j* in electrical engineering. So to allow modular specification and facilitate re-use of document fragments \TeX allows to re-set notation defaults.

`\setnotation`

The first notation provided will stay the default notation unless explicitly changed – this is enabled by the `\setnotation` command: `\setnotation{symbolname}{notation-id}` sets the default notation of `\symbolname` to `notation-id`, i.e. henceforth, `\symbolname` behaves like `\symbolname[notation-id]` from now on.

Often, a default notation is set right after the corresponding notation is introduced – the starred version `\notation*` for that reason introduces a new notation and immediately sets it to be the new default notation. So expressed differently, the *first* `\notation` for a symbol behaves exactly like `\notation*`, and `\notation*{foo}[bar]{...}` behaves exactly like `\notation{foo}[bar]{...}\setnotation{foo}{bar}`.

`\textsymdecl`

In the less mathematical settings where we want a symbol and semantic macro for some concept with a notation *beyond* its mere name, but which should also be available in \TeX ’s text mode, the command `\textsymdecl` is useful. For example, we can declare a symbol `openmath` with the notation `\textsc{OpenMath}` using `\textsymdecl{openmath}[name=OpenMath]{\textsc{OpenMath}}`. The `\openmath` yields `OPENMATH` both in text and math mode.

Operator Notations

Once we have a semantic macro with arguments, such as `\newbinarysymbol`, the semantic macro represents the *application* of the symbol to a list of arguments. What if we want to refer to the operator *itself*, though?

We can do so by supplying the `\notation` (or `\symdef`) with an *operator notation*, indicated with the optional argument `op=`. We can then invoke the operator notation

using `\symbolname!`[notation-identifier]. Since operator notations never take arguments, we do not need to use `\comp` in it, the whole notation is wrapped in a `\comp` automatically:

Example 8

Input:

```
1 \notation{newbinarysymbol}[ab, op={\text{a:}\cdot\text{; b:}\cdot}]
2 {\comp{\text{a:}}#1\comp{\text{; b:}}#2} \symname{newbinarysymbol} is also
3 occasionally written $\newbinarysymbol![ab]$
```

Output:

newbinarysymbol is also occasionally written $a: \cdot ; b: \cdot$

\hookrightarrow `\symbolname!` is translated to OMDoc/MTT as `<OMS name="...?symbolname"/>`
 \rightarrow directly.
 \rightsquigarrow `T` \rightsquigarrow

3.3.3 Argument Modes

The notations so far used *simple* arguments which we call *mode-i* arguments. Declaring a new symbol with `\symdecl{foo}[args=3]` is equivalent to writing `\symdecl{foo}[args=iii]`, indicating that the semantic macro takes three mode-i arguments. However, there are three more argument modes which we will investigate now, namely mode-b, mode-a and mode-B arguments.

Mode-b Arguments

A mode-b argument represents a *variable* that is *bound* by the symbol in its application, making the symbol a *binding operator*. Typical examples of binding operators are e.g. sums \sum , products \prod , integrals \int , quantifiers like \forall and \exists , that λ -operator, etc.

\hookrightarrow Mode-b arguments behave exactly like mode-i arguments within T_EX, but applications of binding operators, i.e. symbols with mode-b arguments, are translated \rightarrow to OMBIND-terms in OMDoc/MTT, rather than OMA.

For example, we can implement a summation operator binding an index variable and taking lower and upper index bounds and the expression to sum over like this:

Example 9

Input:

```
1 \symdef{summation}[args=biii]
2 {\mathop{\comp{\sum}}_{\#1\comp{=}\#2}^{\#3}\#4}
3 $\summation{\svar{x}}{1}{\svar{n}}{\svar{x}}^2$
```

Output:

$$\sum_{x=1}^n x^2$$

where the variable x is now *bound* by the `\summation`-symbol in the expression.

Mode-a Arguments

Mode-a arguments represent a *flexary argument sequence*, i.e. a sequence of arguments of arbitrary length. Formally, operators that take arbitrarily many arguments don't "exist", but in informal mathematics, they are ubiquitous. Mode-a arguments allow us to write e.g. `\addition{a,b,c,d,e}` rather than having to write something like `\addition{a}{\addition{b}{\addition{c}{\addition{d}{e}}}}`!

`\notation` (and consequently `\symdef`, too) take one additional argument for each mode-a argument that indicates how to "accumulate" a comma-separated sequence of arguments. This is best demonstrated on an example.

Let's say we want an operator representing quantification over an ascending chain of elements in some set, i.e. `\ascendingchain{S}{a,b,c,d,e}{t}` should yield $\forall a <_S b <_S c <_S d <_S e. t$. The "base"-notation for this operator is simply `{\comp{\forall} \#2\comp{. ,} \#3}`, where `\#2` represents the full notation fragment *accumulated* from `{a,b,c,d,e}`.

The *additional* argument to `\notation` (or `\symdef`) takes the same arguments as the base notation and two *additional* arguments `\#1` and `\#2` representing successive pairs in the mode-a argument, and accumulates them into `\#2`, i.e. to produce $a <_S b <_S c <_S d <_S e$, we do `{\#1 \comp{<}_{\#1} \#2}`:

Example 10

Input:

```
1 \symdef{ascendingchain}[args=iai]
2   {\comp{\forall} \#2\comp{. ,} \#3}
3   {\#1 \comp{<}_{\#1} \#2}
4
5 Tadaa: $\ascendingchain{S}{a,b,c,d,e}{t}$
```

Output:

Tadaa: $\forall a <_S b <_S c <_S d <_S e. t$

If this seems overkill, keep in mind that you will rarely need the single-hash arguments `\#1`, `\#2` etc. in the a-notation-argument. For a much more representative and simpler example, we can introduce flexary addition via:

Example 11

Input:

```
1 \symdef{addition}[args=a]{\#1}{\#1 \comp{+} \#2}
2
3 Tadaa: $\addition{a,b,c,d,e}$
```

Output:

Tadaa: $a + b + c + d + e$

The `assoc`-key We mentioned earlier that “formally”, flexary arguments don’t really “exist”. Indeed, formally, addition is usually defined as a binary operation, quantifiers bind a single variable etc.

Consequently, we can tell $\text{\texttt{gT\textsubscript{E}X}}$ (or, rather, MMT/OMDOC) how to “resolve” flexary arguments by providing `\symdecl` or `\symdef` with an optional `assoc`-argument, as in `\symdecl{addition}[args=a,assoc=bin]`. The possible values for the `assoc`-key are:

`bin`: A binary, associative argument, e.g. as in `\addition`

`binl`: A binary, left-associative argument, e.g. $a^{b^{c^d}}$, which stands for $((a^b)^c)^d$

`binr`: A binary, right-associative argument, e.g. as in $A \rightarrow B \rightarrow C \rightarrow D$, which stands for $A \rightarrow (B \rightarrow (C \rightarrow D))$

`pre`: Successively prefixed, e.g. as in $\forall x, y, z. P$, which stands for $\forall x. \forall y. \forall z. P$

`conj`: Conjunctive, e.g. as in $a = b = c = d$ or $a, b, c, d \in A$, which stand for $a = d \wedge b = d \wedge c = d$ and $a \in A \wedge b \in A \wedge c \in A \wedge d \in A$, respectively

`pwconj`: Pairwise conjunctive, e.g. as in $a \neq b \neq c \neq d$, which stands for $a \neq b \wedge a \neq c \wedge a \neq d \wedge b \neq c \wedge b \neq d \wedge c \neq d$

As before, at the PDF level, this annotation is invisible (and without effect), but at the level of the generated OMDoc/MMT this leads to more semantical expressions.

Mode-B Arguments

Finally, mode-B arguments simply combine the functionality of both `a` and `b` - i.e. they represent an arbitrarily long sequence of variables to be bound, e.g. for implementing quantifiers:

Example 12

Input:

```
1 \symdef{quantforall}[args=Bi]
2   {\comp{\forall}#1\comp{.}#2}
3   {##1\comp,##2}
4
5 $\quantforall{\svar{x},\svar{y},\svar{z}}{P}$
```

Output:

$\forall x, y, z. P$

3.3.4 Type and Definiens Components

`\symdecl` and `\symdef` take two more optional arguments. \TeX largely ignores them (except for special situations we will talk about later), but MMT can pick up on them for additional services. These are the `type` and `def` keys, which expect expressions in math-mode (ideally using semantic macros, of course!)

The `type` and `def` keys correspond to the `type` and `definiens` components of \hookrightarrow OMDoc/MMT constants.
 Correspondingly, the name “type” should be taken with a grain of salt, since \hookrightarrow OMDoc/MMT – being foundation-independent – does not a priori implement a fixed typing system.

The `type`-key allows us to provide additional information (given the necessary \TeX symbols), e.g. for addition on natural numbers:

Example 13

Input:

```
1 \symdef{Nat}[type=\set]{\comp{\mathbb N}}
2 \symdef{addition}[
3   type=\funtype{\Nat,\Nat}{\Nat},
4   op=+,
5   args=a
6 ]{\#1}{\#1 \comp+ \#2}
7
8 \symname{addition} is an operation $\funtype{\Nat,\Nat}{\Nat}$
```

Output:

addition is an operation $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

.

The `def`-key allows for declaring symbols as abbreviations:

Example 14

Input:

```
1 \symdef{successor}[
2   type=\funtype{\Nat}{\Nat},
3   def=\fun{\svar{x}}{\addition{\svar{x},1}},
4   op=\mathtt{succ},
5   args=1
6 ]{\comp{\mathtt{succ}(\#1\comp{)}}}
7
8 The \symname{successor} operation $\funtype{\Nat}{\Nat}$
9 is defined as $\fun{\svar{x}}{\addition{\svar{x},1}}$
```

Output:

The *successor* operation $\mathbb{N} \rightarrow \mathbb{N}$ is defined as $x \mapsto x + 1$

.

3.3.5 Precedences and Automated Bracketing

Having done `\addition`, the obvious next thing to implement is `\multiplication`. This is straight-forward in theory:

Example 15

Input:

```
1 \symdef{multiplication}[
2   type=\funtype{\Nat,\Nat}{\Nat},
3   op=\cdot,
4   args=a
5 ]{\#1}{\#1 \comp\cdot \#2}
6
7 \symname{multiplication} is an operation $\funtype{\Nat,\Nat}{\Nat}$
```

Output:

multiplication is an operation $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

However, if we *combine* `\addition` and `\multiplication`, we notice a problem:

Example 16

Input:

```
1 $\addition{a,\multiplication{b,\addition{c,\multiplication{d,e}}}}$
```

Output:

$a + b \cdot c + d \cdot e$

We all know that \cdot binds stronger than $+$, so the output $a + b \cdot c + d \cdot e$ does not actually reflect the term we wrote. We can of course insert parentheses manually

Example 17

Input:

```
1 $\addition{a,\multiplication{b,(\addition{c,\multiplication{d,e}})}}$
```

Output:

$a + b \cdot (c + d \cdot e)$

but we can also do better by supplying *precedences* and have \LaTeX insert parentheses automatically.

For that purpose, `\notation` (and hence `\symdef`) take an optional argument `prec=<opprec>;<argprec1>x...x<argprec n>`.

We will investigate the precise meaning of `<opprec>` and the `<argprec>`s shortly – in the vast majority of cases, it is perfectly sufficient to think of `prec=` taking a single number

and having that be *the* precedence of the notation, where lower precedences (somewhat counterintuitively) bind stronger than higher precedences. So fixing our notations for `\addition` and `\multiplication`, we get:

Example 18

Input:

```

1 \notation{multiplication}[
2   op=\cdot,
3   prec=50
4 ]{#1}{##1 \comp\cdot ##2}
5 \notation{addition}[
6   op=+,
7   prec=100
8 ]{#1}{##1 \comp+ ##2}
9
10 $\addition{a,\multiplication{b,\addition{c,\multiplication{d,e}}}}$

```

Output:

$$a + b \cdot (c + d \cdot e)$$

Note that the precise numbers used for precedences are pretty arbitrary - what matters is which precedences are higher than which other precedences when used in conjunction.

`\infprec`
`\neginfprec`

It is occasionally useful to have “infinitely” high or low precedences to enforce or forbid automated bracketing entirely, e.g. for bracket-like notations such as intervals – for those purposes, `\infprec` and `\neginfprec` exist (which are implemented as the maximal and minimal integer values accordingly).g



More precisely, each notation takes

1. One *operator precedence* and
2. one *argument precedence* for each argument.

By default, all precedences are 0, unless the symbol takes no argument, in which case the operator precedence is `\neginfprec` (negative infinity). If we only provide a single number, this is taken as both the operator precedence and all argument precedences.

`\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` insert parentheses.

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

In the example above:

1. `\TeX` starts out with $p_d = \text{\code{\infprec}}$.
2. `\TeX` encounters `\addition` with $p_{op} = 100$. Since $100 \not> \text{\code{\infprec}}$, it inserts no parentheses.



3. Next, \STeX encounters the two arguments for \addition . Both have no specifically provided argument precedence, so \STeX uses $p_d = p_{op} = 100$ for both and recurses.
4. Next, \STeX encounters $\text{\multiplication}\{\mathbf{b}, \dots\}$, whose notation has $p_{op} = 50$.
5. We compare to the current downward precedence p_d set by \addition , arriving at $p_{op} = 50 \not\geq 100 = p_d$, so \STeX again inserts no parentheses.
6. Since the notation of \multiplication has no explicitly set argument precedences, \STeX uses the operator precedence for all arguments of \multiplication , hence sets $p_d = p_{op} = 50$ and recurses.
7. Next, \STeX encounters the inner $\text{\addition}\{c, \dots\}$ whose notation has $p_{op} = 100$.
8. We compare to the current downward precedence p_d set by \multiplication , arriving at $p_{op} = 100 > 50 = p_d$ – which finally prompts \STeX to insert parentheses, and we proceed as before.

3.3.6 Variables

All symbol and notation declarations require a module with which they are associated, hence the commands \symdecl , \notation , \symdef etc. are disabled outside of \smodule -environments.

Variables are different – variables are allowed everywhere, are not exported when the current module (if one exists) is imported (via \importmodule or \usemodule) and (also unlike symbol declarations) “disappear” at the end of the current \TeX group.

\svar So far, we have always used variables using $\text{\svar}\{n\}$, which marks-up n as a variable with name n . More generally, $\text{\svar}[foo]\{\text{\textcode}\}$ marks-up the arbitrary \textcode as representing a variable with name foo .

Of course, this makes it difficult to reuse variables, or introduce “functional” variables with arities > 0 , or provide them with a type or definiens.

\vardef For that, we can use the \vardef command. Its syntax is largely the same as that of \symdef , but unlike symbols, variables have only one notation (**TODO: so far?**), hence there is only \vardef and no \vardecl .

Example 19

Input:


```

1 \vardef{varf}[
2   name=f,
3   type=\funtype{\Nat}{\Nat},
4   op=f,
5   args=1,
6   prec=0;\neginfprec
7 ]{\comp{f}#1}
8 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
9 \vardef{varx}[name=x,type=\Nat]{\comp{x}}
10
11 Given a function  $\text{\textbackslash varf!}:\text{\textbackslash funtype}\{\text{\textbackslash Nat}\}\{\text{\textbackslash Nat}\}$ ,
12 by  $\text{\textbackslash addition}\{\text{\textbackslash varf!},\text{\textbackslash varn}\}$  we mean the function
13  $\text{\textbackslash fun}\{\text{\textbackslash varx}\}\{\text{\textbackslash varf}\{\text{\textbackslash addition}\{\text{\textbackslash varx},\text{\textbackslash varn}\}\}\}$ 

```

Output:

Given a function $f : \mathbb{N} \rightarrow \mathbb{N}$, by $f + n$ we mean the function $x \mapsto f(x + n)$

(of course, “lifting” addition in the way described in the previous example is an operation that deserves its own symbol rather than abusing `\addition`, but... well.)

TODO: `bind=forall/exists`

3.3.7 Variable Sequences

Variable *sequences* occur quite frequently in informal mathematics, hence they deserve special support. Variable sequences behave like variables in that they disappear at the end of the current $\text{\textbackslash TeX}$ group and are not exported from modules, but their declaration is quite different.

\varseq A variable sequence is introduced via the command `\varseq`, which takes the usual optional arguments `name` and `type`. It then takes a starting index, an end index and a *notation* for the individual elements of the sequence parametric in an index. Note that both the starting as well as the ending index may be variables.

This is best shown by example:

Example 20

Input:

```

1 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
2 \varseq{seqa}[name=a,type=\Nat]{1}{\varn}{\comp{a}_\#1}
3
4 The  $\text{\textbackslash i}$ th index of  $\text{\textbackslash seqa}$  is  $\text{\textbackslash seqa}\{\text{\textbackslash i}\}$ .

```

Output:

The i th index of a_1, \dots, a_n is a_i .

Note that the syntax `\seqa!` now automatically generates a presentation based on the starting and ending index.

TODO: more notations for invoking sequences.

Notably, variable sequences are nicely compatible with **a**-type arguments, so we can do the following:

Example 21

Input:

```
1 $\addition{\seqa}$
```

Output:

$$a_1 + \dots + a_n$$

Sequences can be *multidimensional* using the **args**-key, in which case the notation's arity increases and starting and ending indices have to be provided as a comma-separated list:

Example 22

Input:

```
1 \vardef{var}{name=m,type=\Nat}{\comp{m}}
2 \varseq{seqa}[
3   name=a,
4   args=2,
5   type=\Nat,
6 ]{1,1}{\varn,\var}{\comp{a}_{\#1}^{\#2}}
7
8 $\seqa!$ and $\addition{\seqa}$
```

Output:

$$a_1^1, \dots, a_n^m \text{ and } a_1^1 + \dots + a_n^m$$

We can also explicitly provide a “middle” segment to be used, like such:

Example 23

Input:

```
1 \varseq{seqa}[
2   name=a,
3   type=\Nat,
4   args=2,
5   mid={\comp{a}_{\varn}^1, \comp{a}_1^2, \ellipses, \comp{a}_{1}^{\varn}}
6 ]{1,1}{\varn,\var}{\comp{a}_{\#1}^{\#2}}
7
8 $\seqa!$ and $\addition{\seqa}$
```

Output:

$$a_1^1, \dots, a_n^1, a_1^2, \dots, a_n^m, \dots, a_n^m \text{ and } a_1^1 + \dots + a_n^1 + a_1^2 + \dots + a_1^m + \dots + a_n^m$$

3.4 Module Inheritance and Structures

The $\text{\S}\text{\TeX}$ features for modular document management are inherited from the OM-Doc/MMT model that organizes knowledge into a graph, where the nodes are theories (called modules in $\text{\S}\text{\TeX}$) and the edges are truth-preserving mappings (called theory morphismes in MMT). We have already seen modules/theories above.

Before we get into theory morphisms in $\text{\S}\text{\TeX}$ we will see a very simple application of modules: managing multilinguality modularly.

3.4.1 Multilinguality and Translations

If we load the $\text{\S}\text{\TeX}$ document class or package with the option `lang=<lang>`, $\text{\S}\text{\TeX}$ will load the appropriate `babel` language for you – e.g. `lang=de` will load the `babel` language `ngerman`. Additionally, it makes $\text{\S}\text{\TeX}$ aware of the current document being set in (in this example) *german*. This matters for reasons other than mere `babel`-purposes, though:

Every *module* is assigned a language. If no $\text{\S}\text{\TeX}$ package option is set that allows for inferring a language, $\text{\S}\text{\TeX}$ will check whether the current file name ends in e.g. `.en.tex` (or `.de.tex` or `.fr.tex`, or...) and set the language accordingly. Alternatively, a language can be explicitly assigned via `\begin{smodule}[lang=<language>]{Foo}`.

Technically, each `smodule`-environment induces *two* OMDoc/MMT theories:
 \hookrightarrow `\begin{smodule}[lang=<lang>]{Foo}` generates a theory `some/namespace?Foo` that only contains the “formal” part of the module – i.e. exactly the content that is exported when using `\importmodule`.
 \rightsquigarrow Additionally, MMT generates a *language theory* `some/namespace/Foo?<lang>` that includes `some/namespace?Foo` and contains all the other document content – variable declarations, includes for each `\usemodule`, etc.

Notably, the language suffix in a filename is ignored for `\usemodule`, `\importmodule` and in generating/computing URIs for modules. This however allows for providing *translations* for modules between languages without needing to duplicate content:

If a module `Foo` exists in e.g. english in a file `Foo.en.tex`, we can provide a file `Foo.de.tex` right next to it, and write `\begin{smodule}[sig=en]{Foo}`. The `sig`-key then signifies, that the “signature” of the module is contained in the *english* version of the module, which is immediately imported from there, just like `\importmodule` would.

Additionally to translating the informal content of a module file to different languages, it also allows for customizing notations between languages. For example, the *least common multiple* of two numbers is often denoted as $\text{lcm}(a, b)$ in english, but is called *kleinstes gemeinsames Vielfaches* in german and consequently denoted as $\text{kgV}(a, b)$ there.

We can therefore imagine a german version of an `lcm`-module looking something like this:

```

1 \begin{smodule}[sig=en]{lcm}
2   \notation*{lcm}[de]{\comp{\mathtt{kgV}}}{\#1,\#2}
3
4   Das \symref{lcm}{kleinste gemeinsame Vielfache}
5   $\text{lcm}\{a,b\}$ von zwei Zahlen $a,b$ ist...
```

```
6 \end{smodule}
```

If we now do `\importmodule{1cm}` (or `\usemodule{1cm}`) within a *german* document, it will also load the content of the german translation, including the `de`-notation for `\1cm`.

3.4.2 Simple Inheritance and Namespaces

`\importmodule` `\importmodule[Some/Archive]{path?ModuleName}` is only allowed within an `smodule`-environment and makes the symbols declared in `ModuleName` available therein. Additionally the symbols of `ModuleName` will be exported if the current module is imported somewhere else via `\importmodule`.

`\usemodule` behaves the same way, but without exporting the content of the used module.

It is worth going into some detail how exactly `\importmodule` and `\usemodule` 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 URI.



Ideally, \TeX would use arbitrary URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring the module – like MMT does things.

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 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 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`-folder 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 document 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 document, 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`-folder.

- 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`-folder, 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.
- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document.

Since this is less compatible with a modular development, using full URIs directly is strongly discouraged, unless the module is declared in the current file directly.

`\STEXexport` `\importmodule` and `\usemodule` import all symbols, notations, semantic macros and (recursively) `\importmodules`. If you want to additionally export e.g. convenience macros and other (S_{TE}X) code from a module, you can use the command `\STEXexport{<code>}` in your module. Then `<code>` is executed (both immediately and) every time the current module is opened via `\importmodule` or `\usemodule`.



For persistency reasons, everything in an `\STEXexport` is digested by T_EX in the L^AT_EX3-category code scheme. This means that the characters `_` and `:` are considered *letters* and valid parts of control sequence names, and space characters are ignored entirely. For spaces, use the character `~` instead, and keep in mind, that if you want to use subscripts, you should use `\c_math_subscript_token` instead of `_`!

Also note, that `\newcommand` defines macros *globally* and throws an error if the macro already exists, potentially leading to low-level L^AT_EX errors if we put a `\newcommand` in an `\STEXexport` and the `<code>` is executed more than once in a document – which can happen easily.

A safer alternative is to use macro definition principles, that are safe to use even if the macro being defined already exists, and ideally are local to the current T_EX group, such as `\def` or `\let`.

3.4.3 The `mathstructure` Environment

A common occurrence in mathematics is bundling several interrelated “declarations” together into *structures*. For example:

- A *monoid* is a structure $\langle M, \circ, e \rangle$ with $\circ : M \times M \rightarrow M$ and $e \in M$ such that...

- A *topological space* is a structure $\langle X, \mathcal{T} \rangle$ where X is a set and \mathcal{T} is a topology on X
- A *partial order* is a structure $\langle S, \leq \rangle$ where \leq is a binary relation on S such that...

This phenomenon is important and common enough to warrant special support, in particular because it requires being able to *instantiate* such structures (or, rather, structure *signatures*) in order to talk about (concrete or variable) *particular* monoids, topological spaces, partial orders etc.

`mathstructure` (*env.*) The `mathstructure` environment allows us to do exactly that. It behaves exactly like the `smodule` environment, but is itself only allowed inside an `smodule` environment, and allows for instantiation later on.

How this works is again best demonstrated by example:

Example 24

Input:

```

1 \begin{mathstructure}{monoid}
2   \symdef{universe}[type=\set]{\comp{U}}
3   \symdef{op}[
4     args=2,
5     type=\funtype{\universe,\universe}{\universe},
6     op=\circ
7   ]{##1 \comp{\circ} ##2}
8   \symdef{unit}[type=\universe]{\comp{e}}
9 \end{mathstructure}
10
11 A \symname{monoid} is...
```

Output:

A *monoid* is...

Note that the `\symname{monoid}` is appropriately highlighted and (depending on your pdf viewer) shows a URI on hovering – implying that the `mathstructure` environment has generated a *symbol* monoid for us. It has not generated a semantic macro though, since we can not use the monoid-symbol *directly*. Instead, we can instantiate it, for example for integers:

Example 25

Input:

```

1 \symdef{Int}[type=\set]{\comp{\mathbb Z}}
2 \symdef{addition}[
3   type=\funtype{\Int,\Int}{\Int},
4   args=2,
5   op=+
6 ]{##1 \comp{+} ##2}
7 \symdef{zero}[type=\Int]{\comp{0}}
8
9 $\mathstruct{\Int,\addition!,\zero}$ is a \symname{monoid}.
```

Output:

$\langle \mathbb{Z}, +, 0 \rangle$ is a *monoid*.

So far, we have not actually instantiated `monoid`, but now that we have all the symbols to do so, we can:

Example 26

Input:

```
1 \instantiate{intmonoid}{monoid}{\mathbb{Z}_{+,0}}[
2   universe = Int ,
3   op = addition ,
4   unit = zero
5 ]
6
7 $\intmonoid{universe}$, $\intmonoid{unit}$ and $\intmonoid{op}\{a\}\{b\}$.
8
9 Also: $\intmonoid!$
```

Output:

\mathbb{Z} , 0 and $a + b$.
Also: $\mathbb{Z}_{+,0}$

\instantiate

So summarizing: `\instantiate` takes four arguments: The (macro-)name of the instance, a key-value pair assigning declarations in the corresponding `mathstructure` to symbols currently in scope, the name of the `mathstructure` to instantiate, and lastly a notation for the instance itself.

It then generates a semantic macro that takes as argument the name of a declaration in the instantiated `mathstructure` and resolves it to the corresponding instance of that particular declaration.

`\instantiate` and `mathstructure` make use of the *Theories-as-Types* paradigm (see [MRK18]):

- `mathstructure{<name>}` simply creates a nested theory with name `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)`
- `→M` – a *dependent record type with manifest fields*, the fields of which are generated from (and correspond to) the constants in `<name>-structure`.
- `→T` – generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned based on the respective key-value-list.

Notably, `\instantiate` throws an error if not *every* declaration in the instantiated `mathstructure` is being assigned.

You might consequently ask what the usefulness of `mathstructure` even is.

`\varinstantiate`

The answer is that we can also instantiate a **mathstructure** with a *variable*. The syntax of `\varinstantiate` is equivalent to that of `\instantiate`, but all of the key-value-pairs are optional, and if not explicitly assigned (to a symbol *or* a variable declared with `\vardef`) inherit their notation from the one in the **mathstructure** environment.

This allows us to do things like:

Example 27

Input:

```
1 \varinstantiate{varM}{monoid}{M}
2
3 A \symname{monoid} is a structure
4 $\varM!:=\mathstruct{\varM{universe},\varM{op}!,\varM{unit}}{ }$
5 such that
6 $\varM{op}!: \funtype{\varM{universe},\varM{universe}}{\varM{universe}}$ ...
```

Output:

A *monoid* is a structure $M := \langle U, \circ, e \rangle$ such that $\circ : U \times U \rightarrow U$...

.

and

Example 28

Input:

```
1 \varinstantiate{varMb}{monoid}{M_2}[universe = Int]
2
3 Let $\varMb!:=\mathstruct{\varMb{universe},\varMb{op}!,\varMb{unit}}{ }$
4 be a \symname{monoid} on $\Int$ ...
```

Output:

Let $M_2 := \langle \mathbb{Z}, \circ, e \rangle$ be a *monoid* on \mathbb{Z} ...

.

We will return to these two example later, when we also know how to handle the *axioms* of a monoid.

3.4.4 The copymodule Environment

TODO: explain

Given modules:

Example 29

Input:


```

1 \begin{smodule}{magma}
2   \symdef{universe}{\comp{\mathcal U}}
3   \symdef{operation}[args=2,op=\circ]{#1 \comp\circ #2}
4 \end{smodule}
5 \begin{smodule}{monoid}
6   \importmodule{magma}
7   \symdef{unit}{\comp e}
8 \end{smodule}
9 \begin{smodule}{group}
10  \importmodule{monoid}
11  \symdef{inverse}[args=1]{#1\comp{-1}}
12 \end{smodule}

```

Output:

.

We can form a module for *rings* by “cloning” an instance of `group` (for addition) and `monoid` (for multiplication), respectively, and “glueing them together” to ensure they share the same universe:

Example 30

Input:

```

1 \begin{smodule}{ring}
2   \begin{copymodule}{group}{addition}
3     \renamedecl[name=universe]{universe}{runiverse}
4     \renamedecl[name=plus]{operation}{rplus}
5     \renamedecl[name=zero]{unit}{rzero}
6     \renamedecl[name=uminus]{inverse}{ruminus}
7   \end{copymodule}
8   \notation*{rplus}[plus,op=+,prec=60]{#1 \comp+ #2}
9   \notation*{rzero}[zero]{\comp0}
10  \notation*{ruminus}[uminus,op=-]{\comp- #1}
11  \begin{copymodule}{monoid}{multiplication}
12    \assign{universe}{runiverse}
13    \renamedecl[name=times]{operation}{rtimes}
14    \renamedecl[name=one]{unit}{rone}
15  \end{copymodule}
16  \notation*{rtimes}[cdot,op=\cdot,prec=50]{#1 \comp\cdot #2}
17  \notation*{rone}[one]{\comp1}
18  Test: $\rtimes a\{rplus c\{rtimes de\}}$
19 \end{smodule}

```

Output:

Test: $a \cdot (c + d \cdot e)$

TODO: explain donotclone

3.4.5 The interpretmodule Environment

TODO: explain

Example 31

Input:

```
1 \begin{smodule}{int}
2   \symdef{Integers}{\comp{\mathbb Z}}
3   \symdef{plus}[args=2,op=+]{#1 \comp+ #2}
4   \symdef{zero}{\comp0}
5   \symdef{uminus}[args=1,op=-]{\comp-#1}
6
7   \begin{interpretmodule}{group}{intisgroup}
8     \assign{universe}{\Integers}
9     \assign{operation}{\plus!}
10    \assign{unit}{\zero}
11    \assign{inverse}{\uminus!}
12  \end{interpretmodule}
13 \end{smodule}
```

Output:

3.5 Primitive Symbols (The $\text{\texttt{STEX}}$ Metatheory)

The `stex-metatheory` package contains $\text{\texttt{STEX}}$ symbols so ubiquitous, that it is virtually impossible to describe any flexiformal content without them, or that are required to annotate even the most primitive symbols with meaningful (foundation-independent) “type”-annotations, or required for basic structuring principles (theorems, definitions). As such, it serves as the default meta theory for any $\text{\texttt{STEX}}$ module.

We can also see the `stex-metatheory` as a foundation of mathematics in the sense of [Rab15], albeit an informal one (the ones discussed there are all formal foundations). The state of the `stex-metatheory` is necessarily incomplete, and will stay so for a long while: It arises as a collection of empirically useful symbols that are collected as more and more mathematics are encoded in $\text{\texttt{STEX}}$ and are classified as foundational.

Formal foundations should ideally instantiate these symbols with their formal counterparts, e.g. `isa` corresponds to a typing operation in typed setting, or the \in -operator in set-theoretic contexts; `bind` corresponds to a universal quantifier in (n th-order) logic, or a Π in dependent type theories.

We make this theory part of the $\text{\texttt{STEX}}$ collection due to the obiquity of the symbols involved. Note however, that the metatheory is for all practical purposes a “normal” $\text{\texttt{STEX}}$ module, and the symbols contained “normal” $\text{\texttt{STEX}}$ symbols.

Chapter 4

Using \TeX Symbols

Given a symbol declaration `\symdecl{symbolname}`, we obtain a semantic macro `\symbolname`. We can use this semantic macro in math mode to use its notation(s), and we can use `\symbolname!` in math mode to use its operator notation(s). What else can we do?

4.1 `\symref` and its variants

<code>\symref</code> <code>\symname</code>	We have already seen <code>\symname</code> and <code>\symref</code> , the latter being the more general. <code>\symref{<symbolname>}{<code>}</code> marks-up <code><code></code> as referencing <code><symbolname></code> . Since quite often, the <code><code></code> should be (a variant of) the name of the symbol anyway, we also have <code>\symname{<symbolname>}</code> .
---	--

Note that `\symname` uses the *name* of a symbol, not its macroname. More precisely, `\symname` will insert the name of the symbol with “-” replaced by spaces. If a symbol does not have an explicit `name=` given, the two are equal – but for `\symname` it often makes sense to make the two explicitly distinct. For example:

Example 32

Input:

```
1 \symdef{Nat}[
2   name=natural-number,
3   type=\set
4 ]{\comp{\mathbb{N}}}
5
6 A \symname{Nat} is...
```

Output:

A *natural number* is...

`\symname` takes two additional optional arguments, `pre=` and `post=` that get prepended or appended respectively to the symbol name.

`\Symname` Additionally, `\Symname` behaves exactly like `\symname`, but will capitalize the first letter of the name:

Example 33

Input:

```
1 \Symname[post=s]{Nat} are...
```

Output:

Natural numbers are...



This is as good a place as any other to explain how \TeX resolves a string `symbolname` to an actual symbol.

If `\symbolname` is a semantic macro, then \TeX has no trouble resolving `symbolname` to the full 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...`. \TeX attempts to handle this case thusly:

If `string` does *not* correspond to a semantic macro `\string` and does *not* contain a `?`, then \TeX checks all symbols currently in scope until it finds one, whose name is `string`. If `string` is of the form `pre?name`, \TeX first looks through all modules currently in scope, whose full 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.

4.2 Marking Up Text and On-the-Fly Notations

We can also use semantic macros outside of text mode though, which allows us to annotate arbitrary text fragments.

Let us assume again, that we have `\symdef{addition}[args=2]{#1 \comp+ #2}`. Then we can do

Example 34

Input:

```
1 \addition{\comp{The sum of} \arg{${\svar{n}}$} \comp{ and } \arg{${\svar{m}}$}}
2 is...
```

Output:

The sum of *n* and *m* is...

...which marks up the text fragment as representing an *application* of the `addition`-symbol to two argument n and m .

\hookrightarrow As expected, the above example is translated to OMDoc/MMT as an
 \hookrightarrow OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and
 \hookrightarrow `<OMV name="m"/>` as arguments.



Note the difference in treating “arguments” between math mode and text mode. In math mode the (in this case two) tokens/groups following the `\addition` macro are treated as arguments to the addition function, whereas in text mode the group following `\addition` is taken to be the ad-hoc presentation. We drill in on this now.

\arg In text mode, every semantic macro takes exactly one argument, namely the text-fragment to be annotated. The `\arg` command is only valid within the argument to a semantic macro and marks up the *individual arguments* for the symbol.

We can also use semantic macros in text mode to invoke an operator itself instead of its application, with the usual syntax using `!`:

Example 35

Input:

```
1 \addition!{Addition} is...
```

Output:

```
Addition is...
```

Indeed, `\symbolname!{<code>}` is exactly equivalent to `\symref{symbolname}{<code>}` (the latter is in fact implemented in terms of the former).

`\arg` also allows us to switch the order of arguments around and “hide” arguments: For example, `\arg[3]{<code>}` signifies that `<code>` represents the *third* argument to the current operator, and `\arg*[i]{<code>}` signifies that `<code>` represents the *i*th argument, but it should not produce any output (it is exported in the `xhtml` however, so that MMT and other systems can pick up on it).¹

Example 36

Input:

```
1 \addition{\comp{adding}
2   \arg[2]{\svar{k}}
3   \arg*{\svar{n}}{\svar{m}}}} yields...
```

Output:

¹EDNOTE: MK: I do not understand why we have to/want to give the second `arg*`; I think this must be elaborated on.

adding k yields...

Note that since the second `\arg` has no explicit argument number, it automatically represents the first not-yet-given argument – i.e. in this case the first one.²

The same syntax can be used in math mod as well. This allows us to spontaneously introduce new notations on the fly. We can activate it using the starred variants of semantic macros:

Example 37

Input:

```

1 Given $\addition{\svar{n}}{\svar{m}}$, then
2 $\addition*{
3   \arg*{\addition{\svar{n}}{\svar{m}}}
4   \comp{+}
5   \arg{\svar{k}}
6 }$ yields...
```

Output:

Given $n + m$, then $+k$ yields...

4.3 Referencing Symbols and Statements

TODO: references documentation

²EdNOTE: MK: I do not understand this at all.

Chapter 5

STEX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

As mentioned earlier, we can semantically mark-up *statements* such as definitions, theorems, lemmata, examples, etc.

The corresponding environments for that are:

- `sdefinition` for definitions,
- `sassertion` for assertions, i.e. propositions that are declared to be *true*, such as theorems, lemmata, axioms,
- `sexample` for examples and counterexamples, and
- `sparagraph` for “other” semantic paragraphs, such as comments, remarks, conjectures, etc.

The *presentation* of these environments can be customized to use e.g. predefined theorem-environments, see [section 5.3](#) for details.

All of these environments take optional arguments in the form of `key=value`-pairs. Common to all of them are the keys `id=` (for cross-referencing, see [section 4.3](#)), `type=` for customization (see [section 5.3](#)) and additional information (e.g. definition principles, “difficulty” etc), as well as `title=` (for giving the paragraph a title), and finally `for=`.

The `for=` key expects a comma-separated list of existing symbols, allowing for e.g. things like

Example 38







Input:

```
1 \begin{sexample}[
2   id=additionandmultiplication.ex,
3   for={addition,multiplication},
4   type={trivial,boring},
5   title={An Example}
6 ]
7   $\addition{2,3}$ is $5$, $\multiplication{2,3}$ is $6$.
8 \end{sexample}
```

Output:

Example 5.1.1 (An Example). $2 + 3$ is 5, $2 \cdot 3$ is 6.

`\definiendum` `sdefinition` (and `sparagraph` with `type=symdoc`) introduce three new macros: `definiendum` behaves like `symref` (and `definame`/`Definame` like `symname`/`Symname`, respectively), but `\Definame` highlights the referenced symbol as *being defined* in the current definition.

M The special `type=symdoc` for `sparagraph` is intended to be used for “informal definitions”, or encyclopedia-style descriptions for symbols.
M The MMT system can use those (in lieu of an actual `sdefinition` in scope) to
T present to users, e.g. when hovering over symbols.

`\definiens` Additionally, `sdefinition` (and `sparagraph` with `type=symdoc`) introduces `\definiens`[<optional sym which marks up <code> as being the explicit *definiens* of <optional symbolname> (in case `for=` has multiple symbols).

All four statement environments – i.e. `sdefinition`, `sassertion`, `sexample`, and `sparagraph` – also take an optional parameter `name=` – if this one is given a value, the environment will generate a *symbol* by that name (but with no semantic macro). Not only does this allow for `\symref` et al, it allows us to resume our earlier example for monoids much more nicely:³

Example 39

Input:

³EdNOTE: MK: we should reference the example explicitly here.


```

1 \begin{mathstructure}{monoid}
2   \syndef{universe}[type=\set]{\comp{U}}
3   \syndef{op}[
4     args=2,
5     type=\funtype{\universe,\universe}{\universe},
6     op=\circ
7   ]{#1 \comp{\circ} #2}
8   \syndef{unit}[type=\universe]{\comp{e}}
9
10  \begin{sparagraph}[type=symdoc,for=monoid]
11    A \definame{monoid} is a structure
12    $\mathstruct{\universe,\op!,\unit}$
13    where $\op!:\funtype{\universe}{\universe}$ and
14    $\inset{\unit}{\universe}$ such that
15
16    \begin{sassertion}[name=associative,
17      type=axiom,
18      title=Associativity]
19      $\op!$ is associative
20    \end{sassertion}
21    \begin{sassertion}[name=isunit,
22      type=axiom,
23      title=Unit]
24      $\equal{\op{\svar{x}}{\unit}}{\svar{x}}$
25      for all $\inset{\svar{x}}{\universe}$
26    \end{sassertion}
27  \end{sparagraph}
28 \end{mathstructure}
29
30 An example for a \symname{monoid} is...

```

Output:

A **monoid** is a structure $\langle U, \circ, e \rangle$ where $\circ : U \rightarrow U$ and $e \in U$ such that

Axiom 5.1.2 (Associativity). \circ is associative

Axiom 5.1.3 (Unit). $x \circ e = x$ for all $x \in U$

An example for a *monoid* is...

EdN:4

The main difference to before⁴ is that the two **sassertions** now have **name=** attributes. Thus the **mathstructure** *monoid* now contains two additional symbols, namely the axioms for associativity and that e is a unit. Note that both symbols do not represent the mere *propositions* that e.g. \circ is associative, but *the assertion that it is actually true* that \circ is associative.

If we now want to instantiate **monoid** (unless with a variable, of course), we also need to assign **associative** and **neutral** to analogous assertions. So the earlier example

```

1 \instantiate{intmonoid}{monoid}{\mathbb{Z}_{+,0}}[
2   universe = Int ,
3   op = addition ,
4   unit = zero
5 ]

```

⁴EdNOTE: MK: reference

...will not work anymore. We now need to give assertions that addition is associative and that zero is a unit with respect to addition.²

5.2 Proofs

The `stex-proof` package supplies macros and environment that allow to annotate the structure of mathematical proofs in \LaTeX document. This structure can be used by MKM systems for added-value services, either directly from the \LaTeX sources, or after translation.

We will go over the general intuition by way of a running example:

```

1 \begin{spproof}[id=simple-proof]
2   {We prove that  $\sum_{i=1}^n 2i-1 = n^2$  by induction over  $n$ }
3   \begin{spfcases}{For the induction we have to consider three cases:}
4     \begin{spfcase}{ $n=1$ }
5       \begin{spfstep}[type=inline] then we compute  $1=1^2$  \end{spfstep}
6     \end{spfcase}
7     \begin{spfcase}{ $n=2$ }
8       \begin{spfcomment}[type=inline]
9         This case is not really necessary, but we do it for the
10         fun of it (and to get more intuition).
11       \end{spfcomment}
12       \begin{spfstep}[type=inline] We compute  $1+3=2^2=4$ . \end{spfstep}
13     \end{spfcase}
14     \begin{spfcase}{ $n>1$ }
15       \begin{spfstep}[type=assumption,id=ind-hyp]
16         Now, we assume that the assertion is true for a certain  $k \geq 1$ ,
17         i.e.  $\sum_{i=1}^k (2i-1) = k^2$ .
18       \end{spfstep}
19       \begin{spfcomment}
20         We have to show that we can derive the assertion for  $n=k+1$  from
21         this assumption, i.e.  $\sum_{i=1}^{k+1} (2i-1) = (k+1)^2$ .
22       \end{spfcomment}
23       \begin{spfstep}
24         We obtain  $\sum_{i=1}^{k+1} (2i-1) = \sum_{i=1}^k (2i-1) + 2(k+1) - 1$ 
25         \spfjust[method=arith:split-sum]{by splitting the sum}.
26       \end{spfstep}
27       \begin{spfstep}
28         Thus we have  $\sum_{i=1}^{k+1} (2i-1) = k^2 + 2k + 1$ 
29         \spfjust[method=fertilize]{by inductive hypothesis}.
30       \end{spfstep}
31       \begin{spfstep}[type=conclusion]
32         We can \spfjust[method=simplify]{simplify} the right-hand side to
33          $(k+1)^2$ , which proves the assertion.
34       \end{spfstep}
35     \end{spfcase}
36     \begin{spfstep}[type=conclusion]
37       We have considered all the cases, so we have proven the assertion.
38     \end{spfstep}
39   \end{spfcases}
40 \end{spproof}

```

This yields the following result:

²Of course, \LaTeX can not check that the assertions are the “correct” ones – but if the assertions (both in monoid as well as those for addition and zero) are properly marked up, MMT can. **TODO: should**

Proof: We prove that $\sum_{i=1}^n 2i - 1 = n^2$ by induction over n

1. For the induction we have to consider the following cases:

1.1. $n = 1$: then we compute $1 = 1^2$ □

1.2. $n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$ □

1.3. $n > 1$:

1.3.1. Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^k (2i - 1) = k^2$.

1.3.2. We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.

1.3.3. We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum.

1.3.4. Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.

1.3.5. We can simplify the right-hand side to $(k + 1)^2$, which proves the assertion. □

1.4. We have considered all the cases, so we have proven the assertion. □

spproof (*env.*) The **spproof** environment is the main container for proofs. It takes an optional **KeyVal** argument that allows to specify the **id** (identifier) and **for** (for which assertion is this a proof) keys. The regular argument of the **proof** environment contains an introductory comment, that may be used to announce the proof style. The **proof** environment contains a sequence of **spfststep**, **spfstcomment**, and **spfstcases** environments that are used to markup the proof steps.

\spfstidea The **\spfstidea** macro allows to give a one-paragraph description of the proof idea.

\spfstsketch For one-line proof sketches, we use the **\spfstsketch** macro, which takes the same optional argument as **spproof** and another one: a natural language text that sketches the proof.

spfststep (*env.*) Regular proof steps are marked up with the **step** environment, which takes an optional **KeyVal** argument for annotations. A proof step usually contains a local assertion (the text of the step) together with some kind of evidence that this can be derived from already established assertions.

<u><code>\spfjust</code></u>	This evidence is marked up with the <code>\spfjust</code> macro in the <code>stex-proofs</code> package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence. The environment takes an optional <code>KeyVal</code> argument, which can have the <code>method</code> key, whose value is the name of a proof method (this will only need to mean something to the application that consumes the semantic annotations). Furthermore, the justification can contain “premises” (specifications to assertions that were used justify the step) and “arguments” (other information taken into account by the proof method).
<u><code>\premise</code></u>	The <code>\premise</code> macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the running example we have used the <code>\premise</code> macro to identify the inductive hypothesis.
<u><code>\justarg</code></u>	<p>The <code>\justarg</code> macro is very similar to <code>\premise</code> with the difference that it is used to mark up arguments to the proof method. Therefore the content of the first argument is interpreted as a mathematical object rather than as an identifier as in the case of <code>\premise</code>. In our example, we specified that the simplification should take place on the right hand side of the equation. Other examples include proof methods that instantiate. Here we would indicate the substituted object in a <code>\justarg</code> macro.</p> <p>Note that both <code>\premise</code> and <code>\justarg</code> can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.</p>
<code>subproof (env.)</code>	The <code>spfcases</code> environment is used to mark up a subproof. This environment takes an optional <code>KeyVal</code> argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the <code>proof</code> environment). The <code>method</code> key can be used to give the name of the proof method executed to make this subproof.
<code>spfcases (env.)</code>	The <code>spfcases</code> environment is used to mark up a proof by cases. Technically it is a variant of the <code>subproof</code> where the <code>method</code> is <code>by-cases</code> . Its contents are <code>spfcases</code> environments that mark up the cases one by one.
<code>spfcases (env.)</code>	The content of a <code>spfcases</code> environment are a sequence of case proofs marked up in the <code>spfcases</code> environment, which takes an optional <code>KeyVal</code> argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a <code>spfcases</code> environment is the same as that of a <code>proof</code> , i.e. <code>spfsteps</code> , <code>spfcomments</code> , and <code>spfcases</code> environments.
<u><code>\spfcasesketch</code></u>	<code>\spfcasesketch</code> is a variant of the <code>spfcases</code> environment that takes the same arguments, but instead of the <code>spfsteps</code> in the body uses a third argument for a proof sketch.
<code>spfcomment (env.)</code>	The <code>spfcomment</code> environment is much like a <code>step</code> , only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a <code>\premise</code> .

`\sproofend` Traditionally, the end of a mathematical proof is marked with a little box at the end of the last line of the proof (if there is space and on the end of the next line if there isn't), like so:

The `stex-proofs` package provides the `\sproofend` macro for this.

`\sProofEndSymbol` If a different symbol for the proof end is to be used (e.g. *q.e.d*), then this can be obtained by specifying it using the `\sProofEndSymbol` configuration macro (e.g. by specifying `\sProofEndSymbol{q.e.d}`).

Some of the proof structuring macros above will insert proof end symbols for sub-proofs, in most cases, this is desirable to make the proof structure explicit, but sometimes this wastes space (especially, if a proof ends in a case analysis which will supply its own proof end marker). To suppress it locally, just set `proofend={}` in them or use `\sProofEndSymbol{}`.

5.3 Highlighting and Presentation Customizations

The environments starting with `s` (i.e. `smodule`, `sassertion`, `sexample`, `sdefinition`, `sparagraph` and `sproof`) by default produce no additional output whatsoever (except for the environment content of course). Instead, the document that uses them (whether directly or e.g. via `\inputref`) can decide how these environments are supposed to look like.

The `stexthm` package defines some default customizations that can be used, but of course many existing L^AT_EX templates come with their own `definition`, `theorem` and similar environments that authors are supposed (or even required) to use. Their concrete syntax however is usually not compatible with all the additional arguments that S_TE_X allows for semantic information.

Therefore we introduced the separate environments `sdefinition` etc. instead of using `definition` directly. We allow authors to specify how these environments should be styled via the commands `stexpatch*`.

`\stexpatchmodule`
`\stexpatchdefinition`
`\stexpatchassertion`
`\stexpatchexample`
`\stexpatchparagraph`
`\stexpatchproof`

All of these commands take one optional and two proper arguments, i.e. `\stexpatch* [<type>] {<begin-code>} {<end-code>}`.

After S_TE_X reads and processes the optional arguments for these environments, (some of) their values are stored in the macros `\s*<field>` (i.e. `sexampleid`, `\sassertionname`, etc.). It then checks for all the values `<type>` in the `type=`-list, whether an `\stexpatch* [<type>]` for the current environment has been called. If it finds one, it uses the patches `<begin-code>` and `<end-code>` to mark up the current environment. If no patch for (any of) the type(s) is found, it checks whether and `\stexpatch*` was called without optional argument.

For example, if we want to use a predefined `theorem` environment for `sassertions` with `type=theorem`, we can do

```
1 \stexpatchassertion[theorem]{\begin{theorem}}{\end{theorem}}
```

...or, rather, since e.g. `theorem`-like environments defined using `amsthm` take an optional title as argument, we can do:

```

1 \stexpatchassertion[theorem]
2   {\ifx\sassertiontitle\@empty
3     \begin{theorem}
4   \else
5     \begin{theorem}[\sassertiontitle]
6   \fi}
7 {\end{theorem}}

```

Or, if we want *all kinds of sdefinitions* to use a predefined definition-environment irrespective of their type=, then we can issue the following customization patch:

```

1 \stexpatchdefinition
2   {\ifx\sdefinitiontitle\@empty
3     \begin{definition}
4   \else
5     \begin{definition}[\sdefinitiontitle]
6   \fi}
7 {\end{definition}}

```

$\backslash\text{compemph}$ $\backslash\text{varemph}$ $\backslash\text{symrefemph}$ $\backslash\text{defemph}$	<p>Apart from the environments, we can control how \TeX highlights variables, notation components, $\backslash\text{symrefs}$ and $\backslash\text{definiendums}$, respectively.</p> <p>To do so, we simply redefine these four macros. For example, to highlight notation components (i.e. everything in a $\backslash\text{comp}$) in blue, as in this document, we can do $\backslash\text{def}\backslash\text{compemph}\#1\{\text{\textcolor{blue}\#1}\}$. By default, $\backslash\text{compemph}$ et al do nothing.</p>
--	---

$\backslash\text{compemph@uri}$ $\backslash\text{varemph@uri}$ $\backslash\text{symrefemph@uri}$ $\backslash\text{defemph@uri}$	<p>For each of the four macros, there exists an additional macro that takes the full URI of the relevant symbol currently being highlighted as a second argument. That allows us to e.g. use pdf tooltips and links. For example, this document uses⁵</p> <pre> 1 \protected\def\symrefemph@uri#1#2{ 2 \pdftooltip{ 3 \srefsymuri{#2}\symrefemph{#1}} 4 }{ 5 URI:~\detokenize{#2} 6 } 7 } </pre>
--	---

By default, $\backslash\text{compemph@uri}$ is simply defined as $\backslash\text{compemph}\{#1\}$ (analogously for the other three commands).

Chapter 6

Additional Packages

6.1 Tikzinput: Treating TIKZ code as images

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

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

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

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

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

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

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

6.2 Modular Document Structuring

6.2.1 Introduction

The document-structure package supplies an infrastructure for writing OMDOC documents in L^AT_EX. This includes a simple structure sharing mechanism for S_TE_X that allows to move from a copy-and-paste document development model to a copy-and-reference model, which conserves space and simplifies document management. The augmented structure can be used by MKM systems for added-value services, either directly from the S_TE_X sources, or after translation.

The document-structure package supplies macros and environments that allow to label document fragments and to reference them later in the same document or in other documents. In essence, this enhances the document-as-trees model to documents-as-directed-acyclic-graphs (DAG) model. This structure can be used by MKM systems for added-value services, either directly from the S_TE_X sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the S_TE_X collection.

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document source and the formatter does the copying during document formatting/presentation.

6.2.2 Package Options

The document-structure package accepts the following options:

<code>class=<name></code>	load <name>.cls instead of article.cls
<code>topsect=<sect></code>	The top-level sectioning level; the default for <sect> is section

6.2.3 Document Fragments

sfragment (*env.*) The structure of the document is given by nested **sfragment** environments. In the L^AT_EX route, the **sfragment** environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of **sfragment** environments. Correspondingly, the **sfragment** environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the sfragment. The optional metadata argument has the keys **id** for an identifier, **creators** and **contributors** for the Dublin Core metadata [DCM03]. The option **short** allows to give a short title for the generated section. If the title contains semantic macros, we need to give the **loadmodules** key (it needs no value). For instance we would have


```

1 \begin{smodule}{foo}
2   \symdef{bar}{Ba_r}
3   ...
4   \begin{sfragment}[id=sec.bardriv,loadmodules]
5     {Introducing $\protect\bar$ Derivations}

```

TeX automatically computes the sectioning level, from the nesting of `sfragment` environments.

But sometimes, we want to skip levels (e.g. to use a `\subsection*` as an introduction for a chapter).

`blindfragment` (*env.*) Therefore the `document-structure` package provides a variant `blindfragment` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindfragment` environment is useful e.g. for creating frontmatter at the correct level. The example below shows a typical setup for the outer document structure of a book with parts and chapters.

```

1 \begin{document}
2 \begin{blindfragment}
3 \begin{blindfragment}
4 \begin{frontmatter}
5 \maketitle\newpage
6 \begin{sfragment}{Preface}
7 ... <<preface>> ...
8 \end{sfragment}
9 \clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
10 \end{frontmatter}
11 \end{blindfragment}
12 ... <<introductory remarks>> ...
13 \end{blindfragment}
14 \begin{sfragment}{Introduction}
15 ... <<intro>> ...
16 \end{sfragment}
17 ... <<more chapters>> ...
18 \bibliographystyle{alpha}\bibliography{kwarc}
19 \end{document}

```

Here we use two levels of `blindfragment`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindfragment` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter³ and makes the preface of the book a section-level construct. The `frontmatter` environment also suppresses numbering as is traditional for prefaces.

`\skipfragment` The `\skipfragment` “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`.

³We shied away from redefining the `frontmatter` to induce a `blindfragment`, but this may be the “right” way to go in the future.

<hr/> <code>\currentsectionlevel</code> <code>\CurrentSectionLevel</code> <hr/>	The <code>\currentsectionlevel</code> macro supplies the name of the current sectioning level, e.g. “chapter”, or “subsection”. <code>\CurrentSectionLevel</code> is the capitalized variant. They are useful to write something like “In this <code>\currentsectionlevel</code> , we will...” in an <code>sfragment</code> environment, where we do not know which sectioning level we will end up.
--	--

6.2.4 Ending Documents Prematurely

<hr/> <code>\prematurestop</code> <code>\afterprematurestop</code> <hr/>	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.
---	---

`\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 [LMH].

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.

6.2.5 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.

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

<hr/> <code>\ifSGvar</code> <hr/>	With <code>\ifSGvar</code> we can test for the contents of a global variable: the macro call <code>\ifSGvar{⟨vname⟩}{⟨val⟩}{⟨cctx⟩}</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>⟨cctx⟩</code> is formatted.
-----------------------------------	---

6.3 Slides and Course Notes

6.3.1 Introduction

The `notesslides` document class is derived from `beamer.cls` [Tana], 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 `TeX` 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.

6.3.2 Package Options

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

<hr/> slides notes <hr/>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see subsection 6.3.3).
<hr/> sectocframes <hr/>	If the option <code>sectocframes</code> is given, then for the <code>sfragments</code> , special frames with the <code>sfragment</code> title (and number) are generated.
<hr/> frameimages fiboxed <hr/>	If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code> -generated frames (see). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.

6.3.3 Notes and Slides

- frame** (*env.*) Slides are represented with the `frame` environment just like in the `beamer` class, see [\[Tanb\]](#) 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 \LaTeX 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 `\notesttrue` or `\notesfalse` is strongly discouraged however.



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



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

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

`nparagraph (env.)` There are some environments that tend to occur at the top-level of `note` environments.

`nparagraph (env.)` We make convenience versions of these: e.g. the `nparagraph` environment is just an

`ndefinition (env.)` `sparagraph` inside a `note` environment (but looks nicer in the source, since it avoids one

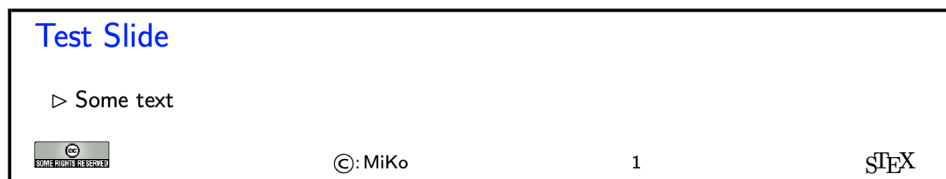
`nexample (env.)` level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,

`nsproof (env.)` `nsproof`, and `nassertion` environments.

`nassertion (env.)`

6.3.4 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 \LaTeX 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.

6.3.5 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 \LaTeX notes.

`\frameimage` In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.

The `\mhframeimage` macro is a variant of `\frameimage` with repository support. Instead of writing

```
1 \frameimage{\MathHub{fooMH/bar/source/baz/foobar}}
```


we can simply write (assuming that `\MathHub` is defined as above)

```
1 \mhframeimage[fooMH/bar]{baz/foobar}
```

Note that the `\mhframeimage` form is more semantic, which allows more advanced document management features in `MathHub`.

If `baz/foobar` is the “current module”, i.e. if we are on the `MathHub` path `...MathHub/fooMH/bar...`, then stating the repository in the first optional argument is redundant, so we can just use

```
1 \mhframeimage{baz/foobar}
```

`\textwarning` The `\textwarning` macro generates a warning sign: 

6.3.6 Excursions

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

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

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

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

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

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

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

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

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



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.

6.4 Representing Problems and Solutions

6.4.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: `hints`, `notes`, and `solutions`⁴. 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.

6.4.2 Problems and Solutions

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> option signifies that we are in

a test situation, so this option does not show the solutions (of course), but leaves space for the students to solve them.

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

Example 40

Input:

⁴for the moment multiple choice problems are not supported, but may well be in a future version

```

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}[for=elephants]
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:

Problem 6.4.1 (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle?

Hint: Think positively, this is simple!

Note: Justify your answer

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

Grading: if they do not give the justification deduct 5 pts

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

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

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

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

6.4.3 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` [*keyvals*] {*text*} takes an optional key/value argument *keyvals* for choice meta-data 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 `textt`. In solutions mode (we start the solutions in the code fragment below) we get

Example 41

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:

Problem 6.4.2 (Functions)

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

- ☐ `def`
Correct!
- ☐ `function`
Wrong! *that is for C and C++*
- ☐ `fun`
Wrong! *that is for Standard ML*
- ☐ `public static void`
Wrong! *that is for Java*

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

Example 42

Input:

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

Output:

Problem 6.4.3 (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).

6.4.4 Including Problems

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

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

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

6.5 Homeworks, Quizzes and Exams

6.5.1 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.

6.5.2 Package Options

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

6.5.3 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>	

6.5.4 Including Assignments

<hr/> <code>\inputassignment</code> <hr/>	The <code>\inputassignment</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.
---	---

6.5.5 Typesetting Exams

<code>\testspace</code>	<code>\testspace</code> takes an argument that expands to a dimension, and leaves vertical space
<code>\testnewpage</code>	accordingly. <code>\testnewpage</code> makes a new page in <code>test</code> mode, and <code>\testemptypage</code> gen-
<code>\testemptypage</code>	erates an empty page with the cautionary message that this page was intentionally left empty.
<code>testheading</code> (<i>env.</i>)	Finally, the <code>\testheading</code> 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.

```
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)

2022-06-30

You have one hour (sharp) for the test;

Write the solutions to the sheet.

The estimated time for solving this exam is 60 minutes, leaving you 0 minutes for revising your exam.

You can reach 40 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 13 points are bonus points.

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

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

	To be used for grading, do not write here											
prob.	6.4.1	6.4.2	6.4.3	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total	10			4	4	6	6	4	4	2	40	
reached												

good luck

⁶EDNOTE: MK: The first three “problems” come from the stex examples above, how do we get rid of this?

Part II

Documentation

Chapter 7

STEX-Basics

This sub package provides general set up code, auxiliary methods and abstractions for xhtml annotations.

7.1 Macros and Environments

<code>\sTeX</code>	Both print this ST _E X logo.
<code>\stex</code>	

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

Logs *<message>*, if the package option `debug` contains *<log-prefix>*.

7.1.1 HTML Annotations

<code>\if@latexml</code>	L ^A T _E X2e conditional for L ^A T _E X _{ML}
--------------------------	---

<code>\latexml_if_p: *</code>	L ^A T _E X3 conditionals for L ^A T _E X _{ML} .
<code>\latexml_if:TF *</code>	

<code>\stex_if_do_html_p: *</code>	Whether to currently produce any HTML annotations (can be false in some advanced structuring environments, for example)
<code>\stex_if_do_html:TF *</code>	

<code>\stex_suppress_html:n</code>	Temporarily disables HTML annotations in its argument code
------------------------------------	--

We have four macros for annotating generated HTML (via L^AT_EX_{ML} or R_US_TE_X) with attributes:

<code>\stex_annotate:nnn</code>	<code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code>
<code>\stex_annotate_invisible:nnn</code>	
<code>\stex_annotate_invisible:n</code>	

Annotates the HTML generated by $\langle content \rangle$ with

`property="stex:⟨property⟩", resource="⟨resource⟩".`

`\stex_annotate_invisible:n` adds the attributes

`stex:visible="false", style="display:none".`

`\stex_annotate_invisible:nnn` combines the functionality of both.

<code>stex_annotate_env (env.)</code>	<code>\begin{stex_annotate_env}{⟨property⟩}{⟨resource⟩}</code> $\langle content \rangle$ <code>\end{stex_annotate_env}</code> behaves like <code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code> .
---------------------------------------	---

7.1.2 Babel Languages

<code>\c_stex_languages_prop</code>
<code>\c_stex_language_abbrevs_prop</code>

Map language abbreviations to their full babel names and vice versa. e.g. `\c_stex_languages_prop{en}` yields `english`, and `\c_stex_language_abbrevs_prop{english}` yields `en`.

7.1.3 Auxiliary Methods

<code>\stex_deactivate_macro:Nn</code>	<code>\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}</code>
<code>\stex_reactivate_macro:N</code>	

Makes the macro $\langle cs \rangle$ throw an error, indicating that it is only allowed in the context of $\langle environments \rangle$.

`\stex_reactivate_macro:N⟨cs⟩` reactivates it again, i.e. this happens ideally in the $\langle begin \rangle$ -code of the associated environments.

<code>\ignorespacesandpars</code>	ignores white space characters and <code>\par</code> control sequences. Expands tokens in the process.
-----------------------------------	--

Chapter 8

STEX-MathHub

This sub package provides code for handling ST_EX archives, files, file paths and related methods.

8.1 Macros and Environments

<code>\stex_kpsewhich:n</code>	<code>\stex_kpsewhich:n</code> executes <code>kpsewhich</code> and stores the return in <code>\l_stex_kpsewhich_return_str</code> . This does not require shell escaping.
--------------------------------	---

8.1.1 Files, Paths, URIs

<code>\stex_path_from_string:Nn</code>	<code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{\langle string \rangle\}$ turns the $\langle string \rangle$ into a path by splitting it at <code>/</code> -characters and stores the result in $\langle path-variable \rangle$. Also applies <code>\stex_path_canonicalize:N</code> .
--	--

<code>\stex_path_to_string:NN</code>	The inverse; turns a path into a string and stores it in the second argument variable, or
<code>\stex_path_to_string:N</code>	leaves it in the input stream.

<code>\stex_path_canonicalize:N</code>	Canonicalizes the path provided; in particular, resolves <code>.</code> and <code>..</code> path segments.
--	--

<code>\stex_path_if_absolute_p:N</code> *	
<code>\stex_path_if_absolute:N\underline{TF}</code> *	

Checks whether the path provided is *absolute*, i.e. starts with an empty segment

<code>\c_stex_pwd_seq</code>	Store the current working directory as path-sequence and string, respectively, and the
<code>\c_stex_pwd_str</code>	(heuristically guessed) full path to the main file, based on the PWD and <code>\jobname</code> .
<code>\c_stex_mainfile_seq</code>	
<code>\c_stex_mainfile_str</code>	

<code>\g_stex_currentfile_seq</code>	The file being currently processed (respecting <code>\input</code> etc.)
--------------------------------------	--

<code>\stex_filestack_push:n</code>	Push and pop (repectively) a file path to the file stack, to keep track of the current file.
<code>\stex_filestack_pop:</code>	Are called in hooks <code>file/before</code> and <code>file/after</code> , respectively.

8.1.2 MathHub Archives

<code>\mathhub</code>	We determine the path to the local MathHub folder via one of four means, in order of
<code>\c_stex_mathhub_seq</code>	precedence:
<code>\c_stex_mathhub_str</code>	

1. The `mathhub` package option, or
2. the `\mathhub-macro`, if it has been defined before the `\usepackage{stex}`-statement, or
3. the `MATHHUB` system variable, or
4. a path specified in `~/.stex/mathhub.path`.

In all four cases, `\c_stex_mathhub_seq` and `\c_stex_mathhub_str` are set accordingly.

<code>\l_stex_current_repository_prop</code>
--

Always points to the *current* MathHub repository (if we currently are in one). Has the following fields corresponding to the entries in the `MANIFEST.MF`-file:

- `id`: The name of the archive, including its group (e.g. `smglom/calculus`),
- `ns`: The content namespace (for modules and symbols),
- `narr`: the narration namespace (for document references),
- `docurl`: The URL that is used as a basis for *external references*,
- `deps`: All archives that this archive depends on (currently not in use).

<code>\stex_set_current_repository:n</code>

Sets the current repository to the one with the provided ID. calls `__stex_mathhub_do_manifest:n`, so works whether this repository's `MANIFEST.MF`-file has already been read or not.

<code>\stex_require_repository:n</code>	Calls <code>__stex_mathhub_do_manifest:n</code> iff the corresponding archive property list does not already exist, and adds a corresponding definition to the <code>.sms</code> -file.
---	--

<code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{<i>repository-name</i>}{<i>code</i>}</code>
-------------------------------------	--

Change the current repository to `{repository-name}` (or not, if `{repository-name}` is empty), and passes its ID on to `{code}` as `#1`. Switches back to the previous repository after executing `{code}`.

8.1.3 Using Content in Archives

<hr/> <hr/> <code>\mhpath</code> *	<code>\mhpath{<archive-ID>}{<filename>}</code>
	Expands to the full path of file <code><filename></code> in repository <code><archive-ID></code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <code>\mhinput</code>	<code>\inputref[<archive-ID>]{<filename>}</code> Both <code>\input</code> the file <code><filename></code> in archive <code><archive-ID></code> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html-mode</code> , inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<hr/> <hr/> <code>\addmhbibresource</code>	<code>\inputref[<archive-ID>]{<filename>}</code> Adds a <code>.bib</code> -file <code><filename></code> in archive <code><archive-ID></code> (relative to the top-directory of the archive!).
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf-</code> archive of the current archive group(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant <code>lib</code> -folders.
<hr/> <hr/> <code>\libusepackage</code>	<code>\libusepackage[<args>]{<filename>}</code> Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<meta{args}>]{<Arg{filename}>}</code> instead of <code>\input</code> . Throws an error, if none or more than one suitable package file is found.
<hr/> <hr/> <code>\mhgraphics</code> <code>\cmhgraphics</code>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrepos</code> . It then resolves the file path in <code>\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source</code> -folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment.
<hr/> <hr/> <code>\lstinputmhlisting</code> <code>\clstinputmhlisting</code>	Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .

Chapter 9

STEX-References

This sub package contains code related to links and cross-references

9.1 Macros and Environments

<code>\STEXreftitle</code>	<code>\STEXreftitle{<some title>}</code>
----------------------------	--

Sets the title of the current document to *<some title>*. A reference to the current document from *some other* document will then be displayed accordingly. e.g. if `\STEXreftitle{foo book}` is called, then referencing Definition 3.5 in this document in another document will display **Definition 3.5 in foo book**.

<code>\stex_get_document_uri:</code>	Computes the current document uri from the current archive's narr -field and its location relative to the archive's source -directory. Reference targets are computed from this URI and the reference-id.
--------------------------------------	---

<code>\l_stex_current_docns_str</code>	Stores its result in <code>\l_stex_current_docns_str</code>
--	---

<code>\stex_get_document_url:</code>	Computes the current URL from the current archive's docurl -field and its location relative to the archive's source -directory. Reference targets are computed from this URL and the reference-id, if this document is only included in SMS mode.
--------------------------------------	---

<code>\l_stex_current_docurl_str</code>	Stores its result in <code>\l_stex_current_docurl_str</code>
---	--

9.1.1 Setting Reference Targets

<code>\stex_ref_new_doc_target:n</code>	<code>\stex_ref_new_doc_target:n{<id>}</code>
---	---

Sets a new reference target with id *<id>*.

<code>\stex_ref_new_sym_target:n</code>	<code>\stex_ref_new_sym_target:n{<uri>}</code>
---	--

Sets a new reference target for the symbol *<uri>*.

9.1.2 Using References

`\sref` `\sref[<opt-args>]{<id>}`

References the label with if *<id>*. Optional arguments: **TODO**

`\srefsym` `\srefsym[<opt-args>]{<symbol>}`

Like `\sref`, but references the *canonical label* for the provided symbol. The canonical target is the last of the following occurring in the document:

- A `\definiendum` or `\definame` for *<symbol>*,
- The `sassertion`, `sexample` or `sparagraph` with `for=<symbol>` that generated *<symbol>* in the first place, or
- A `\sparagraph` with `type=symdoc` and `for=<symbol>`.

`\srefsymuri` `\srefsymuri{<URI>}{<text>}`

A convenient short-hand for `\srefsym[linktext={<text>}] {<URI>}`, but requires the first argument to be a full URI already. Intended to be used in e.g. `\compemph@uri`, `\defemph@uri`, etc.

Chapter 10

sTeX-Modules

This sub package contains code related to Modules

10.1 Macros and Environments

The content of a module with uri $\langle <URI> \rangle$ is stored in four macros. All modifications of these macros are global:

<hr/> <u>$\backslash c_stex_module_<URI>_prop$</u> <hr/>	A property list with the following fields: <ul style="list-style-type: none">name The <i>name</i> of the module,ns the <i>namespace</i> in field ns,file the <i>file</i> containing the module, as a sequence of path fragmentslang the module's <i>language</i>,sig the language of the signature module, if the current file is a translation from some other language,deprecate if this module is deprecated, the module that replaces it,meta the metatheory of the module.
---	---

<hr/> <u>$\backslash c_stex_module_<URI>_code$</u> <hr/>	The code to execute when this module is activated (i.e. imported), e.g. to set all the semantic macros, notations, etc.
---	---

<hr/> <u>$\backslash c_stex_module_<URI>_constants$</u> <hr/>	The names of all constants declared in the module
--	---

<hr/> <u>$\backslash c_stex_module_<URI>_constants$</u> <hr/>	The full URIs of all modules imported in this module
--	--

<code>\l_stex_current_module_str</code>	<code>\l_stex_current_module_str</code> always contains the URI of the current module (if existent).
---	--

<code>\l_stex_all_modules_seq</code>	Stores full URIs for all modules currently in scope.
--------------------------------------	--

<code>\stex_if_in_module_p: *</code>	Conditional for whether we are currently in a module
<code>\stex_if_in_module:TF *</code>	

<code>\stex_if_module_exists_p:n *</code>	
<code>\stex_if_module_exists:nTF *</code>	

Conditional for whether a module with the provided URI is already known.

<code>\stex_add_to_current_module:n</code>	
<code>\STEXexport</code>	

Adds the provided tokens to the `_code` control sequence of the current module.

`\stex_add_to_current_module:n` is used internally, `\STEXexport` is intended for users and additionally executes the provided code immediately.

<code>\stex_add_constant_to_current_module:n</code>	
---	--

Adds the declaration with the provided name to the `_constants` control sequence of the current module.

<code>\stex_add_import_to_current_module:n</code>	
---	--

Adds the module with the provided full URI to the `_imports` control sequence of the current module.

<code>\stex_collect_imports:n</code>	Iterates over all imports of the provided (full URI of a) module and stores them as a topologically sorted list – including the provided module as the last element – in <code>\l_stex_collect_imports_seq</code>
--------------------------------------	---

<code>\stex_do_up_to_module:n</code>	Code that is <i>exported</i> from module (such as symbol declarations) should be local <i>to the current module</i> . For that reason, ideally all symbol declarations and similar commands should be called directly in the module environment, however, that is not always feasible, e.g. in structural features or <code>sparapraphs</code> . <code>\stex_do_up_to_module</code> therefore executes the provided code repeatedly in an <code>\aftergroup</code> up until the group level is equal to that of the innermost smodule environment.
--------------------------------------	--

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

If the current file is `.../source/sub/file.tex` in some archive with namespace `http://some.namespace/foo`, then the namespace of is `http://some.namespace/foo/sub/file`. Otherwise, the namespace is the absolute file path of the current file (i.e. starting with `file:///`).

The result is stored in `\l_stex_module_ns_str`. Additionally, the sub path relative to the current repository is stored in `\l_stex_module_subpath_str`.

10.1.1 The `smodule` environment

`module (env.) \begin{module}[\langle options \rangle]{\langle name \rangle}`

Opens a new module with name `\langle name \rangle`. Options are:

`title` (`\langle token list \rangle`) to display in customizations.

`type` (`\langle string \rangle*`) for use in customizations.

`deprecate` (`\langle module \rangle`) if set, will throw a warning when loaded, urging to use `\langle module \rangle` instead.

`id` (`\langle string \rangle`) for cross-referencing.

`ns` (`\langle URI \rangle`) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed using `\stex_modules_current_namespace:`.

`lang` (`\langle language \rangle`) if not set, computed from the current file name (e.g. `foo.en.tex`).

`sig` (`\langle language \rangle`) 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.

`creators` (`\langle string \rangle*`) names of the creators.

`contributors` (`\langle string \rangle*`) names of contributors.

`srccite` (`\langle string \rangle`) a source citation for the content of this module.

`\stex_module_setup:nn \stex_module_setup:nn{\langle params \rangle}{\langle name \rangle}`

Sets up a new module with name `\langle name \rangle` and optional parameters `\langle params \rangle`. In particular, sets `\l_stex_current_module_str` appropriately.

`\stexpatchmodule \stexpatchmodule [\langle type \rangle] {\langle begincode \rangle} {\langle endcode \rangle}`

Customizes the presentation for those `smodule`-environments with `type=\langle type \rangle`, or all others if no `\langle type \rangle` is given.

`\STEXModule \STEXModule {\langle fragment \rangle}`

Attempts to find a module whose URI ends with `\langle fragment \rangle` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n` Invoked by `\STEXModule`. Needs to be followed either by `!\macro` or `?{\langle symbolname \rangle}`. In the first case, it stores the full URI in `\macro`; in the second case, it invokes the symbol `\langle symbolname \rangle` in the selected module.

<code>\stex_activate_module:n</code>	Activate the module with the provided URI; i.e. executes all macro code of the module's <code>_code</code> -macro (does nothing if the module is already activated in the current context) and adds the module to <code>\l_stex_all_modules_seq</code> .
--------------------------------------	--

Chapter 11

STEX-Module Inheritance

Code related to Module Inheritance, in particular *sms mode*.

11.1 Macros and Environments

11.1.1 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T_EX commands not explicitly allowed via the following lists – all of which either declare module content or are needed in order to declare module content:

`\g_stex_smsmode_allowedmacros_tl`

Macros that are executed as is; i.e. sms mode continues immediately after. These macros may not take any arguments or otherwise gobble tokens.

Initially: `\makeatletter`, `\makeatother`, `\ExplSyntaxOn`, `\ExplSyntaxOff`.

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed and potentially gobble up further tokens. These macros need to make sure, that the very last token they ultimately expand to is `\stex_smsmode_do:`.

Initially: `\symdecl`, `\notation`, `\symdef`, `\importmodule`, `\STEXexport`, `\inlineass`, `\inlinedef`, `\inlineex`, `\endinput`, `\setnotation`, `\copynotation`.

`\g_stex_smsmode_allowedenvs_seq`

The names of environments that should be allowed in SMS mode. The corresponding `\begin`-statements are treated like the macros in `\g_stex_smsmode_allowedmacros_escape_tl`, so `\stex_smsmode_do:` needs to be the last token in the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called directly and sms mode continues afterwards.

Initially: `smodule`, `copymodule`, `interpretmodule`, `sdefinition`, `sexample`, `sassertion`, `sparagraph`.

`\stex_if_smsmode_p:` ★ Tests whether SMS mode is currently active.
`\stex_if_smsmode:` *TF* ★

<code>\stex_file_in_smsmode:nn</code>	<code>\stex_in_smsmode:nn {<filename>} {<code>}</code>
---------------------------------------	--

Executes `<code>` in SMS mode, followed by the content of `<filename>`. `<code>` can be used e.g. to set the current repository, and is executed within a new tex group, and the same group as the file content.

<code>\stex_smsmode_do:</code>	Starts gobbling tokens until one is encountered that is allowed in SMS mode.
--------------------------------	--

11.1.2 Imports and Inheritance

<code>\importmodule</code>	<code>\importmodule[<archive-ID>]{<module-path>}</code>
----------------------------	---

Imports a module by reading it from a file and “activating” it. `\STEX` determines the module and its containing file by passing its arguments on to `\stex_import_module_path:nn`.

<code>\usemodule</code>	<code>\importmodule[<archive-ID>]{<module-path>}</code>
-------------------------	---

Like `\importmodule`, but does not export its contents; i.e. including the current module will not activate the used module

$\backslash\text{stex_import_module_uri:nn}$	$\backslash\text{stex_import_module_uri:nn } \{ \langle \text{archive-ID} \rangle \} \{ \langle \text{module-path} \rangle \}$
---	---

Determines the URI of a module by splitting $\langle \text{module-path} \rangle$ into $\langle \text{path} \rangle ? \langle \text{name} \rangle$. If $\langle \text{module-path} \rangle$ does *not* contain a ?-character, we consider it to be the $\langle \text{name} \rangle$, and $\langle \text{path} \rangle$ to be empty.

If $\langle \text{archive-ID} \rangle$ is empty, it is automatically set to the ID of the current archive (if one exists).

1. If $\langle \text{archive-ID} \rangle$ is empty:

(a) If $\langle \text{path} \rangle$ is empty, then $\langle \text{name} \rangle$ must have been declared earlier in the same file and retrievable from $\backslash\text{g_stex_modules_in_file_seq}$, or a file with name $\langle \text{name} \rangle . \langle \text{lang} \rangle . \text{tex}$ must exist in the same folder, containing a module $\langle \text{name} \rangle$.

That module should have the same namespace as the current one.

(b) If $\langle \text{path} \rangle$ is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

(a) If $\langle \text{path} \rangle$ is empty, then $\langle \text{name} \rangle$ must have been declared earlier in the same file and retrievable from $\backslash\text{g_stex_modules_in_file_seq}$, or a file with name $\langle \text{name} \rangle . \langle \text{lang} \rangle . \text{tex}$ must exist in the top **source** folder of the archive, containing a module $\langle \text{name} \rangle$.

That module should lie directly in the namespace of the archive.

(b) If $\langle \text{path} \rangle$ is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive.

If a module by that namespace exists, it is returned. Otherwise, we call $\backslash\text{stex_require_module:nn}$ on the **source** directory of the archive to find the file.

$\backslash\text{l_stex_import_name_str}$ $\backslash\text{l_stex_import_archive_str}$ $\backslash\text{l_stex_import_path_str}$ $\backslash\text{l_stex_import_ns_str}$	stores the result in these four variables.
---	--

$\backslash\text{stex_import_require_module:nnnn}$	$\{ \langle \text{ns} \rangle \} \{ \langle \text{archive-ID} \rangle \} \{ \langle \text{path} \rangle \} \{ \langle \text{name} \rangle \}$
---	---

Checks whether a module with URI $\langle \text{ns} \rangle ? \langle \text{name} \rangle$ already exists. If not, it looks for a plausible file that declares a module with that URI.

Finally, activates that module by executing its `_code`-macro.

Chapter 12

STEX-Symbols

Code related to symbol declarations and notations

12.1 Macros and Environments

$\backslash\text{symdecl}$	$\backslash\text{symdecl}\{\langle\text{macroname}\rangle\}[\langle\text{args}\rangle]$
----------------------------	---

Declares a new symbol with semantic macro $\backslash\text{macroname}$. Optional arguments are:

- **name**: An (OMDOC) name. By default equal to $\langle\text{macroname}\rangle$.
- **type**: An (ideally semantic) term, representing a *type*. Not used by STEX, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by STEX, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:
 - i** a “normal” argument, e.g. $\backslash\text{symdecl}\{\text{plus}\}[\text{args}=\text{ii}]$ allows for $\backslash\text{plus}\{2\}\{2\}$.
 - a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. $\backslash\text{symdecl}\{\text{plus}\}[\text{args}=\text{a}]$ allows for $\backslash\text{plus}\{2,2,2\}$.
 - b** a *variable* argument. Is treated by STEX like an *i*-argument, but an application is turned into an **OMBind** in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g. $\backslash\text{symdecl}\{\text{forall}\}[\text{args}=\text{bi}]$ allows for $\backslash\text{forall}\{x\in\text{Nat}\}\{x\geq 0\}$.

<hr/> <hr/> <code>\stex_symdecl_do:n</code>	<p>Implements the core functionality of <code>\symdecl</code>, and is called by <code>\symdecl</code> and <code>\symdef</code>. Ultimately stores the symbol $\langle URI \rangle$ in the property list <code>\l_stex_symdecl_<URI>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>name</code> (string), • <code>module</code> (string), • <code>notations</code> (sequence of strings; initially empty), • <code>local</code> (boolean), • <code>type</code> (token list), • <code>args</code> (string of <code>is</code>, <code>as</code> and <code>bs</code>), • <code>arity</code> (integer string), • <code>assoc</code>s (integer string; number of associative arguments),
<hr/> <hr/> <code>\stex_all_symbols:n</code>	Iterates over all currently available symbols. Requires two <code>\seq_map_break:</code> to break fully.
<hr/> <hr/> <code>\stex_get_symbol:n</code>	Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...
<hr/> <hr/> <code>\notation</code>	<p><code>\notation[<args>]{<symbol>}{<notations⁺>}</code> Introduces a new notation for $\langle symbol \rangle$, see <code>\stex_notation_do:nn</code></p>
<hr/> <hr/> <code>\stex_notation_do:nn</code>	<p><code>\stex_notation_do:nn{<URI>}{<notations⁺>}</code> Implements the core functionality of <code>\notation</code>, and is called by <code>\notation</code> and <code>\symdef</code>. Ultimately stores the notation in the property list <code>\g_stex_notation_<URI>#<variant>#<lang>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>symbol</code> (URI string), • <code>language</code> (string), • <code>variant</code> (string), • <code>opprec</code> (integer string), • <code>argprec</code>s (sequence of integer strings)
<hr/> <hr/> <code>\symdef</code>	<p><code>\symdef[<args>]{<symbol>}{<notations⁺>}</code> Combines <code>\symdecl</code> and <code>\notation</code> by introducing a new symbol and assigning a new notation for it.</p>

Chapter 13

STEX-Terms

Code related to symbolic expressions, typesetting notations, notation components, etc.

13.1 Macros and Environments

`\STEXsymbol` Uses `\stex_get_symbol:n` to find the symbol denoted by the first argument and passes the result on to `\stex_invoke_symbol:n`

`\symref` `\symref{<symbol>}{<text>}`
 shortcut for `\STEXsymbol{<symbol>}! [<text>]`

`\stex_invoke_symbol:n` Executes a semantic macro. Outside of math mode or if followed by `*`, it continues to `\stex_term_custom:nn`. In math mode, it uses the default or optionally provided notation of the associated symbol.

If followed by `!`, it will invoke the symbol *itself* rather than its application (and continue to `\stex_term_custom:nn`), i.e. it allows to refer to `\plus![addition]` as an operation, rather than `\plus[addition of]{some}{terms}`.

`\STEXInternalTermMathOMSiiii` `<URI><fragment><precedence><body>`
`\STEXInternalTermMathOMAiiii`
`\STEXInternalTermMathOMBiiii`

Annotates `<body>` as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol `<URI>`, generated by the specific notation `<fragment>` with (upwards) operator precedence `<precedence>`. Inserts parentheses according to the current downwards precedence and operator precedence.

`\STEXInternalTermMathArgiii` `\stex_term_arg:nnn<int><prec><body>`

Annotates `<body>` as the `<int>`th argument of the current OMA or OMBIND, with (downwards) argument precedence `<prec>`.

<hr/> <code>\STEXInternalTermMathAssocArgiiii</code> <hr/>	<code>\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨body⟩</code>
	Annotates $\langle body \rangle$ as the $\langle int \rangle$ th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$ and associative notation $\langle notation \rangle$.
<hr/> <code>\infpref</code> <code>\neginfpref</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<code>\dobrackets {⟨body⟩}</code> Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \TeX brackets (by default (and)), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <code>\withbrackets</code> <hr/>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code> Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \TeX for automated bracketing (by default (and)) to $\langle left \rangle$ and $\langle right \rangle$. Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <code>\stex_term_custom:nn</code> <hr/>	<code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code> Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .
<hr/> <code>\comp</code> <code>\compemph</code> <code>\compemph@uri</code> <code>\defemph</code> <code>\defemph@uri</code> <code>\symrefemph</code> <code>\symrefemph@uri</code> <code>\varemp</code> <code>\varemp@uri</code> <hr/>	<code>\comp{⟨args⟩}</code> Marks $\langle args \rangle$ as a notation component of the current symbol for highlighting, linking, etc. The precise behavior is governed by <code>\@comp</code> , which takes as additional argument the URI of the current symbol. By default, <code>\@comp</code> adds the URI as a PDF tooltip and colors the highlighted part in blue. <code>\@defemph</code> behaves like <code>\@comp</code> , and can be similarly redefined, but marks an expression as <i>definiendum</i> (used by <code>\definiendum</code>)
<hr/> <code>\STEXinvisible</code> <hr/>	Exports its argument as OMDOC (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.
<hr/> <code>\ellipses</code> <hr/>	TODO

Chapter 14

TeX-Structural Features

Code related to structural features

14.1 Macros and Environments

14.1.1 Structures

`mathstructure` (*env.*) TODO

Chapter 15

sTeX-Statements

Code related to statements, e.g. definitions, theorems

15.1 Macros and Environments

`symboldoc (env.) \begin{<symboldoc>}{<symbols>} <text> \end{<symboldoc>}`

Declares *<text>* to be a (natural language, encyclopaedic) description of $\{<symbols>\}$ (a comma separated list of symbol identifiers).

Chapter 16

sTeX-Proofs: Structural Markup for Proofs

Chapter 17

sTeX-Metatheory

17.1 Symbols

Part III
Extensions

Chapter 18

Tikzinput: Treating TIKZ code as images

18.1 Macros and Environments

Chapter 19

document-structure: Semantic Markup for Open Mathematical Documents in L^AT_EX

Chapter 20

NotesSlides – Slides and Course Notes

Chapter 21

`problem.sty`: An Infrastructure for formatting Problems

Chapter 22

**hwexam.sty/cls: An
Infrastructure for formatting
Assignments and Exams**

Part IV
Implementation

Chapter 23

\TeX -Basics Implementation

23.1 The \TeX Document Class

The `stex` document class is pretty straight-forward: It largely extends the `standalone` package and loads the `stex` package, passing all provided options on to the package.

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/05/24}{3.1.0}{sTeX document class}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \bool_set_true:N \c_stex_document_class_bool
12
13 \RequirePackage{stex}
14
15 \stex_html_backend:TF {
16   \LoadClass{article}
17 }{
18   \LoadClass[border=1px,varwidth,crop=false]{standalone}
19   \setlength\textwidth{15cm}
20 }
21 \RequirePackage{standalone}
22
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```

```

31     }
32   }
33   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
34   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
35     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
36     \prop_if_in:NoT \c_stex_languages_prop \l_tmpa_str {
37       \stex_debug:nn{language} {Language~\l_tmpa_str~
38         inferred~from~file~name}
39     }
40   }
41 }
42 }
43 </cls>

```

23.2 Preliminaries

```

44 <*package>
45
46 %%%%%%%%% basics.dtx %%%%%%%%%
47
48 \RequirePackage{expl3,l3keys2e,ltxcmds}
49 \ProvidesExplPackage{stex}{2022/05/24}{3.1.0}{sTeX package}
50
51 \bool_if_exist:NF \c_stex_document_class_bool {
52   \bool_set_false:N \c_stex_document_class_bool
53   \RequirePackage{standalone}
54 }
55
56 \message{^^J*~This~is~sTeX~version~3.1.0~*^^J}
57
58 %\RequirePackage{morewrites}
59 %\RequirePackage{amsmath}
60
61 Package options:
62 \keys_define:nn { stex } {
63   debug      .clist_set:N = \c_stex_debug_clist ,
64   lang       .clist_set:N = \c_stex_languages_clist ,
65   mathhub    .tl_set_x:N  = \mathhub ,
66   usesms     .bool_set:N  = \c_stex_persist_mode_bool ,
67   writesms   .bool_set:N  = \c_stex_persist_write_mode_bool ,
68   image      .bool_set:N  = \c_tikzinput_image_bool ,
69   unknown    .code:n      = {}
70 }
71 \ProcessKeysOptions { stex }

```

\stex The sTeX logo:

\sTeX `\RequirePackage{stex-logo}` % externalized for backwards-compatibility reasons

(End definition for `\stex` and `\sTeX`. These functions are documented on page 64.)

23.3 Messages and logging

```

72 <@@=stex_log>
    Warnings and error messages
73 \msg_new:nnn{stex}{error/unknownlanguage}{
74   Unknown~language:~#1
75 }
76 \msg_new:nnn{stex}{warning/nomathhub}{
77   MATHHUB~system~variable~not~found~and~no~
78   \detokenize{\mathhub}~value~set!
79 }
80 \msg_new:nnn{stex}{error/deactivated-macro}{
81   The~\detokenize{#1}~command~is~only~allowed~in~#2!
82 }

```

\stex_debug:nn A simple macro issuing package messages with subpath.

```

83 \cs_new_protected:Nn \stex_debug:nn {
84   \clist_if_in:NnTF \c_stex_debug_clist { all } {
85     \msg_set:nnn{stex}{debug / #1}{
86       \\Debug~#1:~#2\\
87     }
88     \msg_none:nn{stex}{debug / #1}
89   }{
90     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
91       \msg_set:nnn{stex}{debug / #1}{
92         \\Debug~#1:~#2\\
93       }
94       \msg_none:nn{stex}{debug / #1}
95     }
96   }
97 }

```

(End definition for \stex_debug:nn. This function is documented on page 64.)

Redirecting messages:

```

98 \clist_if_in:NnTF \c_stex_debug_clist {all} {
99   \msg_redirect_module:nnn{ stex }{ none }{ term }
100 }{
101   \clist_map_inline:Nn \c_stex_debug_clist {
102     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
103   }
104 }
105
106 \stex_debug:nn{log}{debug~mode~on}

```

23.4 HTML Annotations

```

107 <@@=stex_annotate>

```

\l_stex_html_arg_tl Used by annotation macros to ensure that the HTML output to annotate is not empty.
\c_stex_html_emptyarg_tl

```

108 \tl_new:N \l_stex_html_arg_tl

```

(End definition for \l_stex_html_arg_tl and \c_stex_html_emptyarg_tl. These variables are documented on page ??.)

`_stex_html_checkempty:n`

```

109 \cs_new_protected:Nn \_stex_html_checkempty:n {
110   \tl_set:Nn \l_stex_html_arg_tl { #1 }
111   \tl_if_empty:NT \l_stex_html_arg_tl {
112     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
113   }
114 }

```

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

`\stex_if_do_html_p:` Whether to (locally) produce HTML output

`\stex_if_do_html:TF`

```

115 \bool_new:N \_stex_html_do_output_bool
116 \bool_set_true:N \_stex_html_do_output_bool
117
118 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
119   \bool_if:nTF \_stex_html_do_output_bool
120     \prg_return_true: \prg_return_false:
121 }

```

(End definition for `\stex_if_do_html:TF`. This function is documented on page 64.)

`\stex_suppress_html:n` Whether to (locally) produce HTML output

```

122 \cs_new_protected:Nn \stex_suppress_html:n {
123   \exp_args:Nne \use:nn {
124     \bool_set_false:N \_stex_html_do_output_bool
125     #1
126   }{
127     \stex_if_do_html:T {
128       \bool_set_true:N \_stex_html_do_output_bool
129     }
130   }
131 }

```

(End definition for `\stex_suppress_html:n`. This function is documented on page 64.)

`\stex_annotate_env:nn`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EX_ML, R_US_ETEX, p_DF_LA_TE_X).

The p_DF_LA_TE_X-macros largely do nothing; the R_US_ETEX-implementations are pretty clear in what they do, the L^AT_EX_ML-implementations resort to perl bindings.

```

132 \tl_if_exist:NF\stex@backend{
133   \ifcsname if@rustex\endcsname
134   \def\stex@backend{rustex}
135   \else
136     \ifcsname if@latexml\endcsname
137     \def\stex@backend{latexml}
138     \else
139     \def\stex@backend{pdflatex}
140     \fi
141   \fi
142 }
143 \input{stex-backend-\stex@backend.cfg}
144
145 \newif\ifstexhtml
146 \stex_html_backend:TF\stexhtmltrue\stexhtmlfalse
147

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 65.)

23.5 Babel Languages

148 `<@=stex_language>`

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

149 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
150   en = english ,
151   de = ngerman ,
152   ar = arabic ,
153   bg = bulgarian ,
154   ru = russian ,
155   fi = finnish ,
156   ro = romanian ,
157   tr = turkish ,
158   fr = french
159 }}
160
161 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
162   english   = en ,
163   ngerman   = de ,
164   arabic    = ar ,
165   bulgarian = bg ,
166   russian   = ru ,
167   finnish   = fi ,
168   romanian  = ro ,
169   turkish   = tr ,
170   french    = fr
171 }}
172 % todo: chinese simplified (zhs)
173 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 65.)

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

```

174 \cs_new_protected:Nn \stex_set_language:Nn {
175   \str_set:Nx \l_tmpa_str {#2}
176   \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {
177     \ifx\@onlypreamble\@notprerr
178       \ltx@ifpackageloaded{babel}{
179         \exp_args:No \selectlanguage #1
180       }{}
181     \else
182       \exp_args:No \str_if_eq:nnTF #1 {turkish} {
183         \RequirePackage[#1,shorthands=:!]{babel}
184       }{
185         \RequirePackage[#1]{babel}
186       }
187     \fi
188   }
189 }
190

```

```

191 \clist_if_empty:NF \c_stex_languages_clist {
192   \bool_set_false:N \l_tmpa_bool
193   \clist_clear:N \l_tmpa_clist
194   \clist_map_inline:Nn \c_stex_languages_clist {
195     \str_set:Nx \l_tmpa_str {#1}
196     \str_if_eq:nnT {#1}{tr}{
197       \bool_set_true:N \l_tmpa_bool
198     }
199     \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
200       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
201     } {
202       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
203     }
204   }
205   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
206   \bool_if:NTF \l_tmpa_bool {
207     \RequirePackage[\clist_use:Nn \l_tmpa_clist,,shorthands=:!]{babel}
208   }{
209     \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
210   }
211 }
212
213 \AtBeginDocument{
214   \stex_html_backend:T {
215     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
216     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
217     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
218     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
219     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
220       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
221       \stex_debug:nn{basics} {Language~\l_tmpa_str~
222         inferred~from~file~name}
223       \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
224     }
225   }
226 }

```

23.6 Persistence

```

227 <@@=stex_persist>
228 \bool_if:NTF \c_stex_persist_mode_bool {
229   \def \stex_persist:n #1 {}
230   \def \stex_persist:x #1 {}
231 }{
232   \bool_if:NTF \c_stex_persist_write_mode_bool {
233     \iow_new:N \c__stex_persist_iow
234     \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
235     \AtEndDocument{
236       \iow_close:N \c__stex_persist_iow
237     }
238     \cs_new_protected:Nn \stex_persist:n {
239       \tl_set:Nn \l_tmpa_tl { #1 }
240       \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl

```



```

241 \regex_replace_all:nnN { \ } { \~ } \l_tmpa_tl
242 \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
243 }
244 \cs_generate_variant:Nn \stex_persist:n {x}
245 }{
246 \def \stex_persist:n #1 {}
247 \def \stex_persist:x #1 {}
248 }
249 }

```

23.7 Auxiliary Methods

\stex_deactivate_macro:Nn

```

250 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
251 \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
252 \def#1{
253 \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
254 }
255 }

```

(End definition for \stex_deactivate_macro:Nn. This function is documented on page 65.)

\stex_reactivate_macro:N

```

256 \cs_new_protected:Nn \stex_reactivate_macro:N {
257 \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
258 }

```

(End definition for \stex_reactivate_macro:N. This function is documented on page 65.)

\ignorespacesandpars

```

259 \protected\def\ignorespacesandpars{
260 \begingroup\catcode13=10\relax
261 \@ifnextchar\par{
262 \endgroup\expandafter\ignorespacesandpars\@gobble
263 }{
264 \endgroup
265 }
266 }
267
268 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
269 \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
270 \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
271 \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
272
273 \tl_clear:N \_tmp_args_tl
274 \int_step_inline:nn \l_tmpa_int {
275 \tl_put_right:Nx \_tmp_args_tl {\exp_not:n{####}\exp_not:n{##1}}
276 }
277
278 \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
279 \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
280 \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
281 \exp_after:wN\exp_after:wN\exp_after:wN {
282 \exp_after:wN #2 \_tmp_args_tl

```

```

283     }
284   }}
285 }
286 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
287 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
288 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
289
290 \cs_new_protected:Nn \stex_copy_control_sequence_ii:NNN {
291   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
292   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
293   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
294
295   \tl_clear:N \_tmp_args_tl
296   \int_step_inline:nn \l_tmpa_int {
297     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{#####}\exp_not:n{##1}}}
298   }
299
300   \edef \_tmp_args_tl {
301     \exp_after:wN \exp_after:wN \exp_after:wN \exp_not:n
302     \exp_after:wN \exp_after:wN \exp_after:wN {
303       \exp_after:wN #2 \_tmp_args_tl
304     }
305   }
306
307   \exp_after:wN \def \exp_after:wN \_tmp_args_tl
308   \exp_after:wN ## \exp_after:wN 1 \exp_after:wN ## \exp_after:wN 2
309   \exp_after:wN { \_tmp_args_tl }
310
311   \edef \_tmp_args_tl {
312     \exp_after:wN \exp_not:n \exp_after:wN {
313       \_tmp_args_tl {####1}{####2}
314     }
315   }
316
317   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
318   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
319     \exp_after:wN \exp_not:n \exp_after:wN {\_tmp_args_tl}
320   }}
321 }
322
323 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {cNN}
324 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {NcN}
325 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {ccN}

```

(End definition for `\ignorespacesandpars`. This function is documented on page 65.)

`\MMTrule`

```

326 \NewDocumentCommand \MMTrule {m m}{
327   \seq_set_split:Nnn \l_tmpa_seq , {#2}
328   \int_zero:N \l_tmpa_int
329   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
330     \seq_if_empty:NF \l_tmpa_seq {
331       $\seq_map_inline:Nn \l_tmpa_seq {
332         \int_incr:N \l_tmpa_int

```

```

333         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
334     }$
335 }
336 }
337 }
338
339 \NewDocumentCommand \MMTinclude {m}{
340     \stex_annotate_invisible:nnn{import}{#1}{ }
341 }
342
343 \tl_new:N \g_stex_document_title
344 \cs_new_protected:Npn \STEXTtitle #1 {
345     \tl_if_empty:NT \g_stex_document_title {
346         \tl_gset:Nn \g_stex_document_title { #1 }
347     }
348 }
349 \cs_new_protected:Nn \stex_document_title:n {
350     \tl_if_empty:NT \g_stex_document_title {
351         \tl_gset:Nn \g_stex_document_title { #1 }
352         \stex_annotate_invisible:n{\noindent
353             \stex_annotate:nnn{doctitle}{ }{ #1 }
354         \par}
355     }
356 }
357 \AtBeginDocument {
358     \let \STEXTtitle \stex_document_title:n
359     \tl_if_empty:NF \g_stex_document_title {
360         \stex_annotate_invisible:n{\noindent
361             \stex_annotate:nnn{doctitle}{ }{ \g_stex_document_title }
362         \par}
363     }
364     \let \stex_maketitle \maketitle
365     \def \maketitle {
366         \tl_if_empty:NF \@title {
367             \exp_args:No \stex_document_title:n \@title
368         }
369         \stex_maketitle:
370     }
371 }
372
373 \cs_new_protected:Nn \stex_par: {
374     \mode_if_vertical:F{
375         \if@minipage\else\if@nobreak\else\par\fi\fi
376     }
377 }
378
379 \end{package}

```

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

Chapter 24

STEX -MathHub Implementation

```
380 <*package>
381
382 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
383
384 <@@=stex_path>
385
386 Warnings and error messages
387 \msg_new:nnn{stex}{error/norepository}{
388   No~archive~#1~found~in~#2
389 }
390 \msg_new:nnn{stex}{error/notinarchive}{
391   Not~currently~in~an~archive,~but~\detokenize{#1}~
392   needs~one!
393 }
394 \msg_new:nnn{stex}{error/nofile}{
395   \detokenize{#1}~could~not~find~file~#2
396 }
397 \msg_new:nnn{stex}{error/twofiles}{
398   \detokenize{#1}~found~two~candidates~for~#2
399 }
```

24.1 Generic Path Handling

We treat paths as L^AT_EX3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

`\stex_path_from_string:Nn`

```
398 \cs_new_protected:Nn \stex_path_from_string:Nn {
399   \str_set:Nx \l_tmpa_str { #2 }
400   \str_if_empty:NTF \l_tmpa_str {
401     \seq_clear:N #1
402   }{
403     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
404     \sys_if_platform_windows:T{
405       \seq_clear:N \l_tmpa_tl
```

```

406     \seq_map_inline:Nn #1 {
407       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
408       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
409     }
410     \seq_set_eq:NN #1 \l_tmpa_tl
411   }
412   \stex_path_canonicalize:N #1
413 }
414 }
415

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 66.)

`\stex_path_to_string:NN`
`\stex_path_to_string:N`

```

416 \cs_new_protected:Nn \stex_path_to_string:NN {
417   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
418 }
419
420 \cs_new:Nn \stex_path_to_string:N {
421   \seq_use:Nn #1 /
422 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 66.)

`\c__stex_path_dot_str` . and .., respectively.
`\c__stex_path_up_str`

```

423 \str_const:Nn \c__stex_path_dot_str {.}
424 \str_const:Nn \c__stex_path_up_str {...}

```

(End definition for `\c__stex_path_dot_str` and `\c__stex_path_up_str`.)

`\stex_path_canonicalize:N` Canonicalizes the path provided; in particular, resolves . and .. path segments.

```

425 \cs_new_protected:Nn \stex_path_canonicalize:N {
426   \seq_if_empty:NF #1 {
427     \seq_clear:N \l_tmpa_seq
428     \seq_get_left:NN #1 \l_tmpa_tl
429     \str_if_empty:NT \l_tmpa_tl {
430       \seq_put_right:Nn \l_tmpa_seq {}
431     }
432     \seq_map_inline:Nn #1 {
433       \str_set:Nn \l_tmpa_tl { ##1 }
434       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
435         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
436           \seq_if_empty:NNTF \l_tmpa_seq {
437             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
438               \c__stex_path_up_str
439             }
440           }{
441             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
442             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
443               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
444                 \c__stex_path_up_str
445               }
446             }{

```

```

447         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
448     }
449 }
450 }{
451     \str_if_empty:NF \l_tmpa_tl {
452         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
453     }
454 }
455 }
456 }
457 \seq_gset_eq:NN #1 \l_tmpa_seq
458 }
459 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 66.)

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:N \underline{TF}`

```

460 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
461     \seq_if_empty:NTF #1 {
462         \prg_return_false:
463     }{
464         \seq_get_left:NN #1 \l_tmpa_tl
465         \sys_if_platform_windows:TF{
466             \str_if_in:NnTF \l_tmpa_tl {:}{
467                 \prg_return_true:
468             }{
469                 \prg_return_false:
470             }
471         }{
472             \str_if_empty:NTF \l_tmpa_tl {
473                 \prg_return_true:
474             }{
475                 \prg_return_false:
476             }
477         }
478     }
479 }

```

(End definition for `\stex_path_if_absolute:N \underline{TF}` . This function is documented on page 66.)

24.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

480 \str_new:N\l_stex_kpsewhich_return_str
481 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
482     \catcode'\ =12
483     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
484     \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
485     \endgroup
486     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
487     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
488 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 66.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

489 \sys_if_platform_windows:TF{
490   \begingroup\escapechar=-1\catcode'\=12
491   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
492   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
493   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
494   }}{
495   \stex_kpsewhich:n{-var-value~PWD}
496   }
497
498 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
499 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
500 \stex_debug:nn {mathhub} {PWD:~\str_use:N\c_stex_pwd_str}

```

(End definition for `\c_stex_pwd_seq` and `\c_stex_pwd_str`. These variables are documented on page 66.)

24.3 File Hooks and Tracking

501 `<@@=stex_files>`

We introduce hooks for file inputs that keep track of the absolute paths of files used. This will be useful to keep track of modules, their archives, namespaces etc.

Note that the absolute paths are only accurate in `\input`-statements for paths relative to the PWD, so they shouldn't be relied upon in any other setting than for \TeX -purposes.

`\g__stex_files_stack` keeps track of file changes

502 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

503 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
504 \stex_path_from_string:Nn \c_stex_mainfile_seq
505   \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 66.)

`\g_stex_currentfile_seq`

506 `\seq_gclear_new:N\g_stex_currentfile_seq`

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 67.)

`\stex_filestack_push:n`

```

507 \cs_new_protected:Nn \stex_filestack_push:n {
508   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
509   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
510     \stex_path_from_string:Nn\g_stex_currentfile_seq{
511       \c_stex_pwd_str/#1
512     }

```

```

513 }
514 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
515 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
516 }

```

(End definition for `\stex_filestack_push:n`. This function is documented on page 67.)

`\stex_filestack_pop:`

```

517 \cs_new_protected:Nn \stex_filestack_pop: {
518   \seq_if_empty:NF\g__stex_files_stack{
519     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
520   }
521   \seq_if_empty:NTF\g__stex_files_stack{
522     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
523   }{
524     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
525     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
526   }
527 }

```

(End definition for `\stex_filestack_pop:`. This function is documented on page 67.)

Hooks for the current file:

```

528 \AddToHook{file/before}{
529   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
530 }
531 \AddToHook{file/after}{
532   \stex_filestack_pop:
533 }

```

24.4 MathHub Repositories

```

534 <@=stex_mathhub>

```

`\mathhub` The path to the mathhub directory. If the `\mathhub`-macro is not set, we query `\c_stex_mathhub_seq` `kpsewhich` for the MATHHUB system variable.

`\c_stex_mathhub_str`

```

535 \str_if_empty:NTF\mathhub{
536   \sys_if_platform_windows:TF{
537     \begingroup\escapechar=-1\catcode'\=12
538     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
539     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
540     \exp_args:NNx\str_if_eq:ont\l_stex_kpsewhich_return_str{\c_percent_str MATHHUB\c_percent_str}
541     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
542   }{
543     \stex_kpsewhich:n{-var-value-MATHHUB}
544   }
545   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
546 }
547 \str_if_empty:NT \c_stex_mathhub_str {
548   \sys_if_platform_windows:TF{
549     \begingroup\escapechar=-1\catcode'\=12
550     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
551     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
552     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}

```



```

553   }{
554     \stex_kpsewhich:n{-var-value-HOME}
555   }
556   \ior_open:NnT \g_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
557     \begingroup\escapechar=-1\catcode'\=12
558     \ior_str_get:NN \g_tmpa_ior \l_tmpa_str
559     \sys_if_platform_windows:T{
560       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
561     }
562     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
563     \endgroup
564     \ior_close:N \g_tmpa_ior
565   }
566 }
567 \str_if_empty:NTF\c_stex_mathhub_str{
568   \msg_warning:nn{stex}{warning/nomathhub}
569 }{
570   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
571   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
572 }
573 }{
574   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
575   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
576     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
577       \c_stex_pwd_str/\mathhub
578     }
579   }
580   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
581   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
582 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 67.)

`_stex_mathhub_do_manifest:n` Checks whether the manifest for archive #1 already exists, and if not, finds and parses the corresponding manifest file

```

583 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
584   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
585     \str_set:Nx \l_tmpa_str { #1 }
586     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
587     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
588     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
589     \_stex_mathhub_find_manifest:N \l_tmpa_seq
590     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
591       \msg_error:nnxx{stex}{error/norepository}{#1}{
592         \stex_path_to_string:N \c_stex_mathhub_str
593       }
594       \input{Fatal-Error!}
595     } {
596       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
597     }
598   }
599 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

\l_stex_mathhub_manifest_file_seq

```
600 \seq_new:N\l__stex_mathhub_manifest_file_seq
```

(End definition for \l_stex_mathhub_manifest_file_seq.)

__stex_mathhub_find_manifest:N

Attempts to find the MANIFEST.MF in some file path and stores its path in \l__stex_mathhub_manifest_file_seq:

```
601 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
602   \seq_set_eq:NN\l_tmpa_seq #1
603   \bool_set_true:N\l_tmpa_bool
604   \bool_while_do:Nn \l_tmpa_bool {
605     \seq_if_empty:NTF \l_tmpa_seq {
606       \bool_set_false:N\l_tmpa_bool
607     }{
608       \file_if_exist:nTF{
609         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
610       }{
611         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
612         \bool_set_false:N\l_tmpa_bool
613       }{
614         \file_if_exist:nTF{
615           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
616         }{
617           \seq_put_right:Nn\l_tmpa_seq{META-INF}
618           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
619           \bool_set_false:N\l_tmpa_bool
620         }{
621           \file_if_exist:nTF{
622             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
623           }{
624             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
625             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
626             \bool_set_false:N\l_tmpa_bool
627           }{
628             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
629           }
630         }
631       }
632     }
633   }
634   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
635 }
```

(End definition for __stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior

File variable used for MANIFEST-files

```
636 \ior_new:N \c__stex_mathhub_manifest_ior
```

(End definition for \c_stex_mathhub_manifest_ior.)

__stex_mathhub_parse_manifest:n

Stores the entries in manifest file in the corresponding property list:

```
637 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
638   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
639   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
```

```

640 \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
641   \str_set:Nn \l_tmpa_str {##1}
642   \exp_args:NNoo \seq_set_split:Nnn
643     \l_tmpb_seq \c_colon_str \l_tmpa_str
644   \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
645     \exp_args:NNe \str_set:Nn \l_tmpb_tl {
646       \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
647     }
648     \exp_args:No \str_case:nnTF \l_tmpa_tl {
649       {id} {
650         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
651           { id } \l_tmpb_tl
652       }
653       {narration-base} {
654         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
655           { narr } \l_tmpb_tl
656       }
657       {url-base} {
658         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
659           { docurl } \l_tmpb_tl
660       }
661       {source-base} {
662         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
663           { ns } \l_tmpb_tl
664       }
665       {ns} {
666         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
667           { ns } \l_tmpb_tl
668       }
669       {dependencies} {
670         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
671           { deps } \l_tmpb_tl
672       }
673     }{}{}
674   }{}
675 }
676 \ior_close:N \c__stex_mathhub_manifest_ior
677 \stex_persist:x {
678   \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
679     \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
680   }
681 }
682 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

683 \cs_new_protected:Nn \stex_set_current_repository:n {
684   \stex_require_repository:n { #1 }
685   \prop_set_eq:Nc \l_stex_current_repository_prop {
686     c_stex_mathhub_#1_manifest_prop
687   }
688 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 67.)

`\stex_require_repository:n`

```

689 \cs_new_protected:Nn \stex_require_repository:n {
690   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
691     \stex_debug:nn{mathhub}{Opening~archive:~#1}
692     \__stex_mathhub_do_manifest:n { #1 }
693   }
694 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 67.)

`\l_stex_current_repository_prop`

Current MathHub repository

```

695 %\prop_new:N \l_stex_current_repository_prop
696 \bool_if:NF \c_stex_persist_mode_bool {
697   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
698   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
699     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
700   } {
701     \__stex_mathhub_parse_manifest:n { main }
702     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
703     \l_tmpa_str
704     \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
705     \c_stex_mathhub_main_manifest_prop
706     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
707     \stex_debug:nn{mathhub}{Current~repository:~
708     \prop_item:Nn \l_stex_current_repository_prop {id}
709   }
710 }
711 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 67.)

`\stex_in_repository:nn`

Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```

712 \cs_new_protected:Nn \stex_in_repository:nn {
713   \str_set:Nx \l_tmpa_str { #1 }
714   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
715   \str_if_empty:NTF \l_tmpa_str {
716     \prop_if_exist:NTF \l_stex_current_repository_prop {
717       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
718       \exp_args:Ne \l_tmpa_cs{
719         \prop_item:Nn \l_stex_current_repository_prop { id }
720       }
721     }{
722       \l_tmpa_cs{}
723     }
724   }{
725     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
726     \stex_require_repository:n \l_tmpa_str
727     \str_set:Nx \l_tmpa_str { #1 }
728     \exp_args:Nne \use:nn {
729       \stex_set_current_repository:n \l_tmpa_str
730       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
731     }{
732       \stex_debug:nn{mathhub}{switching~back~to:~

```

```

733     \prop_if_exist:NTF \l_stex_current_repository_prop {
734       \prop_item:Nn \l_stex_current_repository_prop { id } :~
735       \meaning\l_stex_current_repository_prop
736     }{
737       no~repository
738     }
739   }
740   \prop_if_exist:NTF \l_stex_current_repository_prop {
741     \stex_set_current_repository:n {
742       \prop_item:Nn \l_stex_current_repository_prop { id }
743     }
744   }{
745     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
746   }
747 }
748 }
749 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 67.)

24.5 Using Content in Archives

`\mhpath`

```

750 \def \mhpath #1 #2 {
751   \exp_args:Ne \tl_if_empty:nTF{#1}{
752     \c_stex_mathhub_str /
753     \prop_item:Nn \l_stex_current_repository_prop { id }
754     / source / #2
755   }{
756     \c_stex_mathhub_str / #1 / source / #2
757   }
758 }

```

(End definition for `\mhpath`. This function is documented on page 68.)

`\inputref`

`\mhinput`

```

759 \newif \ifinputref \inputreffalse
760
761 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
762   \stex_in_repository:nn {#1} {
763     \ifinputref
764       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
765     \else
766       \inputreftrue
767       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
768       \inputreffalse
769     \fi
770   }
771 }
772 \NewDocumentCommand \mhinput { 0{} m }{
773   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
774 }
775

```

```

776 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
777   \stex_in_repository:nn {#1} {
778     \stex_html_backend:TF {
779       \str_clear:N \l_tmpa_str
780       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
781         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
782       }
783
784       \tl_if_empty:nTF{ ##1 }{
785         \IfFileExists{#2}{
786           \stex_annotate_invisible:nnn{inputref}{
787             \l_tmpa_str / #2
788           }{}
789         }{
790           \input{#2}
791         }
792       }{
793         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
794           \stex_annotate_invisible:nnn{inputref}{
795             \l_tmpa_str / #2
796           }{}
797         }{
798           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
799         }
800       }
801
802     }{
803       \begingroup
804       \inputreftrue
805       \tl_if_empty:nTF{ ##1 }{
806         \input{#2}
807       }{
808         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
809       }
810       \endgroup
811     }
812   }
813 }
814 \NewDocumentCommand \inputref { 0{} m}{
815   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
816 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 68.)

`\addmhbibresource`

```

817 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
818   \stex_in_repository:nn {#1} {
819     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
820   }
821 }
822 \newcommand\addmhbibresource[2][]{
823   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
824 }

```

(End definition for `\addmhbibresource`. This function is documented on page 68.)

`\libinput`

```
825 \cs_new_protected:Npn \libinput #1 {
826   \prop_if_exist:NF \l_stex_current_repository_prop {
827     \msg_error:nnn{stex}{error/notinarchive}\libinput
828   }
829   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
830     \msg_error:nnn{stex}{error/notinarchive}\libinput
831   }
832   \seq_clear:N \l__stex_mathhub_libinput_files_seq
833   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
834   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
835
836   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
837     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
838     \IfFileExists{ \l_tmpa_str }{
839       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
840     }{}
841     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
842     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
843   }
844
845   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
846   \IfFileExists{ \l_tmpa_str }{
847     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
848   }{}
849
850   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
851     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
852   }{
853     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
854       \input{ ##1 }
855     }
856   }
857 }
```

(End definition for `\libinput`. This function is documented on page 68.)

`\libusepackage`

```
858 \NewDocumentCommand \libusepackage {0{ } m} {
859   \prop_if_exist:NF \l_stex_current_repository_prop {
860     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
861   }
862   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
863     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
864   }
865   \seq_clear:N \l__stex_mathhub_libinput_files_seq
866   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
867   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
868
869   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
870     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
871     \IfFileExists{ \l_tmpa_str.sty }{
872       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
873     }{}
874   }
```

```

874 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
875 \seq_put_right:No \l_tmpa_seq \l_tmpa_str
876 }
877
878 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
879 \IfFileExists{ \l_tmpa_str.sty }{
880 \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
881 }{}
882
883 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
884 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
885 }{
886 \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
887 \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
888 \usepackage[#1]{ #1 }
889 }
890 }{
891 \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
892 }
893 }
894 }

```

(End definition for `\libusepackage`. This function is documented on page 68.)

`\mhgraphics`
`\cmhgraphics`

```

895
896 \AddToHook{begindocument}{
897 \ltx@ifpackageloaded{graphicx}{
898 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
899 \providecommand\mhgraphics[2] [] {%
900 \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
901 \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
902 \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
903 }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 68.)

`\lstinputmhlisting`
`\clstinputmhlisting`

```

904 \ltx@ifpackageloaded{listings}{
905 \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
906 \newcommand\lstinputmhlisting[2] [] {%
907 \def\lst@mhrepos{}\setkeys{lst}{#1}%
908 \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
909 \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
910 }{}
911 }
912
913 </package>

```

(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 68.)

Chapter 25

STEX -References Implementation

```
914 <*package>
915
916 %%%%%%%%% stex-references.dtx %%%%%%%%%
917
918 <@@=stex_refs>
    Warnings and error messages
919
```

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

```
920 %\iow_new:N \c__stex_refs_refs_iow
921 \AtBeginDocument{
922 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
923 }
924 \AtEndDocument{
925 % \iow_close:N \c__stex_refs_refs_iow
926 }
```

`\STEXreftitle`

```
927 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
928
929 \NewDocumentCommand \STEXreftitle { m } {
930   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
931 }
```

(End definition for `\STEXreftitle`. This function is documented on page 69.)

25.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
932 \str_new:N \l_stex_current_docns_str
```

(End definition for `\l_stex_current_docns_str`. This variable is documented on page 69.)

`\stex_get_document_uri:`

```
933 \cs_new_protected:Nn \stex_get_document_uri: {
934   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
935   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
936   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
937   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
938   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
939
940   \str_clear:N \l_tmpa_str
941   \prop_if_exist:NT \l_stex_current_repository_prop {
942     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
943       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
944     }
945   }
946
947   \str_if_empty:NTF \l_tmpa_str {
948     \str_set:Nx \l_stex_current_docns_str {
949       file:/\stex_path_to_string:N \l_tmpa_seq
950     }
951   }{
952     \bool_set_true:N \l_tmpa_bool
953     \bool_while_do:Nn \l_tmpa_bool {
954       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
955       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
956         {source} { \bool_set_false:N \l_tmpa_bool }
957       }{}{
958         \seq_if_empty:NT \l_tmpa_seq {
959           \bool_set_false:N \l_tmpa_bool
960         }
961       }
962     }
963
964     \seq_if_empty:NTF \l_tmpa_seq {
965       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
966     }{
967       \str_set:Nx \l_stex_current_docns_str {
968         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
969       }
970     }
971   }
972 }
```

(End definition for `\stex_get_document_uri:`. This function is documented on page 69.)

`\l_stex_current_docurl_str`

```
973 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 69.)

`\stex_get_document_url:`

```
974 \cs_new_protected:Nn \stex_get_document_url: {
975   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
976   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
977   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

978 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
979 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
980
981 \str_clear:N \l_tmpa_str
982 \prop_if_exist:NT \l_stex_current_repository_prop {
983   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
984     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
985       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
986     }
987   }
988 }
989
990 \str_if_empty:NTF \l_tmpa_str {
991   \str_set:Nx \l_stex_current_docurl_str {
992     file:/\stex_path_to_string:N \l_tmpa_seq
993   }
994 }{
995   \bool_set_true:N \l_tmpa_bool
996   \bool_while_do:Nn \l_tmpa_bool {
997     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
998     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
999       {source} { \bool_set_false:N \l_tmpa_bool }
1000     }{}{
1001       \seq_if_empty:NT \l_tmpa_seq {
1002         \bool_set_false:N \l_tmpa_bool
1003       }
1004     }
1005   }
1006 }
1007 \seq_if_empty:NTF \l_tmpa_seq {
1008   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1009 }{
1010   \str_set:Nx \l_stex_current_docurl_str {
1011     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1012   }
1013 }
1014 }
1015 }

```

(End definition for `\stex_get_document_url`:. This function is documented on page 69.)

25.2 Setting Reference Targets

```

1016 \str_const:Nn \c__stex_refs_url_str{URL}
1017 \str_const:Nn \c__stex_refs_ref_str{REF}
1018 \str_new:N \l__stex_refs_curr_label_str
1019 % @currentlabel -> number
1020 % @currentlabelname -> title
1021 % @currentHref -> name.number <- id of some kind
1022 % \theH# -> \arabic{section}
1023 % \the# -> number
1024 % \hyper@makecurrent{#}
1025 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

1026 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1027   \stex_get_document_uri:
1028   \str_clear:N \l__stex_refs_curr_label_str
1029   \str_set:Nx \l_tmpa_str { #1 }
1030   \str_if_empty:NT \l_tmpa_str {
1031     \int_incr:N \l__stex_refs_unnamed_counter_int
1032     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1033   }
1034   \str_set:Nx \l__stex_refs_curr_label_str {
1035     \l_stex_current_docns_str?\l_tmpa_str
1036   }
1037   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1038     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1039   }
1040   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1041     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1042   }
1043   \stex_if_smsmode:TF {
1044     \stex_get_document_url:
1045     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1046     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1047   }{
1048     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1049     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1050     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1051     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1052   }
1053 }

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 69.)

The following is used to set the necessary macros in the .aux-file.

```

1054 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1055   \str_set:Nn \l_tmpa_str {#1?#2}
1056   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1057   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1058     \seq_new:c {g__stex_refs_labels_#2_seq}
1059   }
1060   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1061     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1062   }
1063 }

```

To avoid resetting the same macros when the .aux-file is read at the end of the document:

```

1064 \AtEndDocument{
1065   \def\stexauxadddocref#1 #2 {}{}
1066 }

```

`\stex_ref_new_sym_target:n`

```

1067 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1068   \stex_if_smsmode:TF {
1069     \str_if_exist:cF{sref_sym_#1_type}{
1070       \stex_get_document_url:
1071       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

1072     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1073   }
1074   ){
1075     \str_if_empty:NF \l__stex_refs_curr_label_str {
1076       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1077       \immediate\write\@auxout{
1078         \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1079           \l__stex_refs_curr_label_str
1080         }
1081       }
1082     }
1083   }
1084 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 69.)

25.3 Using References

```

1085 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

1086
1087 \keys_define:nn { stex / sref } {
1088   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1089   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1090   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1091   post          .tl_set:N = \l__stex_refs_post_tl ,
1092 }
1093 \cs_new_protected:Nn \__stex_refs_args:n {
1094   \tl_clear:N \l__stex_refs_linktext_tl
1095   \tl_clear:N \l__stex_refs_fallback_tl
1096   \tl_clear:N \l__stex_refs_pre_tl
1097   \tl_clear:N \l__stex_refs_post_tl
1098   \str_clear:N \l__stex_refs_repo_str
1099   \keys_set:nn { stex / sref } { #1 }
1100 }

```

The actual macro:

```

1101 \NewDocumentCommand \sref { 0{} m}{
1102   \__stex_refs_args:n { #1 }
1103   \str_if_empty:NTF \l__stex_refs_indocument_str {
1104     \str_set:Nx \l_tmpa_str { #2 }
1105     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1106     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1107       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1108         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1109           \str_clear:N \l_tmpa_str
1110         }
1111       }{
1112         \str_clear:N \l_tmpa_str
1113       }
1114     }{
1115       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1116       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1117 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1118 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1119   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1120   \str_clear:N \l_tmpa_str
1121   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1122     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1123       \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1124     }{
1125       \seq_map_break:n {
1126         \str_set:Nn \l_tmpa_str { ##1 }
1127       }
1128     }
1129   }
1130 }{
1131   \str_clear:N \l_tmpa_str
1132 }
1133 }
1134 \str_if_empty:NTF \l_tmpa_str {
1135   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1136 }{
1137   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1138     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1139       \cs_if_exist:cTF{autoref}{
1140         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1141       }{
1142         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1143       }
1144     }{
1145       \ltx@ifpackageloaded{hyperref}{
1146         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1147       }{
1148         \l__stex_refs_linktext_tl
1149       }
1150     }
1151   }{
1152     \ltx@ifpackageloaded{hyperref}{
1153       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1154     }{
1155       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1156     }
1157   }
1158 }
1159 }{
1160   % TODO
1161 }
1162 }

```

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

`\srefsym`

```

1163 \NewDocumentCommand \srefsym { 0{} m }{
1164   \stex_get_symbol:n { #2 }
1165   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1166 }

```

```

1167
1168 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1169   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1170     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1171   }{
1172     \__stex_refs_args:n { #1 }
1173     \str_if_empty:NTF \l__stex_refs_indocument_str {
1174       \tl_if_exist:cTF{sref_sym_#2 _type}{
1175         % doc uri in \l_tmpb_str
1176         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1177         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1178           % reference
1179           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1180             \cs_if_exist:cTF{autoref}{
1181               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1182             }{
1183               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1184             }
1185           }{
1186             \ltx@ifpackageloaded{hyperref}{
1187               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1188             }{
1189               \l__stex_refs_linktext_tl
1190             }
1191           }
1192         }{
1193           % URL
1194           \ltx@ifpackageloaded{hyperref}{
1195             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1196           }{
1197             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1198           }
1199         }
1200       }{
1201         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1202       }
1203     }{
1204       % TODO
1205     }
1206   }
1207 }

```

(End definition for \srefsym. This function is documented on page 70.)

\srefsymuri

```

1208 \cs_new_protected:Npn \srefsymuri #1 #2 {
1209   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1210 }

```

(End definition for \srefsymuri. This function is documented on page 70.)

```

1211 </package>

```

Chapter 26

STEX -Modules Implementation

```
1212 <*package>
1213
1214 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1215
1216 <@@=stex_modules>
1217
1218 Warnings and error messages
1217 \msg_new:nnn{stex}{error/unknownmodule}{
1218   No~module~#1~found
1219 }
1220 \msg_new:nnn{stex}{error/syntax}{
1221   Syntax~error:~#1
1222 }
1223 \msg_new:nnn{stex}{error/siglanguage}{
1224   Module~#1~declares~signature~#2,~but~does~not~
1225   declare~its~language
1226 }
1227 \msg_new:nnn{stex}{warning/deprecated}{
1228   #1~is~deprecated;~please~use~#2~instead!
1229 }
1230
1231 \msg_new:nnn{stex}{error/conflictingmodules}{
1232   Conflicting~imports~for~module~#1
1233 }
```

`\l_stex_current_module_str` The current module:

```
1234 \str_new:N \l_stex_current_module_str
```

(End definition for `\l_stex_current_module_str`. This variable is documented on page 72.)

`\l_stex_all_modules_seq` Stores all available modules

```
1235 \seq_new:N \l_stex_all_modules_seq
```

(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 72.)


```

\stex_if_in_module_p:
\stex_if_in_module:TF
1236 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1237   \str_if_empty:NTF \l_stex_current_module_str
1238   \prg_return_false: \prg_return_true:
1239 }

```

(End definition for `\stex_if_in_module:TF`. This function is documented on page 72.)

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1240 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1241   \prop_if_exist:cTF { c_stex_module_#1_prop }
1242   \prg_return_true: \prg_return_false:
1243 }

```

(End definition for `\stex_if_module_exists:nTF`. This function is documented on page 72.)

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1244 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1245   \stex_add_to_current_module:n { #1 }
1246   \stex_do_up_to_module:n { #1 }
1247 }}
1248 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1249
1250 \cs_new_protected:Nn \stex_add_to_current_module:n {
1251   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1252 }
1253 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1254 \cs_new_protected:Npn \STEXexport {
1255   \ExplSyntaxOn
1256   \__stex_modules_export:n
1257 }
1258 \cs_new_protected:Nn \__stex_modules_export:n {
1259   \ignorespacesandpars#1\ExplSyntaxOff
1260   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1261   \stex_smsmode_do:
1262 }
1263 \let \stex_module_export_helper:n \use:n
1264 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 72.)

```

\stex_add_constant_to_current_module:n
1265 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1266   \str_set:Nx \l_tmpa_str { #1 }
1267   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1268 }

```

(End definition for `\stex_add_constant_to_current_module:n`. This function is documented on page 72.)

```

\stex_add_import_to_current_module:n
1269 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1270   \str_set:Nx \l_tmpa_str { #1 }
1271   \exp_args:Nno

```

```

1272 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1273 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1274 }
1275 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 72.)

`\stex_collect_imports:n`

```

1276 \cs_new_protected:Nn \stex_collect_imports:n {
1277 \seq_clear:N \l_stex_collect_imports_seq
1278 \__stex_modules_collect_imports:n {#1}
1279 }
1280 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1281 \seq_map_inline:cn {c_stex_module_#1_imports} {
1282 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1283 \__stex_modules_collect_imports:n { ##1 }
1284 }
1285 }
1286 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1287 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1288 }
1289 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 72.)

`\stex_do_up_to_module:n`

```

1290 \int_new:N \l__stex_modules_group_depth_int
1291 \cs_new_protected:Nn \stex_do_up_to_module:n {
1292 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1293 #1
1294 }{
1295 #1
1296 \expandafter \tl_gset:Nn
1297 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1298 \expandafter\expandafter\expandafter\endcsname
1299 \expandafter\expandafter\expandafter { \csname
1300 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1301 \aftergroup\__stex_modules_aftergroup_do:
1302 }
1303 }
1304 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1305 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1306 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1307 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1308 }}}
1309 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1310 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1311 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1312 }{
1313 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1314 \aftergroup\__stex_modules_aftergroup_do:
1315 }
1316 }
1317 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1318 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1319 }

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 72.)

`\stex_modules_compute_namespace:nN` Computes the appropriate namespace from the top-level namespace of a repository (#1) and a file path (#2).

1320

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

`\stex_modules_current_namespace:` Computes the current namespace based on the current MathHub repository (if existent) and the current file.

```
1321 \str_new:N \l_stex_module_ns_str
1322 \str_new:N \l_stex_module_subpath_str
1323 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1324   \seq_set_eq:NN \l_tmpa_seq #2
1325   % split off file extension
1326   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1327   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1328   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1329   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1330
1331   \bool_set_true:N \l_tmpa_bool
1332   \bool_while_do:Nn \l_tmpa_bool {
1333     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1334     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1335       {source} { \bool_set_false:N \l_tmpa_bool }
1336     }{}{
1337       \seq_if_empty:NT \l_tmpa_seq {
1338         \bool_set_false:N \l_tmpa_bool
1339       }
1340     }
1341   }
1342
1343   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1344   % \l_tmpa_seq <- sub-path relative to archive
1345   \str_if_empty:NTF \l_stex_module_subpath_str {
1346     \str_set:Nx \l_stex_module_ns_str {#1}
1347   }{
1348     \str_set:Nx \l_stex_module_ns_str {
1349       #1/\l_stex_module_subpath_str
1350     }
1351   }
1352 }
1353
1354 \cs_new_protected:Nn \stex_modules_current_namespace: {
1355   \str_clear:N \l_stex_module_subpath_str
1356   \prop_if_exist:NTF \l_stex_current_repository_prop {
1357     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1358     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1359   }{
1360     % split off file extension
1361     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1362     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

1363 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1364 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1365 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1366 \str_set:Nx \l_stex_module_ns_str {
1367   file:/\stex_path_to_string:N \l_tmpa_seq
1368 }
1369 }
1370 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 73.)

26.1 The smodule environment

smodule arguments:

```

1371 \keys_define:nn { stex / module } {
1372   title      .tl_set:N      = \smodulename ,
1373   type       .str_set_x:N   = \smodulename ,
1374   id         .str_set_x:N   = \smoduleid ,
1375   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1376   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1377   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1378   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1379   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1380   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1381   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1382   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1383 }
1384
1385 \cs_new_protected:Nn \__stex_modules_args:n {
1386   \str_clear:N \smodulename
1387   \str_clear:N \smodulename
1388   \str_clear:N \smoduleid
1389   \str_clear:N \l_stex_module_ns_str
1390   \str_clear:N \l_stex_module_deprecate_str
1391   \str_clear:N \l_stex_module_lang_str
1392   \str_clear:N \l_stex_module_sig_str
1393   \str_clear:N \l_stex_module_creators_str
1394   \str_clear:N \l_stex_module_contributors_str
1395   \str_clear:N \l_stex_module_meta_str
1396   \str_clear:N \l_stex_module_srccite_str
1397   \keys_set:nn { stex / module } { #1 }
1398 }
1399
1400 % module parameters here? In the body?
1401

```

`\stex_module_setup:nn` Sets up a new module property list:

```

1402 \cs_new_protected:Nn \stex_module_setup:nn {
1403   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1404   \str_set:Nx \l_stex_module_name_str { #2 }
1405   \__stex_modules_args:n { #1 }

```

First, we set up the name and namespace of the module.

Are we in a nested module?

```

1406 \stex_if_in_module:TF {
1407   % Nested module
1408   \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1409   { ns } \l_stex_module_ns_str
1410   \str_set:Nx \l_stex_module_name_str {
1411     \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1412     { name } / \l_stex_module_name_str
1413   }
1414   \str_if_empty:NT \l_stex_module_lang_str {
1415     \str_set:Nx \l_stex_module_lang_str {
1416       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1417       { lang }
1418     }
1419   }
1420 }{
1421   % not nested:
1422   \str_if_empty:NT \l_stex_module_ns_str {
1423     \stex_modules_current_namespace:
1424     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1425       / {\l_stex_module_ns_str}
1426     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1427     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1428       \str_set:Nx \l_stex_module_ns_str {
1429         \stex_path_to_string:N \l_tmpa_seq
1430       }
1431     }
1432   }
1433 }

```

Next, we determine the language of the module:

```

1434 \str_if_empty:NT \l_stex_module_lang_str {
1435   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1436   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1437   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1438   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1439     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1440       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1441     }
1442   }
1443   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1444   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1445     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1446     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1447       inferred~from~file~name}
1448   }
1449 }
1450
1451 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1452   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1453 }}

```

We check if we need to extend a signature module, and set `\l_stex_current_module_prop` accordingly:

```

1454 \str_if_empty:NTF \l_stex_module_sig_str {
1455   \exp_args:Nnx \prop_gset_from_keyval:cn {
1456     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1457   } {
1458     name      = \l_stex_module_name_str ,
1459     ns        = \l_stex_module_ns_str ,
1460     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1461     lang      = \l_stex_module_lang_str ,
1462     sig       = \l_stex_module_sig_str ,
1463     deprecate = \l_stex_module_deprecate_str ,
1464     meta      = \l_stex_module_meta_str
1465   }
1466   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1467   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1468   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1469   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1470   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1471 \str_if_empty:NT \l_stex_module_meta_str {
1472   \str_set:Nx \l_stex_module_meta_str {
1473     \c_stex_metatheory_ns_str ? Metatheory
1474   }
1475 }
1476 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1477   \bool_set_true:N \l_stex_in_meta_bool
1478   \exp_args:Nx \stex_add_to_current_module:n {
1479     \bool_set_true:N \l_stex_in_meta_bool
1480     \stex_activate_module:n {\l_stex_module_meta_str}
1481     \bool_set_false:N \l_stex_in_meta_bool
1482   }
1483   \stex_activate_module:n {\l_stex_module_meta_str}
1484   \bool_set_false:N \l_stex_in_meta_bool
1485 }
1486 }{
1487   \str_if_empty:NT \l_stex_module_lang_str {
1488     \msg_error:nxxx{stex}{error/siglanguage}{
1489       \l_stex_module_ns_str?\l_stex_module_name_str
1490     }{\l_stex_module_sig_str}
1491   }
1492   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1493   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1494     \stex_debug:nn{modules}{(already exists)}
1495   }{
1496     \stex_debug:nn{modules}{(needs loading)}
1497     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1498     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1499     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1500     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1501     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1502     \str_set:Nx \l_tmpa_str {
1503       \stex_path_to_string:N \l_tmpa_seq /

```

```

1504     \l_tmpa_str . \l_stex_module_sig_str .tex
1505   }
1506   \IfFileExists \l_tmpa_str {
1507     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1508       \str_clear:N \l_stex_current_module_str
1509       \seq_clear:N \l_stex_all_modules_seq
1510       \stex_debug:nn{modules}{Loading~signature}
1511     }
1512   }{
1513     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1514   }
1515 }
1516 \stex_if_smsmode:F {
1517   \stex_activate_module:n {
1518     \l_stex_module_ns_str ? \l_stex_module_name_str
1519   }
1520 }
1521 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1522 }
1523 \str_if_empty:NF \l_stex_module_deprecate_str {
1524   \msg_warning:nnxx{stex}{warning/deprecated}{
1525     Module~\l_stex_current_module_str
1526   }{
1527     \l_stex_module_deprecate_str
1528   }
1529 }
1530 \seq_put_right:Nx \l_stex_all_modules_seq {
1531   \l_stex_module_ns_str ? \l_stex_module_name_str
1532 }
1533 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1534 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page [73](#).)

smodule (*env.*) The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}

1535 \cs_new_protected:Nn \__stex_modules_begin_module: {
1536   \stex_reactivate_macro:N \STEXexport
1537   \stex_reactivate_macro:N \importmodule
1538   \stex_reactivate_macro:N \symdecl
1539   \stex_reactivate_macro:N \notation
1540   \stex_reactivate_macro:N \symdef
1541
1542   \stex_debug:nn{modules}{
1543     New~module:\\
1544     Namespace:~\l_stex_module_ns_str\\
1545     Name:~\l_stex_module_name_str\\
1546     Language:~\l_stex_module_lang_str\\
1547     Signature:~\l_stex_module_sig_str\\
1548     Metatheory:~\l_stex_module_meta_str\\
1549     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1550   }
1551

```

```

1552 \stex_if_do_html:T{
1553   \begin{stex_annotate_env} {theory} {
1554     \l_stex_module_ns_str ? \l_stex_module_name_str
1555   }
1556
1557   \stex_annotate_invisible:nnn{header}{} {
1558     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1559     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1560     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1561       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1562     }
1563     \str_if_empty:NF \smoduletype {
1564       \stex_annotate:nnn{type}{\smoduletype}{}
1565     }
1566   }
1567 }
1568 % TODO: Inherit metatheory for nested modules?
1569 }
1570 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `_stex_modules_begin_module:.`)

`_stex_modules_end_module:` implements `\end{module}`

```

1571 \cs_new_protected:Nn \_stex_modules_end_module: {
1572   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module}
1573   \stex_reset_up_to_module:n \l_stex_current_module_str
1574   \stex_if_smsmode:T {
1575     \stex_persist:x {
1576       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1577         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1578       }
1579       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1580         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1581       }
1582       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1583         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1584       }
1585       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1586     }
1587     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1588     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1589   }
1590 }

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1591 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1592 \NewDocumentEnvironment { smodule } { 0 } { m } {
1593   \stex_module_setup:nn{#1}{#2}
1594   %\par
1595   \stex_if_smsmode:F{
1596     \tl_if_empty:NF \smoduletitle {
1597       \exp_args:No \stex_document_title:n \smoduletitle
1598     }

```



```

1599 \tl_clear:N \l_tmpa_tl
1600 \clist_map_inline:Nn \smodulotype {
1601   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1602     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1603   }
1604 }
1605 \tl_if_empty:NTF \l_tmpa_tl {
1606   \__stex_modules_smodule_start:
1607 }{
1608   \l_tmpa_tl
1609 }
1610 }
1611 \__stex_modules_begin_module:
1612 \str_if_empty:NF \smoduleid {
1613   \stex_ref_new_doc_target:n \smoduleid
1614 }
1615 \stex_smsmode_do:
1616 } {
1617   \__stex_modules_end_module:
1618   \stex_if_smsmode:F {
1619     \end{stex_annotate_env}
1620     \clist_set:Nn \l_tmpa_clist \smodulotype
1621     \tl_clear:N \l_tmpa_tl
1622     \clist_map_inline:Nn \l_tmpa_clist {
1623       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1624         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1625       }
1626     }
1627     \tl_if_empty:NTF \l_tmpa_tl {
1628       \__stex_modules_smodule_end:
1629     }{
1630       \l_tmpa_tl
1631     }
1632   }
1633 }

```

\stexpatchmodule

```

1634 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1635 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1636
1637 \newcommand\stexpatchmodule[3] [] {
1638   \str_set:Nx \l_tmpa_str{ #1 }
1639   \str_if_empty:NTF \l_tmpa_str {
1640     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1641     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1642   }{
1643     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1644     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1645   }
1646 }

```

(End definition for \stexpatchmodule. This function is documented on page 73.)

26.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1647 \NewDocumentCommand \STEXModule { m } {
1648   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1649   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1650   \tl_set:Nn \l_tmpa_tl {
1651     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1652   }
1653   \seq_map_inline:Nn \l_stex_all_modules_seq {
1654     \str_set:Nn \l_tmpb_str { ##1 }
1655     \str_if_eq:eeT { \l_tmpa_str } {
1656       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1657     } {
1658       \seq_map_break:n {
1659         \tl_set:Nn \l_tmpa_tl {
1660           \stex_invoke_module:n { ##1 }
1661         }
1662       }
1663     }
1664   }
1665   \l_tmpa_tl
1666 }
1667
1668 \cs_new_protected:Nn \stex_invoke_module:n {
1669   \stex_debug:nn{modules}{Invoking~module~#1}
1670   \peek_charcode_remove:NTF ! {
1671     \__stex_modules_invoke_uri:nN { #1 }
1672   } {
1673     \peek_charcode_remove:NTF ? {
1674       \__stex_modules_invoke_symbol:nn { #1 }
1675     } {
1676       \msg_error:nnx{stex}{error/syntax}{
1677         ?~or~!~expected~after~
1678         \c_backslash_str STEXModule{#1}
1679       }
1680     }
1681   }
1682 }
1683
1684 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1685   \str_set:Nn #2 { #1 }
1686 }
1687
1688 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1689   \stex_invoke_symbol:n{#1?#2}
1690 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 73.)

```

\stex_activate_module:n
1691 \bool_new:N \l_stex_in_meta_bool
1692 \bool_set_false:N \l_stex_in_meta_bool

```

```

1693 \cs_new_protected:Nn \stex_activate_module:n {
1694   \stex_debug:nn{modules}{Activating~module~#1}
1695   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1696     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1697     \use:c{ c_stex_module_#1_code }
1698   }
1699 }

```

(End definition for \stex_activate_module:n. This function is documented on page 74.)

```

1700 \endpackage

```

Chapter 27

STEX -Module Inheritance Implementation

```
1701 <*package>
1702
1703 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1704
```

27.1 SMS Mode

```
1705 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1706 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1707 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1708 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1709
1710 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1711   \makeatletter
1712   \makeatother
1713   \ExplSyntaxOn
1714   \ExplSyntaxOff
1715   \rustexBREAK
1716 }
1717
1718 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1719   \symdef
1720   \importmodule
1721   \notation
1722   \symdecl
1723   \STEXexport
1724   \inlineass
1725   \inlinedef
1726   \inlineex
1727   \endinput
1728   \setnotation
```

```

1729 \copynotation
1730 \assign
1731 \renamedekl
1732 \donotcopy
1733 \instantiate
1734 \textsymdecl
1735 }
1736
1737 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1738   \tl_to_str:n {
1739     smodule,
1740     copymodule,
1741     interpretmodule,
1742     realization,
1743     sdefinition,
1744     sexample,
1745     sassertion,
1746     sparagraph,
1747     mathstructure
1748   }
1749 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 75.)

`\stex_if_smsmode_p:`

```

\stex_if_smsmode:TF
1750 \bool_new:N \g__stex_smsmode_bool
1751 \bool_set_false:N \g__stex_smsmode_bool
1752 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1753   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1754 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 75.)

`_stex_smsmode_in_smsmode:nn`

```

1755 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1756   \vbox_set:Nn \l_tmpa_box {
1757     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1758     \bool_gset_true:N \g__stex_smsmode_bool
1759     #2
1760     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1761   }
1762   \box_clear:N \l_tmpa_box
1763 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1764 \quark_new:N \q__stex_smsmode_break
1765
1766 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1767   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1768   \stex_smsmode_do:
1769 }
1770

```

```

1771 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1772   \__stex_modules_args:n{#1}
1773   \stex_if_in_module:F {
1774     \str_if_empty:NF \l_stex_module_sig_str {
1775       \stex_modules_current_namespace:
1776       \str_set:Nx \l_stex_module_name_str { #2 }
1777       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1778         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1779         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1780         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1781         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1782         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1783         \str_set:Nx \l_tmpa_str {
1784           \stex_path_to_string:N \l_tmpa_seq /
1785           \l_tmpa_str . \l_stex_module_sig_str .tex
1786         }
1787         \IfFileExists \l_tmpa_str {
1788           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1789         }{
1790           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1791         }
1792       }
1793     }
1794   }
1795 }
1796
1797 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
1798   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
1799   \tl_if_empty:nTF{#1}{
1800     \prop_if_exist:NTF \l_stex_current_repository_prop
1801     {
1802       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
1803       \prg_return_true:
1804     } {
1805       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
1806       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
1807       \tl_if_empty:NT \l_tmpa_tl {
1808         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
1809       }
1810       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
1811       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
1812       \prg_return_true: \prg_return_false:
1813     }
1814   }\prg_return_true:
1815 }
1816
1817 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1818   \stex_filestack_push:n{#1}
1819   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1820   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1821   % ----- new -----
1822   \__stex_smsmode_in_smsmode:nn{#1}{
1823     \let\importmodule\__stex_smsmode_importmodule:
1824     \let\stex_module_setup:nn\__stex_smsmode_module:nn

```

```

1825 \let\__stex_modules_begin_module:\relax
1826 \let\__stex_modules_end_module:\relax
1827 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1828 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1829 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1830 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1831 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1832 \everyeof{\q__stex_smsmode_break\noexpand}
1833 \expandafter\expandafter\expandafter
1834 \stex_smsmode_do:
1835 \csname @ @ input\endcsname "#1"\relax
1836
1837 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1838   \stex_filestack_push:n{##1}
1839   \expandafter\expandafter\expandafter
1840   \stex_smsmode_do:
1841   \csname @ @ input\endcsname "##1"\relax
1842   \stex_filestack_pop:
1843 }
1844 }
1845 % ----- new -----
1846 \__stex_smsmode_in_smsmode:nn{#1} {
1847   #2
1848   % ----- new -----
1849   \begingroup
1850   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1851   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1852     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
1853       \stex_import_module_uri:nn ##1
1854       \stex_import_require_module:nnnn
1855       \l_stex_import_ns_str
1856       \l_stex_import_archive_str
1857       \l_stex_import_path_str
1858       \l_stex_import_name_str \endgroup
1859     }
1860   }
1861   \endgroup
1862   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1863   % ----- new -----
1864   \everyeof{\q__stex_smsmode_break\noexpand}
1865   \expandafter\expandafter\expandafter
1866   \stex_smsmode_do:
1867   \csname @ @ input\endcsname "#1"\relax
1868 }
1869 \stex_filestack_pop:
1870 }

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 76.)

`\stex_smsmode_do:` is executed on encountering `\` in smsmode. It checks whether the corresponding command is allowed and executes or ignores it accordingly:

```

1871 \cs_new_protected:Npn \stex_smsmode_do: {
1872   \stex_if_smsmode:T {
1873     \__stex_smsmode_do:w

```

```

1874 }
1875 }
1876 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1877   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1878     \expandafter\if\expandafter\relax\noexpand#1
1879     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1880     \else\expandafter\__stex_smsmode_do:w\fi
1881   }{
1882     \__stex_smsmode_do:w % #1
1883   }
1884 }
1885 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1886   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1887     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1888       #1\__stex_smsmode_do:w
1889     }{
1890       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1891         #1
1892       }{
1893         \cs_if_eq:NNTF \begin #1 {
1894           \__stex_smsmode_check_begin:n
1895         }{
1896           \cs_if_eq:NNTF \end #1 {
1897             \__stex_smsmode_check_end:n
1898           }{
1899             \__stex_smsmode_do:w
1900           }
1901         }
1902       }
1903     }
1904   }
1905 }
1906
1907 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1908   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1909     \begin{#1}
1910   }{
1911     \__stex_smsmode_do:w
1912   }
1913 }
1914 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1915   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1916     \end{#1}\__stex_smsmode_do:w
1917   }{
1918     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1919   }
1920 }

```

(End definition for `\stex_smsmode_do:.` This function is documented on page [76](#).)

27.2 Inheritance

```

1921 <@@=stex_importmodule>

```


`\stex_import_module_uri:nn`

```
1922 \cs_new_protected:Nn \stex_import_module_uri:nn {
1923   \str_set:Nx \l_stex_import_archive_str { #1 }
1924   \str_set:Nn \l_stex_import_path_str { #2 }
1925
1926   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1927   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1928   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1929
1930   \stex_modules_current_namespace:
1931   \bool_lazy_all:nTF {
1932     {\str_if_empty_p:N \l_stex_import_archive_str}
1933     {\str_if_empty_p:N \l_stex_import_path_str}
1934     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1935   }{
1936     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1937     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1938   }{
1939     \str_if_empty:NT \l_stex_import_archive_str {
1940       \prop_if_exist:NT \l_stex_current_repository_prop {
1941         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1942       }
1943     }
1944     \str_if_empty:NTF \l_stex_import_archive_str {
1945       \str_if_empty:NF \l_stex_import_path_str {
1946         \stex_path_from_string:Nn \l_tmpb_seq {
1947           \l_stex_module_ns_str / .. / \l_stex_import_path_str
1948         }
1949         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
1950         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file:///}
1951       }
1952     }{
1953       \stex_require_repository:n \l_stex_import_archive_str
1954       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1955       \l_stex_import_ns_str
1956       \str_if_empty:NF \l_stex_import_path_str {
1957         \str_set:Nx \l_stex_import_ns_str {
1958           \l_stex_import_ns_str / \l_stex_import_path_str
1959         }
1960       }
1961     }
1962   }
1963 }
```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 77.)

`\l_stex_import_name_str`
`\l_stex_import_archive_str`
`\l_stex_import_path_str`
`\l_stex_import_ns_str`

Store the return values of `\stex_import_module_uri:nn`.

```
1964 \str_new:N \l_stex_import_name_str
1965 \str_new:N \l_stex_import_archive_str
1966 \str_new:N \l_stex_import_path_str
1967 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 77.)

```

\stex_import_require_module:nnnnn {\langle ns \rangle} {\langle archive-ID \rangle} {\langle path \rangle} {\langle name \rangle}
1968 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1969 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1970
1971 \stex_debug:nn{requiremodule}{Here:\~1:\~2:\~3:\~4}
1972
1973 \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1974 \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1975
1976 %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1977
1978 % archive
1979 \str_set:Nx \l_tmpa_str { #2 }
1980 \str_if_empty:NTF \l_tmpa_str {
1981 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1982 \seq_put_right:Nn \l_tmpa_seq {...}
1983 } {
1984 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1985 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1986 \seq_put_right:Nn \l_tmpa_seq { source }
1987 }
1988
1989 % path
1990 \str_set:Nx \l_tmpb_str { #3 }
1991 \str_if_empty:NTF \l_tmpb_str {
1992 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1993
1994 \ltx@ifpackageloaded{babel} {
1995 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1996 { \language } \l_tmpb_str {
1997 \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1998 }
1999 } {
2000 \str_clear:N \l_tmpb_str
2001 }
2002
2003 \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2004 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2005 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2006 }{
2007 \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2008 \IfFileExists{ \l_tmpa_str.tex }{
2009 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2010 }{
2011 % try english as default
2012 \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2013 \IfFileExists{ \l_tmpa_str.en.tex }{
2014 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2015 }{
2016 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2017 }
2018 }
2019 }
2020

```

```

2021 } {
2022   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2023   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2024
2025   \ltx@ifpackageloaded{babel} {
2026     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2027       { \language } \l_tmpb_str {
2028       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2029     }
2030   } {
2031     \str_clear:N \l_tmpb_str
2032   }
2033
2034   \stex_path_canonicalize:N \l_tmpb_seq
2035   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2036
2037   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2038   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2039     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2040   }{
2041     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2042     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2043       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2044     }{
2045       % try english as default
2046       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2047       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2048         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2049       }{
2050         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2051         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2052           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2053         }{
2054           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2055           \IfFileExists{ \l_tmpa_str.tex }{
2056             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2057           }{
2058             % try english as default
2059             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2060             \IfFileExists{ \l_tmpa_str.en.tex }{
2061               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2062             }{
2063               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2064             }
2065           }
2066         }
2067       }
2068     }
2069   }
2070 }
2071
2072 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2073   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2074     \seq_clear:N \l_stex_all_modules_seq

```

```

2075     \str_clear:N \l_stex_current_module_str
2076     \str_set:Nx \l_tmpb_str { #2 }
2077     \str_if_empty:NF \l_tmpb_str {
2078       \stex_set_current_repository:n { #2 }
2079     }
2080     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2081   }
2082
2083   \stex_if_module_exists:nF { #1 ? #4 } {
2084     \msg_error:nnx{stex}{error/unknownmodule}{
2085       #1?#4~(in~file~\g__stex_importmodule_file_str)
2086     }
2087   }
2088 }
2089
2090 }
2091 \stex_activate_module:n { #1 ? #4 }
2092 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 77.)

`\importmodule`

```

2093 \NewDocumentCommand \importmodule { 0{} m } {
2094   \stex_import_module_uri:nn { #1 } { #2 }
2095   \stex_debug:nn{modules}{Importing~module:~
2096     \l_stex_import_ns_str ? \l_stex_import_name_str
2097   }
2098   \stex_import_require_module:nnnn
2099   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2100   { \l_stex_import_path_str } { \l_stex_import_name_str }
2101   \stex_if_smsmode:F {
2102     \stex_annotate_invisible:nnn
2103     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2104   }
2105   \exp_args:Nx \stex_add_to_current_module:n {
2106     \stex_import_require_module:nnnn
2107     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2108     { \l_stex_import_path_str } { \l_stex_import_name_str }
2109   }
2110   \exp_args:Nx \stex_add_import_to_current_module:n {
2111     \l_stex_import_ns_str ? \l_stex_import_name_str
2112   }
2113   \stex_smsmode_do:
2114   \ignorespacesandpars
2115 }
2116 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

2117 \NewDocumentCommand \usemodule { 0{} m } {
2118   \stex_if_smsmode:F {
2119     \stex_import_module_uri:nn { #1 } { #2 }
2120     \stex_import_require_module:nnnn
2121     { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```

```

2122     { \l_stex_import_path_str } { \l_stex_import_name_str }
2123     \stex_annotate_invisible:nnn
2124     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2125   }
2126   \stex_smsmode_do:
2127   \ignorespacesandpars
2128 }

```

(End definition for \usemodule. This function is documented on page 76.)

```

2129 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2130   \tl_if_empty:nF{#2}{
2131     \clist_set:Nn \l_tmpa_clist {#2}
2132     \clist_map_inline:Nn \l_tmpa_clist {
2133       \tl_if_head_eq_charcode:nNTF {##1} [{
2134         #1 ##1
2135       } {
2136         #1{##1}
2137       }
2138     }
2139   }
2140 }
2141 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2142
2143
2144 \endpackage

```

Chapter 28

STEX -Symbols Implementation

```
2145 <*package>
2146
2147 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2148
2149 \msg_new:nnn{stex}{error/wrongargs}{
2150   args~value~in~symbol~declaration~for~#1~
2151   needs~to~be~i,~a,~b~or~B,~but~#2~given
2152 }
2153 \msg_new:nnn{stex}{error/unknownsymbol}{
2154   No~symbol~#1~found!
2155 }
2156 \msg_new:nnn{stex}{error/seqlength}{
2157   Expected~#1~arguments;~got~#2!
2158 }
2159 \msg_new:nnn{stex}{error/unknownnotation}{
2160   Unknown~notation~#1~for~#2!
2161 }
```

28.1 Symbol Declarations

```
2162 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2163 \cs_new_protected:Nn \stex_all_symbols:n {
2164   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2165   \seq_map_inline:Nn \l_stex_all_modules_seq {
2166     \seq_map_inline:cn{c_stex_module_##1_constants}{
2167       \__stex_symdecl_all_symbols_cs{##1?####1}
2168     }
2169   }
2170 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page 79.)

\STEXsymbol

```
2171 \NewDocumentCommand \STEXsymbol { m } {
2172   \stex_get_symbol:n { #1 }
2173   \exp_args:No
2174   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2175 }
```

(End definition for \STEXsymbol. This function is documented on page 80.)

symdecl arguments:

```
2176 \keys_define:nn { stex / symdecl } {
2177   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2178   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2179   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2180   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2181   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2182   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2183   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2184   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2185   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2186   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2187   assoc     .choices:nn =
2188     {bin,binl,binr,pre,conj,pwconj}
2189     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2190 }
2191
2192 \bool_new:N \l_stex_symdecl_make_macro_bool
2193
2194 \cs_new_protected:Nn \__stex_symdecl_args:n {
2195   \str_clear:N \l_stex_symdecl_name_str
2196   \str_clear:N \l_stex_symdecl_args_str
2197   \str_clear:N \l_stex_symdecl_deprecate_str
2198   \str_clear:N \l_stex_symdecl_reorder_str
2199   \str_clear:N \l_stex_symdecl_assoctype_str
2200   \bool_set_false:N \l_stex_symdecl_local_bool
2201   \tl_clear:N \l_stex_symdecl_type_tl
2202   \tl_clear:N \l_stex_symdecl_definiens_tl
2203
2204   \keys_set:nn { stex / symdecl } { #1 }
2205 }
```

\symdecl Parses the optional arguments and passes them on to \stex_symdecl_do: (so that \symdef can do the same)

```
2206
2207 \NewDocumentCommand \symdecl { s m O{} } {
2208   \__stex_symdecl_args:n { #3 }
2209   \IfBooleanTF #1 {
2210     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2211   } {
2212     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2213   }
2214   \stex_symdecl_do:n { #2 }
2215   \stex_smsmode_do:
2216 }
```

```

2217
2218 \cs_new_protected:Nn \stex_symdecl_do:nn {
2219   \__stex_symdecl_args:n{#1}
2220   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2221   \stex_symdecl_do:n{#2}
2222 }
2223
2224 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2225 \cs_new_protected:Nn \stex_symdecl_do:n {
2226   \stex_if_in_module:F {
2227     % TODO throw error? some default namespace?
2228   }
2229
2230   \str_if_empty:NT \l_stex_symdecl_name_str {
2231     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2232   }
2233
2234   \prop_if_exist:cT { l_stex_symdecl_
2235     \l_stex_current_module_str ?
2236     \l_stex_symdecl_name_str
2237     _prop
2238   }{
2239     % TODO throw error (beware of circular dependencies)
2240   }
2241
2242   \prop_clear:N \l_tmpa_prop
2243   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2244   \seq_clear:N \l_tmpa_seq
2245   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2246   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2247
2248   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2249     \str_if_empty:NF \l_stex_module_deprecate_str {
2250       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2251     }
2252   }
2253   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2254
2255   \exp_args:No \stex_add_constant_to_current_module:n {
2256     \l_stex_symdecl_name_str
2257   }
2258
2259   % arity/args
2260   \int_zero:N \l_tmpb_int
2261
2262   \bool_set_true:N \l_tmpa_bool
2263   \str_map_inline:Nn \l_stex_symdecl_args_str {
2264     \token_case_meaning:NnF ##1 {
2265       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2266       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```



```

2267     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2268     {\tl_to_str:n a} {
2269         \bool_set_false:N \l_tmpa_bool
2270         \int_incr:N \l_tmpb_int
2271     }
2272     {\tl_to_str:n B} {
2273         \bool_set_false:N \l_tmpa_bool
2274         \int_incr:N \l_tmpb_int
2275     }
2276 }{
2277     \msg_error:nnxx{stex}{error/wrongargs}{
2278         \l_stex_current_module_str ?
2279         \l_stex_symdecl_name_str
2280     }{##1}
2281 }
2282 }
2283 \bool_if:NTF \l_tmpa_bool {
2284     % possibly numeric
2285     \str_if_empty:NTF \l_stex_symdecl_args_str {
2286         \prop_put:Nnn \l_tmpa_prop { args } {}
2287         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2288     }{
2289         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2290         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2291         \str_clear:N \l_tmpa_str
2292         \int_step_inline:nn \l_tmpa_int {
2293             \str_put_right:Nn \l_tmpa_str i
2294         }
2295         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2296     }
2297 } {
2298     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2299     \prop_put:Nnx \l_tmpa_prop { arity }
2300     { \str_count:N \l_stex_symdecl_args_str }
2301 }
2302 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2303
2304 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2305     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2306 }{
2307     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2308 }
2309
2310 % semantic macro
2311
2312 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2313     \exp_args:Nx \stex_do_up_to_module:n {
2314         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2315             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2316         }}
2317     }
2318 }
2319
2320 \stex_debug:nn{symbols}{New~symbol:~

```

```

2321 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2322 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2323 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2324 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2325 }
2326
2327 % circular dependencies require this:
2328 \stex_if_do_html:T {
2329   \stex_annotate_invisible:nnn {symdecl} {
2330     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2331   } {
2332     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2333       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2334     }
2335     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{ }
2336     \stex_annotate_invisible:nnn{macroname}{#1}{ }
2337     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2338       \stex_annotate_invisible:nnn{definiens}{ }
2339       {\l_stex_symdecl_definiens_tl$}
2340     }
2341     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2342       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{ }
2343     }
2344     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2345       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{ }
2346     }
2347   }
2348 }
2349 \prop_if_exist:cF {
2350   \l_stex_symdecl_
2351   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2352   _prop
2353 } {
2354   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2355   \__stex_symdecl_restore_symbol:nnnnnnn
2356   {\l_stex_symdecl_name_str}
2357   { \prop_item:Nn \l_tmpa_prop {args} }
2358   { \prop_item:Nn \l_tmpa_prop {arity} }
2359   { \prop_item:Nn \l_tmpa_prop {assocs} }
2360   { \prop_item:Nn \l_tmpa_prop {defined} }
2361   {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2362   {\l_stex_current_module_str}
2363 }
2364 }
2365 }
2366 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2367   \prop_clear:N \l_tmpa_prop
2368   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2369   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2370   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2371   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2372   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2373   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2374   \tl_if_empty:nF{#6}{

```

```

2375     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2376   }
2377   \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2378   \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2379 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 79.)

`\textsymdecl`

```

2380
2381 \keys_define:nn { stex / textsymdecl } {
2382   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2383   type      .tl_set:N    = \l__stex_symdecl_type_tl
2384 }
2385
2386 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2387   \str_clear:N \l__stex_symdecl_name_str
2388   \tl_clear:N \l__stex_symdecl_type_tl
2389   \keys_set:nn { stex / textsymdecl } { #1 }
2390 }
2391
2392 \NewDocumentCommand \textsymdecl {m O{} m} {
2393   \_stex_textsymdecl_args:n { #2 }
2394   \str_if_empty:NTF \l__stex_symdecl_name_str {
2395     \__stex_symdecl_args:n{name=#1,#2}
2396   }{
2397     \__stex_symdecl_args:n{#2}
2398   }
2399   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2400   \stex_symdecl_do:n{#1-sym}
2401   \stex_execute_in_module:n{
2402     \cs_set_nopar:cpn{#1name}{
2403       \ifvmode\hbox_unpack:N\c_empty_box\fi
2404       \ifmmode\hbox{#3}\else#3\fi\hspace
2405     }
2406     \cs_set_nopar:cpn{#1}{
2407       \ifmmode\csname#1-sym\expandafter\endcsname\else
2408       \ifvmode\hbox_unpack:N\c_empty_box\fi
2409       \symref{#1-sym}{#3}\expandafter\hspace
2410       \fi
2411     }
2412   }
2413   \stex_execute_in_module:x{
2414     \__stex_notation_restore_notation:nnnnn
2415     {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl
2416     }{0}
2417     {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{
2418       \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2419     }}
2420   }
2421 }
2422 \stex_smsmode_do:
2423 }

```

(End definition for `\textsymdecl`. This function is documented on page 19.)

`\stex_get_symbol:n`

```
2424 \str_new:N \l_stex_get_symbol_uri_str
2425
2426 \cs_new_protected:Nn \stex_get_symbol:n {
2427   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2428     \tl_set:Nn \l_tmpa_tl { #1 }
2429     \__stex_symdecl_get_symbol_from_cs:
2430   }{
2431     % argument is a string
2432     % is it a command name?
2433     \cs_if_exist:cTF { #1 }{
2434       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2435       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2436       \str_if_empty:NTF \l_tmpa_str {
2437         \exp_args:Nx \cs_if_eq:NNTF {
2438           \tl_head:N \l_tmpa_tl
2439         } \stex_invoke_symbol:n {
2440           \__stex_symdecl_get_symbol_from_cs:
2441         }{
2442           \__stex_symdecl_get_symbol_from_string:n { #1 }
2443         }
2444       } {
2445         \__stex_symdecl_get_symbol_from_string:n { #1 }
2446       }
2447     }{
2448       % argument is not a command name
2449       \__stex_symdecl_get_symbol_from_string:n { #1 }
2450       % \l_stex_all_symbols_seq
2451     }
2452   }
2453   \str_if_eq:eeF {
2454     \prop_item:cn {
2455       l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2456     }{ deprecate }
2457   }{}{
2458     \msg_warning:nxxx{stex}{warning/deprecated}{
2459       Symbol~\l_stex_get_symbol_uri_str
2460     }{
2461       \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2462     }
2463   }
2464 }
2465
2466 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2467   \tl_set:Nn \l_tmpa_tl {
2468     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2469   }
2470   \str_set:Nn \l_tmpa_str { #1 }
2471
2472   %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2473
2474   \str_if_in:NnTF \l_tmpa_str ? {
2475     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2476     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```

```

2477 \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2478 }{
2479 \str_clear:N \l_tmpb_str
2480 }
2481 \str_if_empty:NTF \l_tmpb_str {
2482 \seq_map_inline:Nn \l_stex_all_modules_seq {
2483 \seq_map_inline:cn{c_stex_module_###1_constants}{
2484 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2485 \seq_map_break:n{\seq_map_break:n{
2486 \tl_set:Nn \l_tmpa_tl {
2487 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2488 }
2489 }}
2490 }
2491 }
2492 }
2493 }{
2494 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2495 \seq_map_inline:Nn \l_stex_all_modules_seq {
2496 \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2497 \seq_map_inline:cn{c_stex_module_###1_constants}{
2498 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2499 \seq_map_break:n{\seq_map_break:n{
2500 \tl_set:Nn \l_tmpa_tl {
2501 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2502 }
2503 }}
2504 }
2505 }
2506 }
2507 }
2508 }
2509
2510 \l_tmpa_tl
2511 }
2512
2513 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2514 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2515 { \tl_tail:N \l_tmpa_tl }
2516 \tl_if_single:NTF \l_tmpa_tl {
2517 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2518 \exp_after:wN \str_set:Nn \exp_after:wN
2519 \l_stex_get_symbol_uri_str \l_tmpa_tl
2520 }{
2521 % TODO
2522 % tail is not a single group
2523 }
2524 }{
2525 % TODO
2526 % tail is not a single group
2527 }
2528 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 79.)

28.2 Notations

```

2529 <@@=stex_notation>

      notation arguments:
2530 \keys_define:nn { stex / notation } {
2531 % lang .tl_set_x:N = \l__stex_notation_lang_str ,
2532 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2533 prec .str_set_x:N = \l__stex_notation_prec_str ,
2534 op .tl_set:N = \l__stex_notation_op_tl ,
2535 primary .bool_set:N = \l__stex_notation_primary_bool ,
2536 primary .default:n = {true} ,
2537 unknown .code:n = \str_set:Nx
2538     \l__stex_notation_variant_str \l_keys_key_str
2539 }
2540
2541 \cs_new_protected:Nn \stex_notation_args:n {
2542 % \str_clear:N \l__stex_notation_lang_str
2543 \str_clear:N \l__stex_notation_variant_str
2544 \str_clear:N \l__stex_notation_prec_str
2545 \tl_clear:N \l__stex_notation_op_tl
2546 \bool_set_false:N \l__stex_notation_primary_bool
2547
2548 \keys_set:nn { stex / notation } { #1 }
2549 }

\notation

2550 \NewDocumentCommand \notation { s m O{}} {
2551 \stex_notation_args:n { #3 }
2552 \tl_clear:N \l_stex_symdecl_definiens_tl
2553 \stex_get_symbol:n { #2 }
2554 \tl_set:Nn \l_stex_notation_after_do_tl {
2555     \__stex_notation_final:
2556     \IfBooleanTF#1{
2557         \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2558     }{}
2559     \stex_smsmode_do:\ignorespacesandpars
2560 }
2561 \stex_notation_do:nnnnn
2562 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { args } }
2563 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
2564 { \l__stex_notation_variant_str }
2565 { \l__stex_notation_prec_str }
2566 }
2567 \stex_deactivate_macro:Nn \notation {module-environments}

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

\stex_notation_do:nnnnn

2568 \seq_new:N \l__stex_notation_precedences_seq
2569 \tl_new:N \l__stex_notation_opprec_tl
2570 \int_new:N \l__stex_notation_currarg_int
2571 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2572
2573 \cs_new_protected:Nn \stex_notation_do:nnnnn {

```

```

2574 \let\STEXInternalCurrentSymbolStr\relax
2575 \seq_clear:N \l__stex_notation_precedences_seq
2576 \tl_clear:N \l__stex_notation_opprec_tl
2577 \str_set:Nx \l__stex_notation_args_str { #1 }
2578 \str_set:Nx \l__stex_notation_arity_str { #2 }
2579 \str_set:Nx \l__stex_notation_suffix_str { #3 }
2580 \str_set:Nx \l__stex_notation_prec_str { #4 }
2581
2582 % precedences
2583 \str_if_empty:NTF \l__stex_notation_prec_str {
2584   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2585     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2586   }{
2587     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2588   }
2589 } {
2590   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2591     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2592     \int_step_inline:nn { \l__stex_notation_arity_str } {
2593       \exp_args:NNo
2594       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2595     }
2596   }{
2597     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2598     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2599       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2600       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2601         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2602           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2603         \seq_map_inline:Nn \l_tmpa_seq {
2604           \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }
2605         }
2606       }
2607     }{
2608       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2609         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2610       }{
2611         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2612       }
2613     }
2614   }
2615 }
2616
2617 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2618 \int_step_inline:nn { \l__stex_notation_arity_str } {
2619   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2620     \exp_args:NNo
2621     \seq_put_right:No \l__stex_notation_precedences_seq {
2622       \l__stex_notation_opprec_tl
2623     }
2624   }
2625 }
2626 \tl_clear:N \l__stex_notation_dummyargs_tl
2627

```

```

2628 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2629   \exp_args:NNe
2630   \cs_set:Npn \l_stex_notation_macrocode_cs {
2631     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2632     { \l__stex_notation_suffix_str }
2633     { \l__stex_notation_opprec_tl }
2634     { \exp_not:n { #5 } }
2635   }
2636   \l_stex_notation_after_do_tl
2637 }{
2638   \str_if_in:NnTF \l__stex_notation_args_str b {
2639     \exp_args:Nne \use:nn
2640     {
2641       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2642       \cs_set:Npn \l__stex_notation_arity_str } { {
2643         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2644         { \l__stex_notation_suffix_str }
2645         { \l__stex_notation_opprec_tl }
2646         { \exp_not:n { #5 } }
2647       }}
2648   }{
2649     \str_if_in:NnTF \l__stex_notation_args_str B {
2650       \exp_args:Nne \use:nn
2651       {
2652         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2653         \cs_set:Npn \l__stex_notation_arity_str } { {
2654           \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2655           { \l__stex_notation_suffix_str }
2656           { \l__stex_notation_opprec_tl }
2657           { \exp_not:n { #5 } }
2658         } }
2659     }{
2660       \exp_args:Nne \use:nn
2661       {
2662         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2663         \cs_set:Npn \l__stex_notation_arity_str } { {
2664           \STEXInternalTermMathOMAIiiii { \STEXInternalCurrentSymbolStr }
2665           { \l__stex_notation_suffix_str }
2666           { \l__stex_notation_opprec_tl }
2667           { \exp_not:n { #5 } }
2668         } }
2669     }
2670   }
2671
2672   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2673   \int_zero:N \l__stex_notation_currarg_int
2674   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2675   \__stex_notation_arguments:
2676 }
2677 }

```

(End definition for \stex_notation_do:nnnnn. This function is documented on page ??.)

`__stex_notation_arguments:` Takes care of annotating the arguments in a notation macro


```

2678 \cs_new_protected:Nn \__stex_notation_arguments: {
2679   \int_incr:N \l__stex_notation_currarg_int
2680   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2681     \l_stex_notation_after_do_tl
2682   }{
2683     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2684     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2685     \str_if_eq:VnTF \l_tmpa_str a {
2686       \__stex_notation_argument_assoc:nn{a}
2687     }{
2688       \str_if_eq:VnTF \l_tmpa_str B {
2689         \__stex_notation_argument_assoc:nn{B}
2690       }{
2691         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2692         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2693           { \STEXInternalTermMathArgiii
2694             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2695             { \l_tmpb_str }
2696             { ####\int_use:N \l__stex_notation_currarg_int }
2697           }
2698         }
2699         \__stex_notation_arguments:
2700       }
2701     }
2702   }
2703 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```

2704 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2705
2706   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2707     {\l__stex_notation_arity_str}{
2708       #2
2709     }
2710   \int_zero:N \l_tmpa_int
2711   \tl_clear:N \l_tmpa_tl
2712   \str_map_inline:Nn \l__stex_notation_args_str {
2713     \int_incr:N \l_tmpa_int
2714     \tl_put_right:Nx \l_tmpa_tl {
2715       \str_if_eq:nnTF {##1}{a}{ {} }{
2716         \str_if_eq:nnTF {##1}{B}{ {} }{
2717           {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2718         }
2719       }
2720     }
2721   }
2722   \exp_after:wN\exp_after:wN\exp_after:wN \def
2723   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2724   \exp_after:wN\exp_after:wN\exp_after:wN ##
2725   \exp_after:wN\exp_after:wN\exp_after:wN 1
2726   \exp_after:wN\exp_after:wN\exp_after:wN ##
2727   \exp_after:wN\exp_after:wN\exp_after:wN 2

```

```

2728 \exp_after:wN\exp_after:wN\exp_after:wN {
2729   \exp_after:wN \exp_after:wN \exp_after:wN
2730   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2731     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2732   }
2733 }
2734
2735 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2736 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2737   \STEXInternalTermMathAssocArgiiii
2738   { #1\int_use:N \l__stex_notation_currarg_int }
2739   { \l_tmpa_str }
2740   { ####\int_use:N \l__stex_notation_currarg_int }
2741   { \l_tmpa_cs {####1} {####2} }
2742 } }
2743 \__stex_notation_arguments:
2744 }

```

(End definition for __stex_notation_argument_assoc:nn.)

__stex_notation_final: Called after processing all notation arguments

```

2745 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2746   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2747   \cs_set_nopar:Npn {#3}{#4}
2748   \tl_if_empty:nF {#5}{
2749     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2750   }
2751   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2752     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2753   }
2754 }
2755
2756 \cs_new_protected:Nn \__stex_notation_final: {
2757
2758   \stex_execute_in_module:x {
2759     \__stex_notation_restore_notation:nnnnn
2760     {\l_stex_get_symbol_uri_str}
2761     {\l__stex_notation_suffix_str}
2762     {\l__stex_notation_arity_str}
2763     {
2764       \exp_after:wN \exp_after:wN \exp_after:wN
2765       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2766       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
2767     }
2768     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2769   }
2770
2771   \stex_debug:nn{symbols}{
2772     Notation~\l__stex_notation_suffix_str
2773     ~for~\l_stex_get_symbol_uri_str^^J
2774     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2775     Argument~precedences:~
2776     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2777     Notation: \cs_meaning:c {

```

```

2778     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2779     \l__stex_notation_suffix_str
2780     _cs
2781 }
2782 }
2783 % HTML annotations
2784 \stex_if_do_html:T {
2785   \stex_annotate_invisible:nnn { notation }
2786   { \l_stex_get_symbol_uri_str } {
2787     \stex_annotate_invisible:nnn { notationfragment }
2788     { \l__stex_notation_suffix_str }{}
2789     \stex_annotate_invisible:nnn { precedence }
2790     { \l__stex_notation_prec_str }{}
2791
2792     \int_zero:N \l_tmpa_int
2793     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2794     \tl_clear:N \l_tmpa_tl
2795     \int_step_inline:nn { \l__stex_notation_arity_str }{
2796       \int_incr:N \l_tmpa_int
2797       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2798       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
2799       \str_if_eq:VnTF \l_tmpb_str a {
2800         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2801           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2802           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2803         } }
2804       }{
2805         \str_if_eq:VnTF \l_tmpb_str B {
2806           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2807             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2808             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2809           } }
2810         }{
2811           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2812             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2813           } }
2814         }
2815       }
2816     }
2817     \stex_annotate_invisible:nnn { notationcomp }{}{
2818       \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
2819       $ \exp_args:Nno \use:nn { \use:c {
2820         stex_notation_ \STEXInternalCurrentSymbolStr
2821         \c_hash_str \l__stex_notation_suffix_str _cs
2822       } } { \l_tmpa_tl } $
2823     }
2824     \tl_if_empty:NF \l__stex_notation_op_tl {
2825       \stex_annotate_invisible:nnn { notationopcomp }{}{
2826         $\l__stex_notation_op_tl$
2827       }
2828     }
2829   }
2830 }
2831 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2832 \keys_define:nn { stex / setnotation } {
2833   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2834   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2835   unknown .code:n      = \str_set:Nx
2836     \l__stex_notation_variant_str \l_keys_key_str
2837 }
2838
2839 \cs_new_protected:Nn \stex_setnotation_args:n {
2840   % \str_clear:N \l__stex_notation_lang_str
2841   \str_clear:N \l__stex_notation_variant_str
2842   \keys_set:nn { stex / setnotation } { #1 }
2843 }
2844
2845 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
2846   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2847     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2848     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2849   }
2850 }
2851
2852 \cs_new_protected:Nn \stex_setnotation:n {
2853   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2854     { \l__stex_notation_variant_str }{
2855     \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2856     \stex_debug:nn {notations}{
2857       Setting~default~notation~
2858       {\l__stex_notation_variant_str }~for~
2859       #1 \\
2860       \expandafter\meaning\csname
2861       l_stex_symdecl_#1_notations\endcsname
2862     }
2863   }{
2864     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2865   }
2866 }
2867
2868 \NewDocumentCommand \setnotation {m m} {
2869   \stex_get_symbol:n { #1 }
2870   \_stex_setnotation_args:n { #2 }
2871   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2872   \stex_smsmode_do:\ignorespacesandpars
2873 }
2874
2875 \cs_new_protected:Nn \stex_copy_notations:nn {
2876   \stex_debug:nn {notations}{
2877     Copying~notations~from~#2~to~#1\\
2878     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2879   }
2880   \tl_clear:N \l_tmpa_tl
2881   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2882     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }

```

```

2883 }
2884 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
2885   \stex_debug:nn{Here}{Here:~##1}
2886   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2887   \edef \l_tmpa_tl {
2888     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2889     \exp_after:wN\exp_after:wN\exp_after:wN {
2890       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2891     }
2892   }
2893
2894   \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2895   \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2896   \exp_after:wN { \l_tmpa_tl }
2897
2898   \edef \l_tmpa_tl {
2899     \exp_after:wN \exp_not:n \exp_after:wN {
2900       \l_tmpa_tl {##### 1}{##### 2}
2901     }
2902   }
2903
2904   \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
2905
2906   \stex_execute_in_module:x {
2907     \__stex_notation_restore_notation:nnnnn
2908     {#1}{##1}
2909     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2910     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2911     {
2912       \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2913         \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2914       }
2915     }
2916   }\endgroup
2917 }
2918 }
2919
2920 \NewDocumentCommand \copynotation {m m} {
2921   \stex_get_symbol:n { #1 }
2922   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2923   \stex_get_symbol:n { #2 }
2924   \exp_args:Noo
2925   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2926   \stex_smsmode_do:\ignorespacesandpars
2927 }
2928

```

(End definition for \setnotation. This function is documented on page 19.)

\symdef

```

2929 \keys_define:nn { stex / symdef } {
2930   name .str_set_x:N = \l_stex_symdecl_name_str ,
2931   local .bool_set:N = \l_stex_symdecl_local_bool ,
2932   args .str_set_x:N = \l_stex_symdecl_args_str ,

```

```

2933 type .tl_set:N = \l_stex_symdecl_type_tl ,
2934 def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2935 reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
2936 op .tl_set:N = \l__stex_notation_op_tl ,
2937 % lang .str_set_x:N = \l__stex_notation_lang_str ,
2938 variant .str_set_x:N = \l__stex_notation_variant_str ,
2939 prec .str_set_x:N = \l__stex_notation_prec_str ,
2940 assoc .choices:nn =
2941 {bin,binl,binr,pre,conj,pwconj}
2942 {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2943 unknown .code:n = \str_set:Nx
2944 \l__stex_notation_variant_str \l_keys_key_str
2945 }
2946
2947 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2948 \str_clear:N \l_stex_symdecl_name_str
2949 \str_clear:N \l_stex_symdecl_args_str
2950 \str_clear:N \l_stex_symdecl_assoctype_str
2951 \str_clear:N \l_stex_symdecl_reorder_str
2952 \bool_set_false:N \l_stex_symdecl_local_bool
2953 \tl_clear:N \l_stex_symdecl_type_tl
2954 \tl_clear:N \l_stex_symdecl_definiens_tl
2955 % \str_clear:N \l__stex_notation_lang_str
2956 \str_clear:N \l__stex_notation_variant_str
2957 \str_clear:N \l__stex_notation_prec_str
2958 \tl_clear:N \l__stex_notation_op_tl
2959
2960 \keys_set:nn { stex / symdef } { #1 }
2961 }
2962
2963 \NewDocumentCommand \symdef { m O{} } {
2964 \__stex_notation_symdef_args:n { #2 }
2965 \bool_set_true:N \l_stex_symdecl_make_macro_bool
2966 \stex_symdecl_do:n { #1 }
2967 \tl_set:Nn \l_stex_notation_after_do_tl {
2968 \__stex_notation_final:
2969 \stex_smsmode_do:\ignorespacesandpars
2970 }
2971 \str_set:Nx \l_stex_get_symbol_uri_str {
2972 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2973 }
2974 \exp_args:Nx \stex_notation_do:nnnnn
2975 { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2976 { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2977 { \l__stex_notation_variant_str }
2978 { \l__stex_notation_prec_str }
2979 }
2980 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

28.3 Variables

```

2981 <@@=stex_variables>

```

```

2982
2983 \keys_define:nn { stex / vardef } {
2984   name .str_set_x:N = \l__stex_variables_name_str ,
2985   args .str_set_x:N = \l__stex_variables_args_str ,
2986   type .tl_set:N     = \l__stex_variables_type_tl ,
2987   def .tl_set:N      = \l__stex_variables_def_tl ,
2988   op .tl_set:N       = \l__stex_variables_op_tl ,
2989   prec .str_set_x:N  = \l__stex_variables_prec_str ,
2990   reorder .str_set_x:N = \l__stex_variables_reorder_str ,
2991   assoc .choices:nn =
2992     {bin,binl,binr,pre,conj,pwconj}
2993     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2994   bind .choices:nn =
2995     {forall,exists}
2996     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2997 }
2998
2999 \cs_new_protected:Nn \__stex_variables_args:n {
3000   \str_clear:N \l__stex_variables_name_str
3001   \str_clear:N \l__stex_variables_args_str
3002   \str_clear:N \l__stex_variables_prec_str
3003   \str_clear:N \l__stex_variables_assoctype_str
3004   \str_clear:N \l__stex_variables_reorder_str
3005   \str_clear:N \l__stex_variables_bind_str
3006   \tl_clear:N \l__stex_variables_type_tl
3007   \tl_clear:N \l__stex_variables_def_tl
3008   \tl_clear:N \l__stex_variables_op_tl
3009
3010   \keys_set:nn { stex / vardef } { #1 }
3011 }
3012
3013 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3014   \__stex_variables_args:n {#2}
3015   \str_if_empty:NT \l__stex_variables_name_str {
3016     \str_set:Nx \l__stex_variables_name_str { #1 }
3017   }
3018   \prop_clear:N \l_tmpa_prop
3019   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3020
3021   \int_zero:N \l_tmpb_int
3022   \bool_set_true:N \l_tmpa_bool
3023   \str_map_inline:Nn \l__stex_variables_args_str {
3024     \token_case_meaning:NnF ##1 {
3025       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3026       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3027       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3028       {\tl_to_str:n a} {
3029         \bool_set_false:N \l_tmpa_bool
3030         \int_incr:N \l_tmpb_int
3031       }
3032       {\tl_to_str:n B} {
3033         \bool_set_false:N \l_tmpa_bool
3034         \int_incr:N \l_tmpb_int
3035       }

```

```

3036   }{
3037     \msg_error:nnxx{stex}{error/wrongargs}{
3038       variable~\l__stex_variables_name_str
3039     }{##1}
3040   }
3041 }
3042 \bool_if:NTF \l_tmpa_bool {
3043   % possibly numeric
3044   \str_if_empty:NTF \l__stex_variables_args_str {
3045     \prop_put:Nnn \l_tmpa_prop { args } {}
3046     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3047   }{
3048     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3049     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3050     \str_clear:N \l_tmpa_str
3051     \int_step_inline:nn \l_tmpa_int {
3052       \str_put_right:Nn \l_tmpa_str i
3053     }
3054     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3055     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3056   }
3057 } {
3058   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3059   \prop_put:Nnx \l_tmpa_prop { arity }
3060   { \str_count:N \l__stex_variables_args_str }
3061 }
3062 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3063 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3064
3065 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
3066
3067 \tl_if_empty:NF \l__stex_variables_op_tl {
3068   \cs_set:cpx {
3069     stex_var_op_notation_\l__stex_variables_name_str _cs
3070   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3071 }
3072
3073 \tl_set:Nn \l_stex_notation_after_do_tl {
3074   \exp_args:Nne \use:nn {
3075     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3076     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3077   } {{
3078     \exp_after:wN \exp_after:wN \exp_after:wN
3079     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3080     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3081   }}
3082 \stex_if_do_html:T {
3083   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3084     \stex_annotate_invisible:nnn { precedence }
3085     { \l__stex_variables_prec_str }{}
3086     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l
3087     \stex_annotate_invisible:nnn{args}{ \l__stex_variables_args_str }{}
3088     \stex_annotate_invisible:nnn{macroname}{#1}{}
3089     \tl_if_empty:NF \l__stex_variables_def_tl {

```



```

3090     \stex_annotate_invisible:nnn{definiens}{}
3091     {${\l__stex_variables_def_tl$}
3092   }
3093   \str_if_empty:NF \l__stex_variables_assoctype_str {
3094     \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3095   }
3096   \str_if_empty:NF \l__stex_variables_reorder_str {
3097     \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3098   }
3099   \int_zero:N \l_tmpa_int
3100   \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3101   \tl_clear:N \l_tmpa_tl
3102   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3103     \int_incr:N \l_tmpa_int
3104     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3105     \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3106     \str_if_eq:VnTF \l_tmpb_str a {
3107       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3108         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3109         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3110       } }
3111     }{
3112       \str_if_eq:VnTF \l_tmpb_str B {
3113         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3114           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3115           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3116         } }
3117       }{
3118         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3119           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3120         } }
3121       }
3122     }
3123   }
3124   \stex_annotate_invisible:nnn { notationcomp }{}{
3125     \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3126     $ \exp_args:Nno \use:nn { \use:c {
3127       stex_var_notation_\l__stex_variables_name_str _cs
3128     } } { \l_tmpa_tl } $
3129   }
3130   \tl_if_empty:NF \l__stex_variables_op_tl {
3131     \stex_annotate_invisible:nnn { notationopcomp }{}{
3132       ${\l__stex_variables_op_tl$
3133     }
3134   }
3135   }
3136   \str_if_empty:NF \l__stex_variables_bind_str {
3137     \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3138   }
3139   }\ignorespacesandpars
3140 }
3141
3142 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3143 }

```

```

3144
3145 \cs_new:Nn \_stex_reset:N {
3146   \tl_if_exist:NTF #1 {
3147     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3148   }{
3149     \let \exp_not:N #1 \exp_not:N \undefined
3150   }
3151 }
3152
3153 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3154   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3155   \exp_args:Nnx \use:nn {
3156     % TODO
3157     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3158       #2
3159     }
3160   }{
3161     \_stex_reset:N \varnot
3162     \_stex_reset:N \vartype
3163     \_stex_reset:N \vardefi
3164   }
3165 }
3166
3167 \NewDocumentCommand \vardef { s } {
3168   \IfBooleanTF#1 {
3169     \__stex_variables_do_complex:nn
3170   }{
3171     \__stex_variables_do_simple:nnn
3172   }
3173 }
3174
3175 \NewDocumentCommand \svar { 0{} m }{
3176   \tl_if_empty:nTF {#1}{
3177     \str_set:Nn \l_tmpa_str { #2 }
3178   }{
3179     \str_set:Nn \l_tmpa_str { #1 }
3180   }
3181   \_stex_term_omv:nn {
3182     var://\l_tmpa_str
3183   }{
3184     \exp_args:Nnx \use:nn {
3185       \def\comp{\_varcomp}
3186       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3187       \comp{ #2 }
3188     }{
3189       \_stex_reset:N \comp
3190       \_stex_reset:N \STEXInternalCurrentSymbolStr
3191     }
3192   }
3193 }
3194
3195
3196
3197 \keys_define:nn { stex / varseq } {

```

```

3198 name .str_set_x:N = \l__stex_variables_name_str ,
3199 args .int_set:N = \l__stex_variables_args_int ,
3200 type .tl_set:N = \l__stex_variables_type_tl ,
3201 mid .tl_set:N = \l__stex_variables_mid_tl ,
3202 bind .choices:nn =
3203 {forall,exists}
3204 {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3205 }
3206
3207 \cs_new_protected:Nn \l__stex_variables_seq_args:n {
3208 \str_clear:N \l__stex_variables_name_str
3209 \int_set:Nn \l__stex_variables_args_int 1
3210 \tl_clear:N \l__stex_variables_type_tl
3211 \str_clear:N \l__stex_variables_bind_str
3212
3213 \keys_set:nn { stex / varseq } { #1 }
3214 }
3215
3216 \NewDocumentCommand \varseq {m O{} m m m}{
3217 \l__stex_variables_seq_args:n { #2 }
3218 \str_if_empty:NT \l__stex_variables_name_str {
3219 \str_set:Nx \l__stex_variables_name_str { #1 }
3220 }
3221 \prop_clear:N \l_tmpa_prop
3222 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3223
3224 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3225 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3226 \msg_error:nnxx{stex}{error/seqlength}
3227 {\int_use:N \l__stex_variables_args_int}
3228 {\seq_count:N \l_tmpa_seq}
3229 }
3230 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3231 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3232 \msg_error:nnxx{stex}{error/seqlength}
3233 {\int_use:N \l__stex_variables_args_int}
3234 {\seq_count:N \l_tmpb_seq}
3235 }
3236 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3237 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3238
3239 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3240 \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3241
3242 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3243 \int_step_inline:nn \l__stex_variables_args_int {
3244 \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3245 }
3246 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3247 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3248 \tl_if_empty:NF \l__stex_variables_mid_tl {
3249 \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3250 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3251 }

```

```

3252 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3253 \int_step_inline:nn \l__stex_variables_args_int {
3254   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3255 }
3256 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3257 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3258
3259
3260 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3261
3262 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3263
3264 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3265
3266 \int_step_inline:nn \l__stex_variables_args_int {
3267   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3268     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3269   }}
3270 }
3271
3272 \tl_set:Nx \l_tmpa_tl {
3273   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3274     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3275   }
3276 }
3277
3278 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3279
3280 \exp_args:Nno \use:nn {
3281   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3282   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3283
3284   \stex_debug:nn{sequences}{New~Sequence:~
3285     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3286     \prop_to_keyval:N \l_tmpa_prop
3287   }
3288   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3289     \tl_if_empty:NF \l__stex_variables_type_tl {
3290       \stex_annotate:nnn {type}{\l__stex_variables_type_tl$}
3291     }
3292     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3293     \str_if_empty:NF \l__stex_variables_bind_str {
3294       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3295     }
3296     \stex_annotate:nnn{startindex}{\l__stex_variables_startindex}{\l__stex_variables_startindex$}
3297     \stex_annotate:nnn{endindex}{\l__stex_variables_endindex}{\l__stex_variables_endindex$}
3298
3299     \tl_clear:N \l_tmpa_tl
3300     \int_step_inline:nn \l__stex_variables_args_int {
3301       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3302         \stex_annotate:nnn{argmarker}{##1}{}
3303       } }
3304     }
3305     \stex_annotate_invisible:nnn { notationcomp }{}{

```

```

3306     \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3307     $ \exp_args:Nno \use:nn { \use:c {
3308         stex_varseq_\l__stex_variables_name_str _cs
3309     } } { \l_tmpa_tl } $
3310 }
3311 \stex_annotate_invisible:nnn { notationopcomp }{}{
3312     $ \prop_item:Nn \l_tmpa_prop { notation } $
3313 }
3314
3315 }}
3316
3317 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3318 \ignorespacesandpars
3319 }
3320
3321 \end{package}

```

Chapter 29

STEX -Terms Implementation

```
3322 <*package>
3323
3324 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3325
3326 <@@=stex_terms>
3327
3328 Warnings and error messages
3329 \msg_new:nnn{stex}{error/nonotation}{
3330   Symbol~#1~invoked,~but~has~no~notation#2!
3331 }
3332 \msg_new:nnn{stex}{error/notationarg}{
3333   Error~in~parsing~notation~#1
3334 }
3335 \msg_new:nnn{stex}{error/noop}{
3336   Symbol~#1~has~no~operator~notation~for~notation~#2
3337 }
3338 \msg_new:nnn{stex}{error/notallowed}{
3339   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3340 }
3341 \msg_new:nnn{stex}{error/doubleargument}{
3342   Argument~#1~of~symbol~#2~already~assigned
3343 }
3344 \msg_new:nnn{stex}{error/overarity}{
3345   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3346 }
```

29.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3346
3347
3348 \bool_new:N \l_stex_allow_semantic_bool
3349 \bool_set_true:N \l_stex_allow_semantic_bool
3350
```

```

3351 \cs_new_protected:Nn \stex_invoke_symbol:n {
3352   \ifvmode\indent\fi
3353   \bool_if:NTF \l_stex_allow_semantic_bool {
3354     \str_if_eq:eeF {
3355       \prop_item:cn {
3356         l_stex_symdecl_#1_prop
3357       }{ deprecate }
3358     }{}{
3359       \msg_warning:nxxx{stex}{warning/deprecated}{
3360         Symbol~#1
3361       }{
3362         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3363       }
3364     }
3365     \if_mode_math:
3366       \exp_after:wN \__stex_terms_invoke_math:n
3367     \else:
3368       \exp_after:wN \__stex_terms_invoke_text:n
3369     \fi: { #1 }
3370   }{
3371     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3372   }
3373 }
3374
3375 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3376   \peek_charcode_remove:NTF ! {
3377     \__stex_terms_invoke_op_custom:nn {#1}
3378   }{
3379     \__stex_terms_invoke_custom:nn {#1}
3380   }
3381 }
3382
3383 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3384   \peek_charcode_remove:NTF ! {
3385     % operator
3386     \peek_charcode_remove:NTF * {
3387       % custom op
3388       \__stex_terms_invoke_op_custom:nn {#1}
3389     }{
3390       % op notation
3391       \peek_charcode:NTF [ {
3392         \__stex_terms_invoke_op_notation:nw {#1}
3393       }{
3394         \__stex_terms_invoke_op_notation:nw {#1}[]
3395       }
3396     }
3397   }{
3398     \peek_charcode_remove:NTF * {
3399       \__stex_terms_invoke_custom:nn {#1}
3400       % custom
3401     }{
3402       % normal
3403       \peek_charcode:NTF [ {
3404         \__stex_terms_invoke_notation:nw {#1}

```

```

3405     }{
3406         \__stex_terms_invoke_notation:nw {#1}[]
3407     }
3408 }
3409 }
3410 }
3411
3412
3413 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3414     \exp_args:Nnx \use:nn {
3415         \def\comp{\_comp}
3416         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3417         \bool_set_false:N \l_stex_allow_semantic_bool
3418         \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3419             \comp{ #2 }
3420         }
3421     }{
3422         \stex_reset:N \comp
3423         \stex_reset:N \STEXInternalCurrentSymbolStr
3424         \bool_set_true:N \l_stex_allow_semantic_bool
3425     }
3426 }
3427
3428 \keys_define:nn { stex / terms } {
3429     % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3430     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3431     unknown .code:n      = \str_set:Nx
3432         \l_stex_notation_variant_str \l_keys_key_str
3433 }
3434
3435 \cs_new_protected:Nn \__stex_terms_args:n {
3436     % \str_clear:N \l_stex_notation_lang_str
3437     \str_clear:N \l_stex_notation_variant_str
3438
3439     \keys_set:nn { stex / terms } { #1 }
3440 }
3441
3442 \cs_new_protected:Nn \stex_find_notation:nn {
3443     \__stex_terms_args:n { #2 }
3444     \seq_if_empty:cTF {
3445         l_stex_symdecl_ #1 _notations
3446     } {
3447         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3448     } {
3449         \str_if_empty:NTF \l_stex_notation_variant_str {
3450             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3451         }{
3452             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3453                 \l_stex_notation_variant_str
3454             }{
3455                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3456             }{
3457                 \msg_error:nnxx{stex}{error/nonotation}{#1}{
3458                     ~\l_stex_notation_variant_str

```



```

3459     }
3460   }
3461 }
3462 }
3463 }
3464
3465 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3466   \exp_args:Nnx \use:nn {
3467     \def\comp{\_comp}
3468     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3469     \stex_find_notation:nn { #1 }{ #2 }
3470     \bool_set_false:N \l_stex_allow_semantic_bool
3471     \cs_if_exist:cTF {
3472       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3473     }{
3474       \_stex_term_oms:nnn { #1 }{
3475         #1 \c_hash_str \l_stex_notation_variant_str
3476       }{
3477         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3478       }
3479     }{
3480       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3481         \cs_if_exist:cTF {
3482           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3483         }{
3484           \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3485             \_stex_reset:N \comp
3486             \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3487             \_stex_reset:N \STEXInternalCurrentSymbolStr
3488             \bool_set_true:N \l_stex_allow_semantic_bool
3489           }
3490           \def\comp{\_comp}
3491           \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3492           \bool_set_false:N \l_stex_allow_semantic_bool
3493           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3494         }{
3495           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3496             ~\l_stex_notation_variant_str
3497           }
3498         }
3499       }{
3500         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3501       }
3502     }
3503   }{
3504     \_stex_reset:N \comp
3505     \_stex_reset:N \STEXInternalCurrentSymbolStr
3506     \bool_set_true:N \l_stex_allow_semantic_bool
3507   }
3508 }
3509
3510 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3511   \stex_find_notation:nn { #1 }{ #2 }
3512   \cs_if_exist:cTF {

```

```

3513     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3514   }{
3515     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3516       \_stex_reset:N \comp
3517       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3518       \_stex_reset:N \STEXInternalCurrentSymbolStr
3519       \bool_set_true:N \l_stex_allow_semantic_bool
3520     }
3521     \def\comp{\_comp}
3522     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3523     \bool_set_false:N \l_stex_allow_semantic_bool
3524     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3525   }{
3526     \msg_error:nnxx{stex}{error/nonotation}{#1}{
3527       ~\l_stex_notation_variant_str
3528     }
3529   }
3530 }
3531
3532 \prop_new:N \l__stex_terms_custom_args_prop
3533
3534 \cs_new_protected:Nn \__stex_terms_custom_comp:n { \bool_set_false:N \l_stex_allow_semantic_bool }
3535
3536 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3537   \exp_args:Nnx \use:nn {
3538     \def\comp{\_stex_terms_custom_comp:n}
3539     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3540     \prop_clear:N \l__stex_terms_custom_args_prop
3541     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3542     \prop_get:cnN {
3543       l_stex_symdecl_#1 _prop
3544     } { args } \l_tmpa_str
3545     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3546     \tl_set:Nn \arg { \_stex_terms_arg: }
3547     \str_if_empty:NTF \l_tmpa_str {
3548       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3549     }{
3550       \str_if_in:NnTF \l_tmpa_str b {
3551         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3552       }{
3553         \str_if_in:NnTF \l_tmpa_str B {
3554           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3555         }{
3556           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3557         }
3558       }
3559     }
3560     % TODO check that all arguments exist
3561   }{
3562     \_stex_reset:N \STEXInternalCurrentSymbolStr
3563     \_stex_reset:N \arg
3564     \_stex_reset:N \comp
3565     \_stex_reset:N \l__stex_terms_custom_args_prop
3566     %\bool_set_true:N \l_stex_allow_semantic_bool

```

```

3567 }
3568 }
3569
3570 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3571   \tl_if_empty:nTF {#2}{
3572     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3573     \bool_set_true:N \l_tmpa_bool
3574     \bool_do_while:Nn \l_tmpa_bool {
3575       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3576         \int_incr:N \l_tmpa_int
3577       }{
3578         \bool_set_false:N \l_tmpa_bool
3579       }
3580     }
3581   }{
3582     \int_set:Nn \l_tmpa_int { #2 }
3583   }
3584   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3585   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3586     \msg_error:nnxxx{stex}{error/overarity}
3587     {\int_use:N \l_tmpa_int}
3588     {\STEXInternalCurrentSymbolStr}
3589     {\str_count:N \l_tmpa_str}
3590   }
3591   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3592   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3593     \bool_lazy_any:nF {
3594       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3595       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3596     }{
3597       \msg_error:nnxx{stex}{error/doubleargument}
3598       {\int_use:N \l_tmpa_int}
3599       {\STEXInternalCurrentSymbolStr}
3600     }
3601   }
3602   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3603   \bool_set_true:N \l_stex_allow_semantic_bool
3604   \IfBooleanTF#1{
3605     \stex_annotate_invisible:n { %TODO
3606       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3607     }
3608   }{ %TODO
3609     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3610   }
3611   \bool_set_false:N \l_stex_allow_semantic_bool
3612 }
3613
3614
3615 \cs_new_protected:Nn \_stex_term_arg:nn {
3616   \bool_set_true:N \l_stex_allow_semantic_bool
3617   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3618   \bool_set_false:N \l_stex_allow_semantic_bool
3619 }
3620

```

```

3621 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
3622   \exp_args:Nnx \use:nn
3623     { \int_set:Nn \l__stex_terms_downprec { #2 }
3624       \stex_term_arg:nn { #1 }{ #3 }
3625     }
3626   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3627 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 80.)

`\STEXInternalTermMathAssocArgiii`

```

3628 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiii #1#2#3#4 {
3629   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3630   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#1}{#2}}
3631   \tl_if_empty:nTF { #3 }{
3632     \STEXInternalTermMathArgiii{#1}{#2}{}
3633   }{
3634     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3635       \expandafter\if\expandafter\relax\noexpand#3
3636         \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3637       \else
3638         \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3639       \fi
3640       \l_tmpa_tl
3641     }{
3642       \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3643     }
3644   }
3645 }
3646
3647 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3648   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3649   \str_if_empty:NTF \l_tmpa_str {
3650     \exp_args:Nx \cs_if_eq:NNTF {
3651       \tl_head:N #1
3652     } \stex_invoke_sequence:n {
3653       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3654       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3655       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3656       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3657       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3658         \exp_not:n{\exp_args:Nnx \use:nn} {
3659           \exp_not:n {
3660             \def\comp{\_varcomp}
3661             \str_set:Nn \STEXInternalCurrentSymbolStr
3662               { \varseq:/{\l_tmpa_str}
3663             \exp_not:n{ ##1 }
3664           }{
3665             \exp_not:n {
3666               \stex_reset:N \comp
3667               \stex_reset:N \STEXInternalCurrentSymbolStr
3668             }
3669           }
3670         }}}

```

```

3671 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3672 \seq_reverse:N \l_tmpa_seq
3673 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3674 \seq_map_inline:Nn \l_tmpa_seq {
3675   \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3676     \exp_args:Nno
3677     \l_tmpa_cs { ##1 } \l_tmpa_tl
3678   }
3679 }
3680 \tl_set:Nx \l_tmpa_tl {
3681   \stex_term_omv:nn {varseq://\l_tmpa_str}{
3682     \exp_args:No \exp_not:n \l_tmpa_tl
3683   }
3684 }
3685 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3686 }{
3687   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3688 }
3689 } {
3690   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3691 }
3692 }
3693 }
3694 }
3695 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3696   \clist_set:Nn \l_tmpa_clist{ #2 }
3697   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3698     \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3699   }{
3700     \clist_reverse:N \l_tmpa_clist
3701     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3702     \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3703       \exp_args:No \exp_not:n \l_tmpa_tl
3704     }}
3705     \clist_map_inline:Nn \l_tmpa_clist {
3706       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3707         \exp_args:Nno
3708         \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3709       }
3710     }
3711   }
3712   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3713 }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page [81](#).)

29.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3714 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3715 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}

```

```

3716 \int_new:N \l__stex_terms_downprec
3717 \int_set_eq:NN \l__stex_terms_downprec \infpref

```

(End definition for `\infpref`, `\neginfpref`, and `\l__stex_terms_downprec`. These variables are documented on page 81.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str

```

```

3718 \tl_set:Nn \l__stex_terms_left_bracket_str (
3719 \tl_set:Nn \l__stex_terms_right_bracket_str )

```

(End definition for `\l__stex_terms_left_bracket_str` and `\l__stex_terms_right_bracket_str`.)

```

\__stex_terms_maybe_brackets:nn

```

Compares precedences and insert brackets accordingly

```

3720 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3721   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3722     \bool_set_false:N \l__stex_terms_brackets_done_bool
3723     #2
3724   } {
3725     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3726       \bool_if:NTF \l__stex_inarray_bool { #2 } {
3727         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3728         \dobrackets { #2 }
3729       }
3730     }{ #2 }
3731   }
3732 }

```

(End definition for `__stex_terms_maybe_brackets:nn`.)

\dobrackets

```

3733 \bool_new:N \l__stex_terms_brackets_done_bool
3734 %\RequirePackage{scalerel}
3735 \cs_new_protected:Npn \dobrackets #1 {
3736   %\ThisStyle{\if D\m@switch
3737   %   \exp_args:Nnx \use:nn
3738   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3739   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3740   % \else
3741   \exp_args:Nnx \use:nn
3742   {
3743     \bool_set_true:N \l__stex_terms_brackets_done_bool
3744     \int_set:Nn \l__stex_terms_downprec \infpref
3745     \l__stex_terms_left_bracket_str
3746     #1
3747   }
3748   {
3749     \bool_set_false:N \l__stex_terms_brackets_done_bool
3750     \l__stex_terms_right_bracket_str
3751     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3752   }
3753   %\fi}
3754 }

```

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

\withbrackets

```
3755 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3756   \exp_args:Nnx \use:nn
3757   {
3758     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3759     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3760     #3
3761   }
3762   {
3763     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3764     {\l__stex_terms_left_bracket_str}
3765     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3766     {\l__stex_terms_right_bracket_str}
3767   }
3768 }
```

(End definition for \withbrackets. This function is documented on page 81.)

\STEXinvisible

```
3769 \cs_new_protected:Npn \STEXinvisible #1 {
3770   \stex_annotate_invisible:n { #1 }
3771 }
```

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

OMDoc terms:

\STEXInternalTermMathOMSiiii

```
3772 \cs_new_protected:Nn \_stex_term_oms:nnn {
3773   \stex_annotate:nnn{ OMID }{ #2 }{
3774     #3
3775   }
3776 }
3777
3778 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
3779   \_stex_terms_maybe_brackets:nn { #3 }{
3780     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3781   }
3782 }
```

(End definition for \STEXInternalTermMathOMSiiii. This function is documented on page 80.)

_stex_term_math_omv:nn

```
3783 \cs_new_protected:Nn \_stex_term_omv:nn {
3784   \stex_annotate:nnn{ OMV }{ #1 }{
3785     #2
3786   }
3787 }
```

(End definition for _stex_term_math_omv:nn. This function is documented on page ??.)

\STEXInternalTermMathOMAiiai

```
3788 \cs_new_protected:Nn \_stex_term_oma:nnn {
3789   \stex_annotate:nnn{ OMA }{ #2 }{
3790     #3
3791   }
```

```

3792 }
3793
3794 \cs_new_protected:Npn \STEXInternalTermMathOMAiiai #1#2#3#4 {
3795   \__stex_terms_maybe_brackets:nn { #3 }{
3796     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3797   }
3798 }

```

(End definition for \STEXInternalTermMathOMAiiai. This function is documented on page 80.)

\STEXInternalTermMathOMBiai

```

3799 \cs_new_protected:Nn \stex_term_ombind:nnn {
3800   \stex_annotate:nnn{ OMBIND }{ #2 }{
3801     #3
3802   }
3803 }
3804
3805 \cs_new_protected:Npn \STEXInternalTermMathOMBiai #1#2#3#4 {
3806   \__stex_terms_maybe_brackets:nn { #3 }{
3807     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3808   }
3809 }

```

(End definition for \STEXInternalTermMathOMBiai. This function is documented on page 80.)

\symref
\symname

```

3810 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3811
3812 \keys_define:nn { stex / symname } {
3813   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3814   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3815   root     .tl_set_x:N      = \l__stex_terms_root_tl
3816 }
3817
3818 \cs_new_protected:Nn \stex_symname_args:n {
3819   \tl_clear:N \l__stex_terms_post_tl
3820   \tl_clear:N \l__stex_terms_pre_tl
3821   \tl_clear:N \l__stex_terms_root_str
3822   \keys_set:nn { stex / symname } { #1 }
3823 }
3824
3825 \NewDocumentCommand \symref { m m }{
3826   \let\compemph_uri_prev:\compemph@uri
3827   \let\compemph@uri\symrefemph@uri
3828   \STEXsymbol{#1}!\{ #2 }
3829   \let\compemph@uri\compemph_uri_prev:
3830 }
3831
3832 \NewDocumentCommand \synonym { O{} m m }{
3833   \stex_symname_args:n { #1 }
3834   \let\compemph_uri_prev:\compemph@uri
3835   \let\compemph@uri\symrefemph@uri
3836   % TODO
3837   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3838   \let\compemph@uri\compemph_uri_prev:

```



```

3839 }
3840
3841 \NewDocumentCommand \symname { 0{} m }{
3842   \stex_symname_args:n { #1 }
3843   \stex_get_symbol:n { #2 }
3844   \str_set:Nx \l_tmpa_str {
3845     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3846   }
3847   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3848
3849   \let\compemph_uri_prev:\compemph@uri
3850   \let\compemph@uri\symrefemph@uri
3851   \exp_args:NNx \use:nn
3852   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3853     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3854   } }
3855   \let\compemph@uri\compemph_uri_prev:
3856 }
3857
3858 \NewDocumentCommand \Symname { 0{} m }{
3859   \stex_symname_args:n { #1 }
3860   \stex_get_symbol:n { #2 }
3861   \str_set:Nx \l_tmpa_str {
3862     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3863   }
3864   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3865   \let\compemph_uri_prev:\compemph@uri
3866   \let\compemph@uri\symrefemph@uri
3867   \exp_args:NNx \use:nn
3868   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3869     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3870     \l__stex_terms_post_tl
3871   } }
3872   \let\compemph@uri\compemph_uri_prev:
3873 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 80.)

29.3 Notation Components

```

3874 <@@=stex_notationcomps>

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri

3875 \cs_new_protected:Npn \_comp #1 {
3876   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3877     \stex_html_backend:TF {
3878       \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3879     }{
3880       \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3881     }
3882   }
3883 }
3884
3885 \cs_new_protected:Npn \_varcomp #1 {

```

```

3886 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3887   \stex_html_backend:TF {
3888     \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3889   }{
3890     \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
3891   }
3892 }
3893 }
3894
3895 \def\comp{\_comp}
3896
3897 \cs_new_protected:Npn \compemph@uri #1 #2 {
3898   \compemph{ #1 }
3899 }
3900
3901
3902 \cs_new_protected:Npn \compemph #1 {
3903   #1
3904 }
3905
3906 \cs_new_protected:Npn \defemph@uri #1 #2 {
3907   \defemph{#1}
3908 }
3909
3910 \cs_new_protected:Npn \defemph #1 {
3911   \textbf{#1}
3912 }
3913
3914 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3915   \symrefemph{#1}
3916 }
3917
3918 \cs_new_protected:Npn \symrefemph #1 {
3919   \emph{#1}
3920 }
3921
3922 \cs_new_protected:Npn \varemp@uri #1 #2 {
3923   \varemp{#1}
3924 }
3925
3926 \cs_new_protected:Npn \varemp #1 {
3927   #1
3928 }

```

(End definition for `\comp` and others. These functions are documented on page 81.)

\ellipses

```

3929 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3930 \bool_new:N \l_stex_inarray_bool
\parrayline 3931 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3932 \NewDocumentCommand \parray { m m } {
\parraycell

```

```

3933 \begingroup
3934 \bool_set_true:N \l_stex_inarray_bool
3935 \begin{array}{#1}
3936 #2
3937 \end{array}
3938 \endgroup
3939 }
3940
3941 \NewDocumentCommand \prmatrix { m } {
3942 \begingroup
3943 \bool_set_true:N \l_stex_inarray_bool
3944 \begin{matrix}
3945 #1
3946 \end{matrix}
3947 \endgroup
3948 }
3949
3950 \def \maybepline {
3951 \bool_if:NT \l_stex_inarray_bool {\hline}
3952 }
3953
3954 \def \parrayline #1 #2 {
3955 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3956 }
3957
3958 \def \pmrow #1 { \parrayline{}{ #1 } }
3959
3960 \def \parraylineh #1 #2 {
3961 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3962 }
3963
3964 \def \parraycell #1 {
3965 #1 \bool_if:NT \l_stex_inarray_bool {&}
3966 }

```

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

29.4 Variables

```

3967 <@@=stex_variables>

```

`\stex_invoke_variable:n` Invokes a variable

```

3968 \cs_new_protected:Nn \stex_invoke_variable:n {
3969 \if_mode_math:
3970 \exp_after:wN \__stex_variables_invoke_math:n
3971 \else:
3972 \exp_after:wN \__stex_variables_invoke_text:n
3973 \fi: {#1}
3974 }
3975
3976 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3977 \peek_charcode_remove:NTF ! {
3978 \__stex_variables_invoke_op_custom:nn {#1}
3979 }{

```

```

3980     \__stex_variables_invoke_custom:nn {#1}
3981   }
3982 }
3983
3984
3985 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3986   \peek_charcode_remove:NTF ! {
3987     \peek_charcode_remove:NTF ! {
3988       \peek_charcode:NTF [ {
3989         % TODO throw error
3990       }{
3991         \__stex_variables_invoke_op_custom:nn
3992       }
3993     }{
3994       \__stex_variables_invoke_op:n { #1 }
3995     }
3996   }{
3997     \peek_charcode_remove:NTF * {
3998       \__stex_variables_invoke_custom:nn { #1 }
3999     }{
4000       \__stex_variables_invoke_math_ii:n { #1 }
4001     }
4002   }
4003 }
4004
4005 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4006   \exp_args:Nnx \use:nn {
4007     \def\comp{\_varcomp}
4008     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4009     \bool_set_false:N \l_stex_allow_semantic_bool
4010     \stex_term_omv:nn {var://#1}{
4011       \comp{ #2 }
4012     }
4013   }{
4014     \_stex_reset:N \comp
4015     \_stex_reset:N \STEXInternalCurrentSymbolStr
4016     \bool_set_true:N \l_stex_allow_semantic_bool
4017   }
4018 }
4019
4020 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4021   \cs_if_exist:cTF {
4022     stex_var_op_notation_ #1 _cs
4023   }{
4024     \exp_args:Nnx \use:nn {
4025       \def\comp{\_varcomp}
4026       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4027       \_stex_term_omv:nn { var://#1 }{
4028         \use:c{stex_var_op_notation_ #1 _cs }
4029       }
4030     }{
4031       \_stex_reset:N \comp
4032       \_stex_reset:N \STEXInternalCurrentSymbolStr
4033     }

```

```

4034 }{
4035   \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
4036     \__stex_variables_invoke_math_ii:n {#1}
4037   }{
4038     \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
4039   }
4040 }
4041 }
4042
4043 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4044   \cs_if_exist:cTF {
4045     stex_var_notation_#1_cs
4046   }{
4047     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4048       \_stex_reset:N \comp
4049       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4050       \_stex_reset:N \STEXInternalCurrentSymbolStr
4051       \bool_set_true:N \l_stex_allow_semantic_bool
4052     }
4053     \def\comp{\_varcomp}
4054     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4055     \bool_set_false:N \l_stex_allow_semantic_bool
4056     \use:c{stex_var_notation_#1_cs}
4057   }{
4058     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
4059   }
4060 }
4061
4062 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4063   \exp_args:Nnx \use:nn {
4064     \def\comp{\_varcomp}
4065     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4066     \prop_clear:N \l__stex_terms_custom_args_prop
4067     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4068     \prop_get:cnN {
4069       l_stex_variable_#1_prop
4070     }{ args } \l_tmpa_str
4071     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4072     \tl_set:Nn \arg { \__stex_terms_arg: }
4073     \str_if_empty:NTF \l_tmpa_str {
4074       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4075     }{
4076       \str_if_in:NnTF \l_tmpa_str b {
4077         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4078       }{
4079         \str_if_in:NnTF \l_tmpa_str B {
4080           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4081         }{
4082           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4083         }
4084       }
4085     }
4086     % TODO check that all arguments exist
4087   }{

```

```

4088 \stex_reset:N \STEXInternalCurrentSymbolStr
4089 \stex_reset:N \arg
4090 \stex_reset:N \comp
4091 \stex_reset:N \l__stex_terms_custom_args_prop
4092 %\bool_set_true:N \l_stex_allow_semantic_bool
4093 }
4094 }

```

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

29.5 Sequences

```

4095 <@@=stex_sequences>
4096
4097 \cs_new_protected:Nn \stex_invoke_sequence:n {
4098   \peek_charcode_remove:NTF ! {
4099     \stex_term_omv:nn {varseq://#1}{
4100       \exp_args:Nnx \use:nn {
4101         \def\comp{\_varcomp}
4102         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4103         \prop_item:cn{stex_varseq_#1_prop}{notation}
4104       }{
4105         \stex_reset:N \comp
4106         \stex_reset:N \STEXInternalCurrentSymbolStr
4107       }
4108     }
4109   }{
4110     \bool_set_false:N \l_stex_allow_semantic_bool
4111     \def\comp{\_varcomp}
4112     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4113     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4114       \stex_reset:N \comp
4115       \stex_reset:N \STEXInternalSymbolAfterInvokationTL
4116       \stex_reset:N \STEXInternalCurrentSymbolStr
4117       \bool_set_true:N \l_stex_allow_semantic_bool
4118     }
4119     \use:c { stex_varseq_#1_cs }
4120   }
4121 }
4122 </package>

```

Chapter 30

STEX -Structural Features Implementation

```
4123 ⟨*package⟩
4124
4125 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4126
      Warnings and error messages
4127 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4128   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4129 }
4130 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4131   Symbol~#1~not~assigned~in~interpretmodule~#2
4132 }
4133
4134 \msg_new:nnn{stex}{error/unknownstructure}{
4135   No~structure~#1~found!
4136 }
4137
4138 \msg_new:nnn{stex}{error/unknownfield}{
4139   No~field~#1~in~instance~#2~found!\#3
4140 }
4141
4142 \msg_new:nnn{stex}{error/keyval}{
4143   Invalid~key=value~pair:#1
4144 }
4145 \msg_new:nnn{stex}{error/instantiate/missing}{
4146   Assignments~missing~in~instantiate:~#1
4147 }
4148 \msg_new:nnn{stex}{error/incompatible}{
4149   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4150 }
4151
```

30.1 Imports with modification

```

4152 <@@=stex_copymodule>
4153 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4154   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4155     \tl_set:Nn \l_tmpa_tl { #1 }
4156     \__stex_copymodule_get_symbol_from_cs:
4157   }{
4158     % argument is a string
4159     % is it a command name?
4160     \cs_if_exist:cTF { #1 }{
4161       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4162       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4163       \str_if_empty:NTF \l_tmpa_str {
4164         \exp_args:Nx \cs_if_eq:NNTF {
4165           \tl_head:N \l_tmpa_tl
4166         } \stex_invoke_symbol:n {
4167           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4168         }{
4169           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4170         }
4171       } {
4172         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4173       }
4174     }{
4175       % argument is not a command name
4176       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4177       % \l_stex_all_symbols_seq
4178     }
4179   }
4180 }
4181
4182 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4183   \str_set:Nn \l_tmpa_str { #1 }
4184   \bool_set_false:N \l_tmpa_bool
4185   \bool_if:NF \l_tmpa_bool {
4186     \tl_set:Nn \l_tmpa_tl {
4187       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4188     }
4189     \str_set:Nn \l_tmpa_str { #1 }
4190     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4191     \seq_map_inline:Nn #2 {
4192       \str_set:Nn \l_tmpb_str { ##1 }
4193       \str_if_eq:eeT { \l_tmpa_str } {
4194         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4195       } {
4196         \seq_map_break:n {
4197           \tl_set:Nn \l_tmpa_tl {
4198             \str_set:Nn \l_stex_get_symbol_uri_str {
4199               ##1
4200             }
4201           }
4202         }
4203       }

```



```

4204     }
4205     \l_tmpa_tl
4206   }
4207 }
4208
4209 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4210   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4211     { \tl_tail:N \l_tmpa_tl }
4212   \tl_if_single:NTF \l_tmpa_tl {
4213     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4214       \exp_after:wN \str_set:Nn \exp_after:wN
4215         \l_stex_get_symbol_uri_str \l_tmpa_tl
4216       \__stex_copymodule_get_symbol_check:n { #1 }
4217     }{
4218       % TODO
4219       % tail is not a single group
4220     }
4221   }{
4222     % TODO
4223     % tail is not a single group
4224   }
4225 }
4226
4227 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4228   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4229     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4230       :~\seq_use:Nn #1 {,~}
4231     }
4232   }
4233 }
4234
4235 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4236   % import module
4237   \stex_import_module_uri:nn { #1 } { #2 }
4238   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4239   \stex_import_require_module:nnnn
4240     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4241     { \l_stex_import_path_str } { \l_stex_import_name_str }
4242
4243   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4244   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4245
4246   % fields
4247   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4248   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4249     \seq_map_inline:cn {c_stex_module_##1_constants}{
4250       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4251         ##1 ? ####1
4252       }
4253     }
4254   }
4255
4256   % setup prop
4257   \seq_clear:N \l_tmpa_seq

```

```

4258 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4259   name      = \l_stex_current_copymodule_name_str ,
4260   module    = \l_stex_current_module_str ,
4261   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4262   includes  = \l_tmpa_seq %,
4263 % fields    = \l_tmpa_seq
4264 }
4265 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4266   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4267 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4268 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4269
4270 \stex_if_do_html:T {
4271   \begin{stex_annotate_env} {#4} {
4272     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4273   }
4274   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
4275 }
4276 }
4277
4278 \cs_new_protected:Nn \stex_copymodule_end:n {
4279 % apply to every field
4280 \def \l_tmpa_cs ##1 ##2 {#1}
4281
4282 \tl_clear:N \__stex_copymodule_module_tl
4283 \tl_clear:N \__stex_copymodule_exec_tl
4284
4285 %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4286 \seq_clear:N \__stex_copymodule_fields_seq
4287
4288 \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4289   \seq_map_inline:cn {c_stex_module_##1_constants}{
4290
4291     \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4292     \l_tmpa_cs{##1}{####1}
4293
4294     \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4295       \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4296       \stex_if_do_html:T {
4297         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4298           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4299         }
4300       }
4301     }{
4302       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4303     }
4304
4305     \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4306     \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4307     \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4308
4309     \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4310       \stex_if_do_html:T {
4311         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4312     $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4313     }
4314   }
4315   \prop_put:Nnn \l_tmpa_prop { defined } { true }
4316 }
4317
4318 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4319 \tl_put_right:Nx \__stex_copymodule_module_tl {
4320   \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4321   \prop_set_from_keyval:cn {
4322     l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4323   }{
4324     \prop_to_keyval:N \l_tmpa_prop
4325   }
4326 }
4327
4328 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4329   \stex_if_do_html:T {
4330     \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4331       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4332     }
4333   }
4334   \tl_put_right:Nx \__stex_copymodule_module_tl {
4335     \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4336       \stex_invoke_symbol:n {
4337         \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4338       }
4339     }
4340   }
4341 }
4342
4343 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4344
4345 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4346   \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4347 }
4348
4349 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4350   \stex_if_do_html:TF{
4351     \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4352   }{
4353     \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4354   }
4355 }
4356 }
4357 }
4358
4359
4360 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4361 \tl_put_left:Nx \__stex_copymodule_module_tl {
4362   \prop_set_from_keyval:cn {
4363     l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4364   }{
4365     \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4366     }
4367 }
4368
4369 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4370   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4371 }
4372
4373 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4374 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4375 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4376
4377 \__stex_copymodule_exec_tl
4378 \stex_if_do_html:T {
4379   \end{stex_annotate_env}
4380 }
4381 }
4382
4383 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4384   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4385   \stex_deactivate_macro:Nn \symdecl {module~environments}
4386   \stex_deactivate_macro:Nn \symdef {module~environments}
4387   \stex_deactivate_macro:Nn \notation {module~environments}
4388   \stex_reactivate_macro:N \assign
4389   \stex_reactivate_macro:N \renamedekl
4390   \stex_reactivate_macro:N \donotcopy
4391   \stex_smsmode_do:
4392 }{
4393   \stex_copymodule_end:n {}
4394 }
4395
4396 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4397   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4398   \stex_deactivate_macro:Nn \symdecl {module~environments}
4399   \stex_deactivate_macro:Nn \symdef {module~environments}
4400   \stex_deactivate_macro:Nn \notation {module~environments}
4401   \stex_reactivate_macro:N \assign
4402   \stex_reactivate_macro:N \renamedekl
4403   \stex_reactivate_macro:N \donotcopy
4404   \stex_smsmode_do:
4405 }{
4406   \stex_copymodule_end:n {
4407     \tl_if_exist:cF {
4408       l__stex_copymodule_copymodule_##1?##2_def_tl
4409     }{
4410       \str_if_eq:eeF {
4411         \prop_item:cn{
4412           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4413       }{ true }{
4414         \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4415           ##1?##2
4416         }{\l_stex_current_copymodule_name_str}
4417       }
4418     }
4419   }

```

```

4420 }
4421
4422 \iffalse \begin{stex_annotate_env} \fi
4423 \NewDocumentEnvironment {realization} { 0 } { m } {
4424   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4425   \stex_deactivate_macro:Nn \symdecl {module~environments}
4426   \stex_deactivate_macro:Nn \symdef {module~environments}
4427   \stex_deactivate_macro:Nn \notation {module~environments}
4428   \stex_reactivate_macro:N \donotcopy
4429   \stex_reactivate_macro:N \assign
4430   \stex_smsmode_do:
4431 } {
4432   \stex_import_module_uri:nn { #1 } { #2 }
4433   \tl_clear:N \__stex_copymodule_exec_tl
4434   \tl_set:Nx \__stex_copymodule_module_tl {
4435     \stex_import_require_module:nnnn
4436     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4437     { \l_stex_import_path_str } { \l_stex_import_name_str }
4438   }
4439
4440   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4441     \seq_map_inline:cn {c_stex_module_##1_constants}{
4442       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4443       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4444         \stex_if_do_html:T {
4445           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4446             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4447               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4448             }
4449           }
4450         }
4451         \tl_put_right:Nx \__stex_copymodule_module_tl {
4452           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4453         }
4454       }
4455     }
4456
4457     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4458
4459     \__stex_copymodule_exec_tl
4460     \stex_if_do_html:T {\end{stex_annotate_env}}
4461   }
4462
4463   \NewDocumentCommand \donotcopy { m } {
4464     \str_clear:N \l_stex_import_name_str
4465     \str_set:Nn \l_tmpa_str { #1 }
4466     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4467     \seq_map_inline:Nn \l_stex_all_modules_seq {
4468       \str_set:Nn \l_tmpb_str { ##1 }
4469       \str_if_eq:eeT { \l_tmpa_str } {
4470         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4471       } {
4472         \seq_map_break:n {
4473           \stex_if_do_html:T {

```

```

4474         \stex_if_smsmode:F {
4475             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4476                 \stex_annotate:nnn{domain}{##1}{}}
4477         }
4478     }
4479 }
4480 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4481 }
4482 }
4483 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4484     \str_set:Nn \l_tmpb_str { #####1 }
4485     \str_if_eq:eeT { \l_tmpa_str } {
4486         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4487     } {
4488         \seq_map_break:n {\seq_map_break:n {
4489             \stex_if_do_html:T {
4490                 \stex_if_smsmode:F {
4491                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4492                         \stex_annotate:nnn{domain}{
4493                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4494                         }}
4495                 }
4496             }
4497         }
4498         \str_set:Nx \l_stex_import_name_str {
4499             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4500         }
4501     }}
4502 }
4503 }
4504 }
4505 \str_if_empty:NTF \l_stex_import_name_str {
4506     % TODO throw error
4507 }{
4508     \stex_collect_imports:n {\l_stex_import_name_str }
4509     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4510         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4511         \seq_map_inline:cn {c_stex_module_###1_constants}{
4512             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4513             \bool_lazy_any:nT {
4514                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4515                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4516                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4517             }{
4518                 % TODO throw error
4519             }
4520         }
4521     }
4522     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4523     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4524     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4525 }
4526 \stex_smsmode_do:
4527 }

```

```

4528
4529 \NewDocumentCommand \assign { m m }{
4530   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4531   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4532   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4533   \stex_smsmode_do:
4534 }
4535
4536 \keys_define:nn { stex / renamedecl } {
4537   name          .str_set_x:N = \l_stex_renamedecl_name_str
4538 }
4539 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4540   \str_clear:N \l_stex_renamedecl_name_str
4541   \keys_set:nn { stex / renamedecl } { #1 }
4542 }
4543
4544 \NewDocumentCommand \renamedecl { O{} m m }{
4545   \__stex_copymodule_renamedecl_args:n { #1 }
4546   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4547   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4548   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4549   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4550     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4551       \l_stex_get_symbol_uri_str
4552     } }
4553   } {
4554     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4555       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4556       \prop_set_eq:cc {l_stex_symdecl_
4557         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4558         _prop
4559       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4560       \seq_set_eq:cc {l_stex_symdecl_
4561         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4562         _notations
4563       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4564       \prop_put:cnx {l_stex_symdecl_
4565         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4566         _prop
4567       }{ name }{ \l_stex_renamedecl_name_str }
4568       \prop_put:cnx {l_stex_symdecl_
4569         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4570         _prop
4571       }{ module }{ \l_stex_current_module_str }
4572       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4573         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4574       }
4575       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4576         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4577       } }
4578     }
4579   \stex_smsmode_do:
4580 }
4581

```

```

4582 \stex_deactivate_macro:Nn \assign {copymodules}
4583 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4584 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4585
4586

```

30.2 The feature environment

`structural@feature (env.)`

```

4587 <@@=stex_features>
4588
4589 \NewDocumentEnvironment{structural_feature_module}{m m m}{
4590   \stex_if_in_module:F {
4591     \msg_set:nnn{stex}{error/nomodule}{
4592       Structural~Feature~has~to~occur~in~a~module:\\
4593       Feature~#2~of~type~#1\\
4594       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4595     }
4596     \msg_error:nn{stex}{error/nomodule}
4597   }
4598
4599   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4600
4601   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4602
4603   \stex_if_do_html:T {
4604     \begin{stex_annotate_env}{feature:#1}{\l_stex_feature_parent_str ? #2 - #1}
4605     \stex_annotate_invisible:nnn{header}{}{#3 }
4606   }
4607 }{
4608   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4609   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4610   \stex_debug:nn{features}{
4611     Feature: \l_stex_last_feature_str
4612   }
4613   \stex_if_do_html:T {
4614     \end{stex_annotate_env}
4615   }
4616 }

```

30.3 Structure

`structure (env.)`

```

4617 <@@=stex_structures>
4618 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4619   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4620     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4621   }
4622   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4623   {#1}{#2}
4624 }
4625

```



```

4626 \keys_define:nn { stex / features / structure } {
4627   name          .str_set_x:N = \l__stex_structures_name_str ,
4628 }
4629
4630 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4631   \str_clear:N \l__stex_structures_name_str
4632   \keys_set:nn { stex / features / structure } { #1 }
4633 }
4634
4635 \NewDocumentEnvironment{mathstructure}{m O{}}{
4636   \__stex_structures_structure_args:n { #2 }
4637   \str_if_empty:NT \l__stex_structures_name_str {
4638     \str_set:Nx \l__stex_structures_name_str { #1 }
4639   }
4640   \stex_suppress_html:n {
4641     \bool_set_true:N \l_stex_symdecl_make_macro_bool
4642     \exp_args:Nx \stex_symdecl_do:nn {
4643       name = \l__stex_structures_name_str ,
4644       def = {\STEXsymbol{module-type}}{
4645         \STEXInternalTermMathOMSiiii {
4646           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4647             { ns } ?
4648           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4649             { name } / \l__stex_structures_name_str - structure
4650         }{}{0}{}
4651       }}
4652     }{ #1 }
4653   }
4654   \exp_args:Nnnx
4655   \begin{structural_feature_module}{ structure }
4656     { \l__stex_structures_name_str }{}
4657   \stex_smsmode_do:
4658 }{
4659   \end{structural_feature_module}
4660   \_stex_reset_up_to_module:n \l_stex_last_feature_str
4661   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4662   \seq_clear:N \l_tmpa_seq
4663   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4664     \seq_map_inline:cn{c_stex_module_##1_constants}{
4665       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4666     }
4667   }
4668   \exp_args:Nnno
4669   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4670   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4671   \stex_add_structure_to_current_module:nn
4672     \l__stex_structures_name_str
4673     \l_stex_last_feature_str
4674
4675   \stex_execute_in_module:x {
4676     \tl_set:cn { #1 }{
4677       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4678     }
4679   }

```

```

4680 }
4681
4682 \cs_new:Nn \stex_invoke_structure:nn {
4683   \stex_invoke_symbol:n { #1?#2 }
4684 }
4685
4686 \cs_new_protected:Nn \stex_get_structure:n {
4687   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4688     \tl_set:Nn \l_tmpa_tl { #1 }
4689     \__stex_structures_get_from_cs:
4690   }{
4691     \cs_if_exist:cTF { #1 }{
4692       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4693       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4694       \str_if_empty:NTF \l_tmpa_str {
4695         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4696           \__stex_structures_get_from_cs:
4697         }{
4698           \__stex_structures_get_from_string:n { #1 }
4699         }
4700       }{
4701         \__stex_structures_get_from_string:n { #1 }
4702       }
4703     }{
4704       \__stex_structures_get_from_string:n { #1 }
4705     }
4706   }
4707 }
4708
4709 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4710   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4711     { \tl_tail:N \l_tmpa_tl }
4712   \str_set:Nx \l_tmpa_str {
4713     \exp_after:wN \use_i:nn \l_tmpa_tl
4714   }
4715   \str_set:Nx \l_tmpb_str {
4716     \exp_after:wN \use_ii:nn \l_tmpa_tl
4717   }
4718   \str_set:Nx \l_stex_get_structure_str {
4719     \l_tmpa_str ? \l_tmpb_str
4720   }
4721   \str_set:Nx \l_stex_get_structure_module_str {
4722     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4723   }
4724 }
4725
4726 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4727   \tl_set:Nn \l_tmpa_tl {
4728     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4729   }
4730   \str_set:Nn \l_tmpa_str { #1 }
4731   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4732
4733   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

4734 \prop_if_exist:cT {c_stex_module_##1_structures} {
4735 \prop_map_inline:cn {c_stex_module_##1_structures} {
4736 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4737 \prop_map_break:n{\seq_map_break:n{
4738 \tl_set:Nn \l_tmpa_tl {
4739 \str_set:Nn \l_stex_get_structure_str {##1?####1}
4740 \str_set:Nn \l_stex_get_structure_module_str {####2}
4741 }
4742 }}
4743 }
4744 }
4745 }
4746 }
4747 \l_tmpa_tl
4748 }

```

\instantiate

```

4749
4750 \keys_define:nn { stex / instantiate } {
4751 name .str_set_x:N = \l__stex_structures_name_str
4752 }
4753 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4754 \str_clear:N \l__stex_structures_name_str
4755 \keys_set:nn { stex / instantiate } { #1 }
4756 }
4757
4758 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4759 \beginingroup
4760 \stex_get_structure:n {#3}
4761 \__stex_structures_instantiate_args:n { #2 }
4762 \str_if_empty:NT \l__stex_structures_name_str {
4763 \str_set:Nn \l__stex_structures_name_str { #1 }
4764 }
4765 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4766 \seq_clear:N \l__stex_structures_fields_seq
4767 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4768 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4769 \seq_map_inline:cn {c_stex_module_##1_constants}{
4770 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4771 }
4772 }
4773
4774 \tl_if_empty:nF{#5}{
4775 \seq_set_split:Nnn \l_tmpa_seq , {#5}
4776 \prop_clear:N \l_tmpa_prop
4777 \seq_map_inline:Nn \l_tmpa_seq {
4778 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4779 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4780 \msg_error:nnn{stex}{error/keyval}{##1}
4781 }
4782 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4783 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4784 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4785 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```

```

4786     \exp_args:Nxx \str_if_eq:nnF
4787     {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
4788     {\prop_item:cn{l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{args}}{
4789     \msg_error:nnxxxx{stex}{error/incompatible}
4790     {l__stex_structures_dom_str}
4791     {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
4792     {\l_stex_get_symbol_uri_str}
4793     {\prop_item:cn{l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{args}}
4794     }
4795     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4796   }
4797 }
4798
4799 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4800   \str_set:Nx \l_tmpa_str {field:l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4801   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4802
4803   \stex_add_constant_to_current_module:n {\l_tmpa_str}
4804   \stex_execute_in_module:x {
4805     \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4806     name   = \l_tmpa_str ,
4807     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4808     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4809     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4810   }
4811   \seq_clear:c {l_stex_symdecl\_l_stex_current_module_str?\l_tmpa_str _notations}
4812 }
4813
4814 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4815   \stex_find_notation:nn{##1}{}
4816   \stex_execute_in_module:x {
4817     \seq_put_right:cn {l_stex_symdecl\_l_stex_current_module_str?\l_tmpa_str _notation
4818   }
4819
4820   \stex_copy_control_sequence_ii:ccN
4821   {stex_notation\_l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4822   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4823   \l_tmpa_tl
4824   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4825
4826
4827   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4828     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4829     \stex_execute_in_module:x {
4830       \tl_set:cn
4831       {stex_op_notation\_l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4832       { \exp_args:No \exp_not:n \l_tmpa_cs}
4833     }
4834   }
4835
4836 }
4837
4838 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4839 }

```

```

4840
4841 \stex_execute_in_module:x {
4842   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4843     domain = \l_stex_get_structure_module_str ,
4844     \prop_to_keyval:N \l_tmpa_prop
4845   }
4846   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4847 }
4848 \stex_debug:nn{instantiate}{
4849   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4850   \prop_to_keyval:N \l_tmpa_prop
4851 }
4852 \exp_args:Nxx \stex_symdecl_do:nn {
4853   type={\STEXsymbol{module-type}}{
4854     \STEXInternalTermMathOMSiiii {
4855       \l_stex_get_structure_module_str
4856     }{}{0}{}
4857   }}
4858 }{\l__stex_structures_name_str}
4859 % {
4860   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4861   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4862   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4863 % }
4864 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4865 \endgroup
4866 \stex_smsmode_do:\ignorespacesandpars
4867 }
4868
4869 \cs_new_protected:Nn \stex_symbol_or_var:n {
4870   \cs_if_exist:cTF{#1}{
4871     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4872     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4873     \str_if_empty:NTF \l_tmpa_str {
4874       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4875       \stex_invoke_variable:n {
4876         \bool_set_true:N \l_stex_symbol_or_var_bool
4877         \bool_set_false:N \l_stex_instance_or_symbol_bool
4878         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4879         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4880         \str_set:Nx \l_stex_get_symbol_uri_str {
4881           \exp_after:wN \use:n \l_tmpa_tl
4882         }
4883       }{ % TODO \stex_invoke_varinstance:n
4884         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
4885           \bool_set_true:N \l_stex_symbol_or_var_bool
4886           \bool_set_true:N \l_stex_instance_or_symbol_bool
4887           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4888           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4889           \str_set:Nx \l_stex_get_symbol_uri_str {
4890             \exp_after:wN \use:n \l_tmpa_tl
4891           }
4892         }{
4893           \bool_set_false:N \l_stex_symbol_or_var_bool

```

```

4894         \stex_get_symbol:n{#1}
4895     }
4896 }
4897 }{
4898     \__stex_structures_symbolorvar_from_string:n{ #1 }
4899 }
4900 }{
4901     \__stex_structures_symbolorvar_from_string:n{ #1 }
4902 }
4903 }
4904
4905 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4906     \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4907         \bool_set_true:N \l_stex_symbol_or_var_bool
4908         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4909     }{
4910         \bool_set_false:N \l_stex_symbol_or_var_bool
4911         \stex_get_symbol:n{#1}
4912     }
4913 }
4914
4915 \keys_define:nn { stex / varinstantiate } {
4916     name          .str_set_x:N = \l__stex_structures_name_str,
4917     bind          .choices:nn =
4918         {forall,exists}
4919         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4920 }
4921 }
4922 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4923     \str_clear:N \l__stex_structures_name_str
4924     \str_clear:N \l__stex_structures_bind_str
4925     \keys_set:nn { stex / varinstantiate } { #1 }
4926 }
4927
4928 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4929     \begingroup
4930         \stex_get_structure:n {#3}
4931         \__stex_structures_varinstantiate_args:n { #2 }
4932         \str_if_empty:NT \l__stex_structures_name_str {
4933             \str_set:Nn \l__stex_structures_name_str { #1 }
4934         }
4935         \stex_if_do_html:TF{
4936             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4937         }{\use:n}
4938         {
4939             \stex_if_do_html:T{
4940                 \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
4941             }
4942             \seq_clear:N \l__stex_structures_fields_seq
4943             \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4944             \seq_map_inline:Nn \l_stex_collect_imports_seq {
4945                 \seq_map_inline:cn {c_stex_module_##1_constants}{
4946                     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4947                 }

```

```

4948 }
4949 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4950 \prop_clear:N \l_tmpa_prop
4951 \tl_if_empty:nF {#5} {
4952   \seq_set_split:Nnn \l_tmpa_seq , {#5}
4953   \seq_map_inline:Nn \l_tmpa_seq {
4954     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4955     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4956       \msg_error:nnn{stex}{error/keyval}{##1}
4957     }
4958     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
4959     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4960     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq
4961     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4962     \stex_if_do_html:T{
4963       \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
4964         \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{}\l_stex_get_symbol_uri_str}{}}
4965     }
4966     \bool_if:NTF \l_stex_symbol_or_var_bool {
4967       \exp_args:Nxx \str_if_eq:nnF
4968         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4969         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{
4970         \msg_error:nnxxx{stex}{error/incompatible}
4971         {\l__stex_structures_dom_str}
4972         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4973         {\l_stex_get_symbol_uri_str}
4974         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}}
4975     }
4976     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4977   }}{
4978     \exp_args:Nxx \str_if_eq:nnF
4979       {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4980       {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4981       \msg_error:nnxxx{stex}{error/incompatible}
4982       {\l__stex_structures_dom_str}
4983       {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4984       {\l_stex_get_symbol_uri_str}
4985       {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}}
4986     }
4987     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4988   }}
4989   }
4990 }
4991 \tl_gclear:N \g__stex_structures_aftergroup_tl
4992 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4993   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l__stex_structures_fields_seq ##1} \l_tmpa_str}
4994   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4995   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4996     \stex_find_notation:nn{##1}{}
4997     \cs_gset_eq:cc{g__stex_structures_tmpa\l_tmpa_str _cs}
4998       {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4999     \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa\l_tmpa_str _cs}}
5000     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5001       \cs_gset_eq:cc {g__stex_structures_tmpa_op\l_tmpa_str _cs}

```

```

5002         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5003         \stex_debug:nn{varinstantiate}{Operator-Notation:~\cs_meaning:c{g__stex_struct
5004     }
5005 }
5006
5007 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5008     \prop_set_from_keyval:cn { \l_stex_variable_ \l_tmpa_str _prop}{
5009         name      = \l_tmpa_str ,
5010         args      = \prop_item:cn { \l_stex_symdecl_##1_prop}{args} ,
5011         arity     = \prop_item:cn { \l_stex_symdecl_##1_prop}{arity} ,
5012         assocs    = \prop_item:cn { \l_stex_symdecl_##1_prop}{assocs}
5013     }
5014     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
5015     {g__stex_structures_tmpa_\l_tmpa_str _cs}
5016     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
5017     {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5018 }
5019 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn { \l_stex_symdecl_##1_prop}{name}}{\stex_inv
5020 }
5021 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5022     \prop_set_from_keyval:cn { \l_stex_varinstance_\l__stex_structures_name_str _prop }{
5023         domain = \l_stex_get_structure_module_str ,
5024         \prop_to_keyval:N \l_tmpa_prop
5025     }
5026     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5027     \tl_set:cn { \l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
5028         \exp_args:Nnx \exp_not:N \use:nn {
5029             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5030             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5031                 \exp_not:n{
5032                     \_varcomp{#4}
5033                 }
5034             }
5035         }{
5036             \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5037         }
5038     }
5039 }
5040 }
5041 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5042 \aftergroup\g__stex_structures_aftergroup_tl
5043 \endgroup
5044 \stex_smsmode_do:\ignorespacesandpars
5045 }
5046
5047 \cs_new_protected:Nn \stex_invoke_instance:n {
5048     \peek_charcode_remove:NTF ! {
5049         \stex_invoke_symbol:n{#1}
5050     }{
5051         \_stex_invoke_instance:nn {#1}
5052     }
5053 }
5054
5055

```



```

5056 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5057   \peek_charcode_remove:NTF ! {
5058     \exp_args:Nnx \use:nn {
5059       \def\comp{\_varcomp}
5060       \use:c{l_stex_varinstance_#1_op_tl}
5061     }{
5062       \_stex_reset:N \comp
5063     }
5064   }{
5065     \_stex_invoke_varinstance:nn {#1}
5066   }
5067 }
5068
5069 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5070   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5071     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5072   }{
5073     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5074     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5075       \prop_to_keyval:N \l_tmpa_prop
5076     }
5077   }
5078 }
5079
5080 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5081   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5082     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5083     \l_tmpa_tl
5084   }{
5085     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5086   }
5087 }

```

(End definition for \instantiate. This function is documented on page 33.)

\stex_invoke_structure:nnn

```

5088 % #1: URI of the instance
5089 % #2: URI of the instantiated module
5090 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5091   \tl_if_empty:nTF{ #3 }{
5092     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5093       c_stex_feature_ #2 _prop
5094     }
5095     \tl_clear:N \l_tmpa_tl
5096     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5097     \seq_map_inline:Nn \l_tmpa_seq {
5098       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5099       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5100       \cs_if_exist:cT {
5101         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5102       }{
5103         \tl_if_empty:NF \l_tmpa_tl {
5104           \tl_put_right:Nn \l_tmpa_tl {,}
5105         }

```

```

5106         \tl_put_right:Nx \l_tmpa_tl {
5107             \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5108         }
5109     }
5110 }
5111 \exp_args:No \mathstruct \l_tmpa_tl
5112 }{
5113     \stex_invoke_symbol:n{#1/#3}
5114 }
5115 }

```

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

```

5116 </package>

```

Chapter 31

STEX -Statements Implementation

```
5117 <*package>
5118
5119 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5120
5121 <@@=stex_statements>
    Warnings and error messages
5122
\titleemph
5123 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

31.1 Definitions

definiendum

```
5124 \keys_define:nn {stex / definiendum }{
5125   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5126   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5127   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5128   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5129 }
5130 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5131   \str_clear:N \l__stex_statements_definiendum_root_str
5132   \tl_clear:N \l__stex_statements_definiendum_post_tl
5133   \str_clear:N \l__stex_statements_definiendum_gfa_str
5134   \keys_set:nn { stex / definiendum }{ #1 }
5135 }
5136 \NewDocumentCommand \definiendum { O{} m m } {
5137   \__stex_statements_definiendum_args:n { #1 }
5138   \stex_get_symbol:n { #2 }
5139   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5140   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5141     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5142     \tl_set:Nn \l_tmpa_tl { #3 }
5143   } {
5144     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5145     \tl_set:Nn \l_tmpa_tl {
5146       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5147     }
5148   }
5149 } {
5150   \tl_set:Nn \l_tmpa_tl { #3 }
5151 }
5152
5153 % TODO root
5154 \stex_html_backend:TF {
5155   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5156 } {
5157   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5158 }
5159 }
5160 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 42.)

definame

```

5161
5162 \NewDocumentCommand \definame { 0{ } m } {
5163   \__stex_statements_definiendum_args:n { #1 }
5164   % TODO: root
5165   \stex_get_symbol:n { #2 }
5166   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5167   \str_set:Nx \l_tmpa_str {
5168     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5169   }
5170   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5171   \stex_html_backend:TF {
5172     \stex_if_do_html:T {
5173       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5174         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5175       }
5176     }
5177   } {
5178     \exp_args:Nnx \defemph@uri {
5179       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5180     } { \l_stex_get_symbol_uri_str }
5181   }
5182 }
5183 \stex_deactivate_macro:Nn \definame {definition~environments}
5184
5185 \NewDocumentCommand \Definame { 0{ } m } {
5186   \__stex_statements_definiendum_args:n { #1 }
5187   \stex_get_symbol:n { #2 }
5188   \str_set:Nx \l_tmpa_str {
5189     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5190   }
5191   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5192 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5193 \stex_html_backend:TF {
5194   \stex_if_do_html:T {
5195     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5196       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5197     }
5198   }
5199 } {
5200   \exp_args:Nnx \defemph@uri {
5201     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5202   } { \l_stex_get_symbol_uri_str }
5203 }
5204 }
5205 \stex_deactivate_macro:Nn \Definame {definition-environments}
5206
5207 \NewDocumentCommand \premise { m }{
5208   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5209 }
5210 \NewDocumentCommand \conclusion { m }{
5211   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5212 }
5213 \NewDocumentCommand \definiens { 0{} m }{
5214   \str_clear:N \l_stex_get_symbol_uri_str
5215   \tl_if_empty:nF {#1} {
5216     \stex_get_symbol:n { #1 }
5217   }
5218   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5219     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5220       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5221     }{
5222       % TODO throw error
5223     }
5224   }
5225   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5226   {\l_stex_current_module_str}{
5227     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5228   }{true}{
5229     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5230     \exp_args:Nx \stex_add_to_current_module:n {
5231       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5232     }
5233   }
5234 }
5235 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5236 }
5237
5238 \NewDocumentCommand \varbindforall {m}{
5239   \stex_symbol_or_var:n {#1}
5240   \bool_if:NTF\l_stex_symbol_or_var_bool{
5241     \stex_if_do_html:T {
5242       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5243     }
5244   }{
5245     % todo throw error

```

```

5246 }
5247 }
5248
5249 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5250 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5251 \stex_deactivate_macro:Nn \definiens {definition~environments}
5252 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5253

```

(End definition for definame. This function is documented on page 42.)

sdefinition (env.)

```

5254
5255 \keys_define:nn {stex / sdefinition }{
5256   type      .str_set_x:N = \sdefinitiontype,
5257   id        .str_set_x:N = \sdefinitionid,
5258   name      .str_set_x:N = \sdefinitionname,
5259   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5260   title     .tl_set:N    = \sdefinitiontitle
5261 }
5262 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5263   \str_clear:N \sdefinitiontype
5264   \str_clear:N \sdefinitionid
5265   \str_clear:N \sdefinitionname
5266   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5267   \tl_clear:N \sdefinitiontitle
5268   \keys_set:nn { stex / sdefinition }{ #1 }
5269 }
5270
5271 \NewDocumentEnvironment{sdefinition}{0{}}{
5272   \__stex_statements_sdefinition_args:n{ #1 }
5273   \stex_reactivate_macro:N \definiendum
5274   \stex_reactivate_macro:N \definame
5275   \stex_reactivate_macro:N \Definame
5276   \stex_reactivate_macro:N \premise
5277   \stex_reactivate_macro:N \definiens
5278   \stex_reactivate_macro:N \varbindforall
5279   \stex_if_smsmode:F{
5280     \seq_clear:N \l_tmpb_seq
5281     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5282       \tl_if_empty:nF{ ##1 }{
5283         \stex_get_symbol:n { ##1 }
5284         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5285           \l_stex_get_symbol_uri_str
5286         }
5287       }
5288     }
5289     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5290     \exp_args:Nnnx
5291     \begin{sbox}{\stex_annotate_env}{\definition}{\seq_use:Nn \l_tmpb_seq {,}}
5292     \str_if_empty:NF \sdefinitiontype {
5293       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5294     }
5295     \str_if_empty:NF \sdefinitionname {

```

```

5296 \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5297 }
5298 \clist_set:No \l_tmpa_clist \sdefinitiontype
5299 \tl_clear:N \l_tmpa_tl
5300 \clist_map_inline:Nn \l_tmpa_clist {
5301   \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5302     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5303   }
5304 }
5305 \tl_if_empty:NTF \l_tmpa_tl {
5306   \__stex_statements_sdefinition_start:
5307 }{
5308   \l_tmpa_tl
5309 }
5310 }
5311 \stex_ref_new_doc_target:n \sdefinitionid
5312 \stex_smsmode_do:
5313 }{
5314   \stex_suppress_html:n {
5315     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5316   }
5317   \stex_if_smsmode:F {
5318     \clist_set:No \l_tmpa_clist \sdefinitiontype
5319     \tl_clear:N \l_tmpa_tl
5320     \clist_map_inline:Nn \l_tmpa_clist {
5321       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5322         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5323       }
5324     }
5325     \tl_if_empty:NTF \l_tmpa_tl {
5326       \__stex_statements_sdefinition_end:
5327     }{
5328       \l_tmpa_tl
5329     }
5330     \end{stex_annotate_env}
5331   }
5332 }

```

\stexpatchdefinition

```

5333 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5334   \stex_par:\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5335     ~(\sdefinitiontitle)
5336   }~}
5337 }
5338 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5339
5340 \newcommand\stexpatchdefinition[3]{} {
5341   \str_set:Nx \l_tmpa_str{ #1 }
5342   \str_if_empty:NTF \l_tmpa_str {
5343     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5344     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5345   }{
5346     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5347     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5348     }
5349 }

```

(End definition for `\stexpatchdefinition`. This function is documented on page 47.)

`\inlinedef` inline:

```

5350 \keys_define:nn {stex / inlinedef }{
5351   type      .str_set_x:N = \sdefinitiontype,
5352   id        .str_set_x:N = \sdefinitionid,
5353   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5354   name      .str_set_x:N = \sdefinitionname
5355 }
5356 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5357   \str_clear:N \sdefinitiontype
5358   \str_clear:N \sdefinitionid
5359   \str_clear:N \sdefinitionname
5360   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5361   \keys_set:nn { stex / inlinedef }{ #1 }
5362 }
5363 \NewDocumentCommand \inlinedef { 0{} m } {
5364   \begingroup
5365   \__stex_statements_inlinedef_args:n{ #1 }
5366   \stex_reactivate_macro:N \definiendum
5367   \stex_reactivate_macro:N \definame
5368   \stex_reactivate_macro:N \Definame
5369   \stex_reactivate_macro:N \premise
5370   \stex_reactivate_macro:N \definiens
5371   \stex_reactivate_macro:N \varbindforall
5372   \stex_ref_new_doc_target:n \sdefinitionid
5373   \stex_if_smsmode:TF{\stex_suppress_html:n {
5374     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5375   }}{
5376     \seq_clear:N \l_tmpb_seq
5377     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5378       \tl_if_empty:nF{ ##1 }{
5379         \stex_get_symbol:n { ##1 }
5380         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5381           \l_stex_get_symbol_uri_str
5382         }
5383       }
5384     }
5385     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5386     \exp_args:Nnx
5387     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5388       \str_if_empty:NF \sdefinitiontype {
5389         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5390       }
5391       #2
5392       \str_if_empty:NF \sdefinitionname {
5393         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5394         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5395       }
5396     }
5397   }

```



```

5398 \endgroup
5399 \stex_smsmode_do:
5400 }

```

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

31.2 Assertions

`sassertion (env.)`

```

5401
5402 \keys_define:nn {stex / sassertion }{
5403   type      .str_set_x:N = \sassertiontype,
5404   id        .str_set_x:N = \sassertionid,
5405   title     .tl_set:N    = \sassertiontitle ,
5406   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5407   name      .str_set_x:N = \sassertionname
5408 }
5409 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5410   \str_clear:N \sassertiontype
5411   \str_clear:N \sassertionid
5412   \str_clear:N \sassertionname
5413   \clist_clear:N \l__stex_statements_sassertion_for_clist
5414   \tl_clear:N \sassertiontitle
5415   \keys_set:nn { stex / sassertion }{ #1 }
5416 }
5417
5418 %\tl_new:N \g__stex_statements_aftergroup_tl
5419
5420 \NewDocumentEnvironment{sassertion}{0{}}{
5421   \__stex_statements_sassertion_args:n{ #1 }
5422   \stex_reactivate_macro:N \premise
5423   \stex_reactivate_macro:N \conclusion
5424   \stex_reactivate_macro:N \varbindforall
5425   \stex_if_smsmode:F {
5426     \seq_clear:N \l_tmpb_seq
5427     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5428       \tl_if_empty:nF{ ##1 }{
5429         \stex_get_symbol:n { ##1 }
5430         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5431           \l_stex_get_symbol_uri_str
5432         }
5433       }
5434     }
5435     \exp_args:Nnnx
5436     \begin{sassertion}{\seq_use:Nn \l_tmpb_seq {,}}
5437     \str_if_empty:NF \sassertiontype {
5438       \stex_annotate_invisible:nnn{type}{\sassertiontype}{\sassertiontype}{}
5439     }
5440     \str_if_empty:NF \sassertionname {
5441       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{\sassertionname}{}
5442     }
5443     \clist_set:N \l_tmpa_clist \sassertiontype
5444     \tl_clear:N \l_tmpa_tl

```

```

5445 \clist_map_inline:Nn \l_tmpa_clist {
5446   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5447     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5448   }
5449 }
5450 \tl_if_empty:NTF \l_tmpa_tl {
5451   \__stex_statements_sassertion_start:
5452 }{
5453   \l_tmpa_tl
5454 }
5455 }
5456 \str_if_empty:NTF \sassertionid {
5457   \str_if_empty:NF \sassertionname {
5458     \stex_ref_new_doc_target:n {}
5459   }
5460 } {
5461   \stex_ref_new_doc_target:n \sassertionid
5462 }
5463 \stex_smsmode_do:
5464 ){
5465   \str_if_empty:NF \sassertionname {
5466     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5467     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5468   }
5469   \stex_if_smsmode:F {
5470     \clist_set:No \l_tmpa_clist \sassertiontype
5471     \tl_clear:N \l_tmpa_tl
5472     \clist_map_inline:Nn \l_tmpa_clist {
5473       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5474         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5475       }
5476     }
5477     \tl_if_empty:NTF \l_tmpa_tl {
5478       \__stex_statements_sassertion_end:
5479     }{
5480       \l_tmpa_tl
5481     }
5482     \end{stex_annotate_env}
5483   }
5484 }

```

\stexpatchassertion

```

5485
5486 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5487   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5488     (\sassertiontitle)
5489   }~}
5490 }
5491 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5492
5493 \newcommand\stexpatchassertion[3] [] {
5494   \str_set:Nx \l_tmpa_str{ #1 }
5495   \str_if_empty:NTF \l_tmpa_str {
5496     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

5497     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5498   }{
5499     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5500     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5501   }
5502 }

```

(End definition for `\stexpatchassertion`. This function is documented on page 47.)

`\inlineass` inline:

```

5503 \keys_define:nn {stex / inlineass }{
5504   type      .str_set_x:N = \sassertiontype,
5505   id        .str_set_x:N = \sassertionid,
5506   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5507   name      .str_set_x:N = \sassertionname
5508 }
5509 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5510   \str_clear:N \sassertiontype
5511   \str_clear:N \sassertionid
5512   \str_clear:N \sassertionname
5513   \clist_clear:N \l__stex_statements_sassertion_for_clist
5514   \keys_set:nn { stex / inlineass }{ #1 }
5515 }
5516 \NewDocumentCommand \inlineass { 0{} m } {
5517   \beginngroup
5518   \stex_reactivate_macro:N \premise
5519   \stex_reactivate_macro:N \conclusion
5520   \stex_reactivate_macro:N \varbindforall
5521   \__stex_statements_inlineass_args:n{ #1 }
5522   \str_if_empty:NTF \sassertionid {
5523     \str_if_empty:NF \sassertionname {
5524       \stex_ref_new_doc_target:n {}
5525     }
5526   } {
5527     \stex_ref_new_doc_target:n \sassertionid
5528   }
5529
5530   \stex_if_smsmode:TF{
5531     \str_if_empty:NF \sassertionname {
5532       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5533       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5534     }
5535   }{
5536     \seq_clear:N \l_tmpb_seq
5537     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5538       \tl_if_empty:nF{ ##1 }{
5539         \stex_get_symbol:n { ##1 }
5540         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5541           \l_stex_get_symbol_uri_str
5542         }
5543       }
5544     }
5545     \exp_args:Nnx
5546     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{

```

```

5547 \str_if_empty:NF \sassertiontype {
5548 \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{ }
5549 }
5550 #2
5551 \str_if_empty:NF \sassertionname {
5552 \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassertionname}}
5553 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5554 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5555 }
5556 }
5557 }
5558 \endgroup
5559 \stex_smsmode_do:
5560 }

```

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

31.3 Examples

`sexample (env.)`

```

5561
5562 \keys_define:nn {stex / sexample }{
5563   type      .str_set_x:N = \exampletype,
5564   id        .str_set_x:N = \sexampleid,
5565   title     .tl_set:N    = \sexampletitle,
5566   name      .str_set_x:N = \sexamplename ,
5567   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5568 }
5569 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5570   \str_clear:N \sexampletype
5571   \str_clear:N \sexampleid
5572   \str_clear:N \sexamplename
5573   \tl_clear:N \sexampletitle
5574   \clist_clear:N \l__stex_statements_sexample_for_clist
5575   \keys_set:nn { stex / sexample }{ #1 }
5576 }
5577
5578 \NewDocumentEnvironment{sexample}{0{}}{
5579   \__stex_statements_sexample_args:n{ #1 }
5580   \stex_reactivate_macro:N \premise
5581   \stex_reactivate_macro:N \conclusion
5582   \stex_if_smsmode:F {
5583     \seq_clear:N \l_tmpb_seq
5584     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5585       \tl_if_empty:nF{ ##1 }{
5586         \stex_get_symbol:n { ##1 }
5587         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5588           \l_stex_get_symbol_uri_str
5589         }
5590       }
5591     }
5592     \exp_args:Nnnx
5593     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```

```

5594 \str_if_empty:NF \sexamplotype {
5595   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5596 }
5597 \str_if_empty:NF \sexamplename {
5598   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5599 }
5600 \clist_set:No \l_tmpa_clist \sexamplotype
5601 \tl_clear:N \l_tmpa_tl
5602 \clist_map_inline:Nn \l_tmpa_clist {
5603   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5604     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5605   }
5606 }
5607 \tl_if_empty:NTF \l_tmpa_tl {
5608   \__stex_statements_sexample_start:
5609 }{
5610   \l_tmpa_tl
5611 }
5612 }
5613 \str_if_empty:NF \sexampleid {
5614   \stex_ref_new_doc_target:n \sexampleid
5615 }
5616 \stex_smsmode_do:
5617 }{
5618   \str_if_empty:NF \sexamplename {
5619     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5620   }
5621   \stex_if_smsmode:F {
5622     \clist_set:No \l_tmpa_clist \sexamplotype
5623     \tl_clear:N \l_tmpa_tl
5624     \clist_map_inline:Nn \l_tmpa_clist {
5625       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5626         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5627       }
5628     }
5629     \tl_if_empty:NTF \l_tmpa_tl {
5630       \__stex_statements_sexample_end:
5631     }{
5632       \l_tmpa_tl
5633     }
5634     \end{stex_annotate_env}
5635   }
5636 }

```

\stexpatchexample

```

5637
5638 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5639   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
5640     (\sexampltitle)
5641   }~}
5642 }
5643 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5644
5645 \newcommand\stexpatchexample[3]{} {

```

```

5646 \str_set:Nx \l_tmpa_str{ #1 }
5647 \str_if_empty:NTF \l_tmpa_str {
5648   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5649   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5650 }{
5651   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5652   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5653 }
5654 }

```

(End definition for `\stexpatchexample`. This function is documented on page 47.)

`\inlineex` inline:

```

5655 \keys_define:nn {stex / inlineex }{
5656   type      .str_set_x:N = \sexamplotype,
5657   id        .str_set_x:N = \sexampleid,
5658   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5659   name      .str_set_x:N = \sexamplename
5660 }
5661 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5662   \str_clear:N \sexamplotype
5663   \str_clear:N \sexampleid
5664   \str_clear:N \sexamplename
5665   \clist_clear:N \l__stex_statements_sexample_for_clist
5666   \keys_set:nn { stex / inlineex }{ #1 }
5667 }
5668 \NewDocumentCommand \inlineex { 0{} m } {
5669   \begingroup
5670   \stex_reactivate_macro:N \premise
5671   \stex_reactivate_macro:N \conclusion
5672   \__stex_statements_inlineex_args:n{ #1 }
5673   \str_if_empty:NF \sexampleid {
5674     \stex_ref_new_doc_target:n \sexampleid
5675   }
5676   \stex_if_smsmode:TF{
5677     \str_if_empty:NF \sexamplename {
5678       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5679     }
5680   }{
5681     \seq_clear:N \l_tmpb_seq
5682     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5683       \tl_if_empty:nF{ ##1 }{
5684         \stex_get_symbol:n { ##1 }
5685         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5686           \l_stex_get_symbol_uri_str
5687         }
5688       }
5689     }
5690     \exp_args:Nnx
5691     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
5692       \str_if_empty:NF \sexamplotype {
5693         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5694       }
5695       #2

```

```

5696     \str_if_empty:NF \sexamplename {
5697       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5698       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5699     }
5700   }
5701 }
5702 \endgroup
5703 \stex_smsmode_do:
5704 }

```

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

31.4 Logical Paragraphs

`sparagraph` (*env.*)

```

5705 \keys_define:nn { stex / spparagraph } {
5706   id      .str_set_x:N = \sparagraphid ,
5707   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
5708   type    .str_set_x:N = \sparagraphtype ,
5709   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5710   from    .tl_set:N    = \sparagraphfrom ,
5711   to      .tl_set:N    = \sparagraphto ,
5712   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
5713   name    .str_set:N    = \sparagraphname ,
5714   imports .tl_set:N    = \l__stex_statements_sparagraph_imports_tl
5715 }
5716
5717 \cs_new_protected:Nn \stex_sparagraph_args:n {
5718   \tl_clear:N \l_stex_sparagraph_title_tl
5719   \tl_clear:N \sparagraphfrom
5720   \tl_clear:N \sparagraphto
5721   \tl_clear:N \l_stex_sparagraph_start_tl
5722   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5723   \str_clear:N \sparagraphid
5724   \str_clear:N \sparagraphtype
5725   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5726   \str_clear:N \sparagraphname
5727   \keys_set:nn { stex / spparagraph } { #1 }
5728 }
5729 \newif\if@in@omtext\@in@omtextfalse
5730
5731 \NewDocumentEnvironment {sparagraph} { 0{} } {
5732   \stex_sparagraph_args:n { #1 }
5733   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5734     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5735   }{
5736     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5737   }
5738   \@in@omtexttrue
5739   \stex_if_smsmode:F {
5740     \seq_clear:N \l_tmpb_seq
5741     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5742       \tl_if_empty:nF{ ##1 }{

```

```

5743     \stex_get_symbol:n { ##1 }
5744     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5745         \l_stex_get_symbol_uri_str
5746     }
5747 }
5748 }
5749 \exp_args:Nnnx
5750 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
5751 \str_if_empty:NF \sparagraphtype {
5752     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5753 }
5754 \str_if_empty:NF \sparagraphfrom {
5755     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5756 }
5757 \str_if_empty:NF \sparagraphto {
5758     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5759 }
5760 \str_if_empty:NF \sparagraphname {
5761     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5762 }
5763 \clist_set:No \l_tmpa_clist \sparagraphtype
5764 \tl_clear:N \l_tmpa_tl
5765 \clist_map_inline:Nn \sparagraphtype {
5766     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5767         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5768     }
5769 }
5770 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5771 \tl_if_empty:NTF \l_tmpa_tl {
5772     \__stex_statements_sparagraph_start:
5773 }{
5774     \l_tmpa_tl
5775 }
5776 }
5777 \clist_set:No \l_tmpa_clist \sparagraphtype
5778 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5779     {
5780         \stex_reactivate_macro:N \definiendum
5781         \stex_reactivate_macro:N \definame
5782         \stex_reactivate_macro:N \Definame
5783         \stex_reactivate_macro:N \premise
5784         \stex_reactivate_macro:N \definiens
5785     }
5786 \str_if_empty:NTF \sparagraphid {
5787     \str_if_empty:NTF \sparagraphname {
5788         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5789             \stex_ref_new_doc_target:n {}
5790         }
5791     } {
5792         \stex_ref_new_doc_target:n {}
5793     }
5794 } {
5795     \stex_ref_new_doc_target:n \sparagraphid
5796 }

```



```

5797 \exp_args:NNx
5798 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5799   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5800     \tl_if_empty:nF{ ##1 }{
5801       \stex_get_symbol:n { ##1 }
5802       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5803     }
5804   }
5805 }
5806 \stex_smsmode_do:
5807 \ignorespacesandpars
5808 }{
5809   \str_if_empty:NF \sparagraphname {
5810     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5811     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5812   }
5813   \stex_if_smsmode:F {
5814     \clist_set:No \l_tmpa_clist \sparagraphtype
5815     \tl_clear:N \l_tmpa_tl
5816     \clist_map_inline:Nn \l_tmpa_clist {
5817       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5818         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5819       }
5820     }
5821     \tl_if_empty:NTF \l_tmpa_tl {
5822       \__stex_statements_sparagraph_end:
5823     }{
5824       \l_tmpa_tl
5825     }
5826     \end{stex_annotate_env}
5827   }
5828 }

```

\stexpatchparagraph

```

5829
5830 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5831   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5832     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5833       \titleemph{\l_stex_sparagraph_title_tl}:~
5834     }
5835   }{
5836     \titleemph{\l_stex_sparagraph_start_tl}~
5837   }
5838 }
5839 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5840
5841 \newcommand\stexpatchparagraph[3] [] {
5842   \str_set:Nx \l_tmpa_str{ #1 }
5843   \str_if_empty:NTF \l_tmpa_str {
5844     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5845     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5846   }{
5847     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5848     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

5849     }
5850 }
5851
5852 \keys_define:nn { stex / inlinepara } {
5853   id      .str_set:N = \sparagraphid ,
5854   type    .str_set:N = \sparagraphtype ,
5855   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5856   from    .tl_set:N   = \sparagraphfrom ,
5857   to      .tl_set:N   = \sparagraphto ,
5858   name    .str_set:N   = \sparagraphname
5859 }
5860 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5861   \tl_clear:N \sparagraphfrom
5862   \tl_clear:N \sparagraphto
5863   \str_clear:N \sparagraphid
5864   \str_clear:N \sparagraphtype
5865   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5866   \str_clear:N \sparagraphname
5867   \keys_set:nn { stex / inlinepara }{ #1 }
5868 }
5869 \NewDocumentCommand \inlinepara { O{} m } {
5870   \begin{group}
5871     \__stex_statements_inlinepara_args:n{ #1 }
5872     \clist_set:Nn \l_tmpa_clist \sparagraphtype
5873     \str_if_empty:NTF \sparagraphid {
5874       \str_if_empty:NTF \sparagraphname {
5875         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5876           \stex_ref_new_doc_target:n {}
5877         }
5878       } {
5879         \stex_ref_new_doc_target:n {}
5880       }
5881     } {
5882       \stex_ref_new_doc_target:n \sparagraphid
5883     }
5884     \stex_if_smsmode:TF{
5885       \str_if_empty:NF \sparagraphname {
5886         \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5887         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5888       }
5889     }{
5890       \seq_clear:N \l_tmpb_seq
5891       \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5892         \tl_if_empty:nF{ ##1 }{
5893           \stex_get_symbol:n { ##1 }
5894           \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5895             \l_stex_get_symbol_uri_str
5896           }
5897         }
5898       }
5899       \exp_args:Nnx
5900       \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
5901         \str_if_empty:NF \sparagraphtype {
5902           \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

5903     }
5904     \str_if_empty:NF \sparagraphfrom {
5905         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5906     }
5907     \str_if_empty:NF \sparagraphto {
5908         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5909     }
5910     \str_if_empty:NF \sparagraphname {
5911         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5912         \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5913         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5914     }
5915     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5916         \clist_map_inline:Nn \l_tmpb_seq {
5917             \stex_ref_new_sym_target:n {##1}
5918         }
5919     }
5920     #2
5921 }
5922 }
5923 \endgroup
5924 \stex_smsmode_do:
5925 }
5926

```

(End definition for `\stexpatchparagraph`. This function is documented on page [47](#).)

```

5927 </package>

```

Chapter 32

The Implementation

```
5928 <*package>
5929 <@@=stex_sproof>
5930
5931 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
5932
```

32.1 Proofs

We first define some keys for the proof environment.

```
5933 \keys_define:nn { stex / spf } {
5934   id          .str_set_x:N = \spfid,
5935   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5936   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5937   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5938   type        .str_set_x:N = \spftype,
5939   title       .tl_set:N    = \spftitle,
5940   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5941   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5942   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5943 }
5944 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5945   \str_clear:N \spfid
5946   \tl_clear:N \l__stex_sproof_spf_for_tl
5947   \tl_clear:N \l__stex_sproof_spf_from_tl
5948   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5949   \str_clear:N \spftype
5950   \tl_clear:N \spftitle
5951   \tl_clear:N \l__stex_sproof_spf_continues_tl
5952   \tl_clear:N \l__stex_sproof_spf_functions_tl
5953   \tl_clear:N \l__stex_sproof_spf_method_tl
5954   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5955   \keys_set:nn { stex / spf }{ #1 }
5956 }
```

```
\c__stex_sproof_flow_str We define this macro, so that we can test whether the display key has the value flow
5957 \str_set:Nn\c__stex_sproof_flow_str{inline}
```

(End definition for `\c__stex_sproof_flow_str.`)

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.

```

5958 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5959 \cs_new_protected:Npn \sproofnumber {
5960   \int_set:Nn \l_tmpa_int {1}
5961   \bool_while_do:nn {
5962     \int_compare_p:nNn {
5963       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5964     } > 0
5965   }{
5966     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5967     \int_incr:N \l_tmpa_int
5968   }
5969 }
5970 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5971   \int_set:Nn \l_tmpa_int {1}
5972   \bool_while_do:nn {
5973     \int_compare_p:nNn {
5974       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5975     } > 0
5976   }{
5977     \int_incr:N \l_tmpa_int
5978   }
5979   \int_compare:nNnF \l_tmpa_int = 1 {
5980     \int_decr:N \l_tmpa_int
5981   }
5982   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5983     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5984   }
5985 }
5986
5987 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5988   \int_set:Nn \l_tmpa_int {1}
5989   \bool_while_do:nn {
5990     \int_compare_p:nNn {
5991       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5992     } > 0
5993   }{
5994     \int_incr:N \l_tmpa_int
5995   }
5996   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5997 }
5998
5999 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6000   \int_set:Nn \l_tmpa_int {1}
6001   \bool_while_do:nn {

```

```

6002     \int_compare_p:nNn {
6003       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6004     } > 0
6005   }{
6006     \int_incr:N \l_tmpa_int
6007   }
6008   \int_decr:N \l_tmpa_int
6009   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6010 }

```

\sproofend This macro places a little box at the end of the line if there is space, or at the end of the next line if there isn't

```

6011 \def\sproof@box{
6012   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6013 }
6014 \def\sproofend{
6015   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6016     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6017   }
6018 }

```

(End definition for \sproofend. This function is documented on page 47.)

spf@*@kw

```

6019 \def\spf@proofsketch@kw{Proof~Sketch}
6020 \def\spf@proof@kw{Proof}
6021 \def\spf@step@kw{Step}

```

(End definition for spf@*@kw. This function is documented on page ??.)

For the other languages, we set up triggers

```

6022 \AddToHook{begindocument}{
6023   \ltx@ifpackageloaded{babel}{
6024     \makeatletter
6025     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6026     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6027       \input{sproof-ngerman.ldf}
6028     }
6029     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6030       \input{sproof-finnish.ldf}
6031     }
6032     \clist_if_in:NnT \l_tmpa_clist {french}{
6033       \input{sproof-french.ldf}
6034     }
6035     \clist_if_in:NnT \l_tmpa_clist {russian}{
6036       \input{sproof-russian.ldf}
6037     }
6038     \makeatother
6039   }{}
6040 }

```

spfsketch

```

6041 \newcommand\spfsketch[2] [] {
6042   \begin{group}
6043   \let \premise \stex_proof_premise:

```

```

6044 \__stex_sproof_spf_args:n{#1}
6045 \stex_if_smsmode:TF {
6046   \str_if_empty:NF \spfid {
6047     \stex_ref_new_doc_target:n \spfid
6048   }
6049 }{
6050   \seq_clear:N \l_tmpa_seq
6051   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6052     \tl_if_empty:nF{ ##1 }{
6053       \stex_get_symbol:n { ##1 }
6054       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6055         \l_stex_get_symbol_uri_str
6056       }
6057     }
6058   }
6059   \exp_args:Nnx
6060   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6061     \str_if_empty:NF \spftype {
6062       \stex_annotate_invisible:nnn{type}{\spftype}{
6063     }
6064     \clist_set:Nn \l_tmpa_clist \spftype
6065     \tl_set:Nn \l_tmpa_tl {
6066       \titleemph{
6067         \tl_if_empty:NTF \spftitle {
6068           \spf@proofsketch@kw
6069         }{
6070           \spftitle
6071         }
6072       }~
6073     }
6074     \clist_map_inline:Nn \l_tmpa_clist {
6075       \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6076         \tl_clear:N \l_tmpa_tl
6077       }
6078     }
6079     \str_if_empty:NF \spfid {
6080       \stex_ref_new_doc_target:n \spfid
6081     }
6082     \l_tmpa_tl #2 \sproofend
6083   }
6084 }
6085 \endgroup
6086 \stex_smsmode_do:
6087 }
6088

```

(End definition for *spfsketch*. This function is documented on page 45.)

spfeq This is very similar to `\spfsketch`, but uses a computation array⁷⁸

```

6089 \newenvironment{spfeq}[2][ ]{
6090   \__stex_sproof_spf_args:n{#1}

```

⁷EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

⁸EDNOTE: document above

```

6091 \let \premise \stex_proof_premise:
6092 \stex_if_smsmode:TF {
6093   \str_if_empty:NF \spfid {
6094     \stex_ref_new_doc_target:n \spfid
6095   }
6096 }{
6097   \seq_clear:N \l_tmpa_seq
6098   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6099     \tl_if_empty:nF{ ##1 }{
6100       \stex_get_symbol:n { ##1 }
6101       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6102         \l_stex_get_symbol_uri_str
6103       }
6104     }
6105   }
6106   \exp_args:Nnnx
6107   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
6108   \str_if_empty:NF \spftype {
6109     \stex_annotate_invisible:nnn{type}{\spftype}{ }
6110   }
6111
6112   \clist_set:No \l_tmpa_clist \spftype
6113   \tl_clear:N \l_tmpa_tl
6114   \clist_map_inline:Nn \l_tmpa_clist {
6115     \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
6116       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
6117     }
6118     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6119       \tl_set:Nn \l_tmpa_tl {\use:n{}}
6120     }
6121   }
6122   \tl_if_empty:NTF \l_tmpa_tl {
6123     \__stex_sproof_spfeq_start:
6124   }{
6125     \l_tmpa_tl
6126   }{~#2}
6127   \str_if_empty:NF \spfid {
6128     \stex_ref_new_doc_target:n \spfid
6129   }
6130   \begin{displaymath}\begin{array}{rc1l}
6131   }
6132   \stex_smsmode_do:
6133 }{
6134   \stex_if_smsmode:F {
6135     \end{array}\end{displaymath}
6136     \clist_set:No \l_tmpa_clist \spftype
6137     \tl_clear:N \l_tmpa_tl
6138     \clist_map_inline:Nn \l_tmpa_clist {
6139       \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
6140         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
6141       }
6142     }
6143     \tl_if_empty:NTF \l_tmpa_tl {
6144       \__stex_sproof_spfeq_end:

```



```

6145     }{
6146       \l_tmpa_tl
6147     }
6148     \end{stex_annotate_env}
6149   }
6150 }
6151
6152 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
6153   \titleemph{
6154     \tl_if_empty:NTF \spftitle {
6155       \spf@proof@kw
6156     }{
6157       \spftitle
6158     }
6159   }:
6160 }
6161 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
6162
6163 \newcommand\stexpatchspfeq[3] [] {
6164   \str_set:Nx \l_tmpa_str{ #1 }
6165   \str_if_empty:NTF \l_tmpa_str {
6166     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
6167     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
6168   }{
6169     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
6170     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
6171   }
6172 }
6173

```

(End definition for `spfeq`. This function is documented on page ??.)

sproof (*env.*) In this environment, we initialize the proof depth counter `\count10` to 10, and set up the description environment that will take the proof steps. At the end of the proof, we position the proof end into the last line.

```

6174 \newenvironment{sproof}[2] []{
6175   \let \premise \stex_proof_premise:
6176   \intarray_gzero:N \l__stex_sproof_counter_intarray
6177   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6178   \__stex_sproof_spf_args:n{#1}
6179   \stex_if_smsmode:TF {
6180     \str_if_empty:NF \spfid {
6181       \stex_ref_new_doc_target:n \spfid
6182     }
6183   }{
6184     \seq_clear:N \l_tmpa_seq
6185     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6186       \tl_if_empty:NF{ ##1 }{
6187         \stex_get_symbol:n { ##1 }
6188         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6189           \l_stex_get_symbol_uri_str
6190         }
6191       }
6192     }

```

```

6193 \exp_args:Nnnx
6194 \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
6195 \str_if_empty:NF \spftype {
6196   \stex_annotate_invisible:nnn{type}{\spftype}{}
6197 }
6198
6199 \clist_set:No \l_tmpa_clist \spftype
6200 \tl_clear:N \l_tmpa_tl
6201 \clist_map_inline:Nn \l_tmpa_clist {
6202   \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
6203     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
6204   }
6205   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6206     \tl_set:Nn \l_tmpa_tl {\use:n{}}
6207   }
6208 }
6209 \tl_if_empty:NTF \l_tmpa_tl {
6210   \__stex_sproof_sproof_start:
6211 }{
6212   \l_tmpa_tl
6213 }{~#2}
6214 \str_if_empty:NF \spfid {
6215   \stex_ref_new_doc_target:n \spfid
6216 }
6217 \begin{description}
6218 }
6219 \stex_smsmode_do:
6220 }{
6221   \stex_if_smsmode:F{
6222     \end{description}
6223     \clist_set:No \l_tmpa_clist \spftype
6224     \tl_clear:N \l_tmpa_tl
6225     \clist_map_inline:Nn \l_tmpa_clist {
6226       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6227         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6228       }
6229     }
6230     \tl_if_empty:NTF \l_tmpa_tl {
6231       \__stex_sproof_sproof_end:
6232     }{
6233       \l_tmpa_tl
6234     }
6235     \end{stex_annotate_env}
6236   }
6237 }
6238
6239 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6240   \par\noindent\titleemph{
6241     \tl_if_empty:NTF \spftype {
6242       \spf@proof@kw
6243     }{
6244       \spftype
6245     }
6246   }::

```

```

6247 }
6248 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6249
6250 \newcommand\stexpatchproof[3] [] {
6251   \str_set:Nx \l_tmpa_str{ #1 }
6252   \str_if_empty:NTF \l_tmpa_str {
6253     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6254     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6255   }{
6256     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6257     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6258   }
6259 }

```

\spfidea

```

6260 \newcommand\spfidea[2] []{
6261   \__stex_sproof_spf_args:n{#1}
6262   \titleemph{
6263     \tl_if_empty:NTF \spftype {Proof-Idea}{
6264       \spftype
6265     }:
6266   }~#2
6267   \sproofend
6268 }

```

(End definition for \spfidea. This function is documented on page 45.)

The next two environments (proof steps) and comments, are mostly semantical, they take KeyVal arguments that specify their semantic role. In draft mode, they read these values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

spfstep (env.)

```

6269 \newenvironment{spfstep}[1] []{
6270   \__stex_sproof_spf_args:n{#1}
6271   \stex_if_smsmode:TF {
6272     \str_if_empty:NF \spfid {
6273       \stex_ref_new_doc_target:n \spfid
6274     }
6275   }{
6276     \@in@omtexttrue
6277     \clist_set:No \l_tmpa_clist \spftype
6278     \tl_set:Nn \l_tmpa_tl {
6279       \item[\sproofnumber]
6280       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6281     }
6282     \clist_map_inline:Nn \l_tmpa_clist {
6283       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6284         \tl_clear:N \l_tmpa_tl
6285       }
6286     }
6287     \l_tmpa_tl
6288     \seq_clear:N \l_tmpa_seq
6289     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {

```

```

6290     \tl_if_empty:nF{ ##1 }{
6291       \stex_get_symbol:n { ##1 }
6292       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6293         \l_stex_get_symbol_uri_str
6294       }
6295     }
6296   }
6297   \exp_args:Nnnx
6298   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6299   \str_if_empty:NF \spftype {
6300     \stex_annotate_invisible:nnn{type}{\spftype}{}}
6301   }
6302   \tl_if_empty:NF \spftitle {
6303     {(\titleemph{\spftitle})\enspace}
6304   }
6305   \str_if_empty:NF \spfid {
6306     \stex_ref_new_doc_target:n \spfid
6307   }
6308 }
6309 \stex_smsmode_do:
6310 \ignorespacesandpars
6311 }{
6312   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6313     \__stex_sproof_inc_counter:
6314   }
6315   \stex_if_smsmode:F {
6316     \end{stex_annotate_env}
6317   }
6318 }

```

spfcomment (*env.*)

```

6319 \newenvironment{spfcomment}[1][{}]{
6320   \__stex_sproof_spf_args:n{#1}
6321   \clist_set:Nn \l_tmpa_clist \spftype
6322   \tl_set:Nn \l_tmpa_tl {
6323     \item[\sproofnumber]
6324     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6325   }
6326   \clist_map_inline:Nn \l_tmpa_clist {
6327     \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6328       \tl_clear:N \l_tmpa_tl
6329     }
6330   }
6331   \l_tmpa_tl
6332 }{
6333   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6334     \__stex_sproof_inc_counter:
6335   }
6336 }

```

The next two environments also take a `KeyVal` argument, but also a regular one, which contains a start text. Both environments start a new numbered proof level.

subproof (*env.*) In the **subproof** environment, a new (lower-level) **proproof** environment is started.

```

6337 \newenvironment{subproof}[2][]{
6338   \_stex_sproof_spf_args:n{#1}
6339   \stex_if_smsmode:TF{
6340     \str_if_empty:NF \spfid {
6341       \stex_ref_new_doc_target:n \spfid
6342     }
6343   }{
6344     \seq_clear:N \l_tmpa_seq
6345     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6346       \tl_if_empty:nF{ ##1 }{
6347         \stex_get_symbol:n { ##1 }
6348         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6349           \l_stex_get_symbol_uri_str
6350         }
6351       }
6352     }
6353     \exp_args:Nnnx
6354     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6355     \str_if_empty:NF \spftype {
6356       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6357     }
6358
6359     \clist_set:No \l_tmpa_clist \spftype
6360     \tl_set:Nn \l_tmpa_tl {
6361       \item[\sproofnumber]
6362       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6363     }
6364     \clist_map_inline:Nn \l_tmpa_clist {
6365       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6366         \tl_clear:N \l_tmpa_tl
6367       }
6368     }
6369     \l_tmpa_tl
6370     \tl_if_empty:NF \spftitle {
6371       {(\titleemph{\spftitle})\enspace}
6372     }
6373     {~#2}
6374     \str_if_empty:NF \spfid {
6375       \stex_ref_new_doc_target:n \spfid
6376     }
6377   }
6378   \_stex_sproof_add_counter:
6379   \stex_smsmode_do:
6380 }{
6381   \_stex_sproof_remove_counter:
6382   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6383     \_stex_sproof_inc_counter:
6384   }
6385   \stex_if_smsmode:F{
6386     \end{stex_annotate_env}
6387   }
6388 }

```

spfcases (*env.*) In the **pfcases** environment, the start text is displayed as the first comment of the proof.

```

6389 \newenvironment{spfcases}[2] [] {
6390   \tl_if_empty:nTF{#1}{
6391     \begin{subproof}[method=by-cases]{#2}
6392   }{
6393     \begin{subproof}[#1,method=by-cases]{#2}
6394   }
6395 }{
6396   \end{subproof}
6397 }

```

`spfcase (env.)` In the `pfcase` environment, the start text is displayed specification of the case after the `\item`

```

6398 \newenvironment{spfcase}[2] [] {
6399   \__stex_sproof_spf_args:n{#1}
6400   \stex_if_smsmode:TF {
6401     \str_if_empty:NF \spfid {
6402       \stex_ref_new_doc_target:n \spfid
6403     }
6404   }{
6405     \seq_clear:N \l_tmpa_seq
6406     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6407       \tl_if_empty:nF{ ##1 }{
6408         \stex_get_symbol:n { ##1 }
6409         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6410           \l_stex_get_symbol_uri_str
6411         }
6412       }
6413     }
6414     \exp_args:Nnnx
6415     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6416     \str_if_empty:NF \spftype {
6417       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6418     }
6419     \clist_set:Nn \l_tmpa_clist \spftype
6420     \tl_set:Nn \l_tmpa_tl {
6421       \item[\sproofnumber]
6422       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6423     }
6424     \clist_map_inline:Nn \l_tmpa_clist {
6425       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6426         \tl_clear:N \l_tmpa_tl
6427       }
6428     }
6429     \l_tmpa_tl
6430     \tl_if_empty:nF{#2}{
6431       \titleemph{#2}:~
6432     }
6433   }
6434   \__stex_sproof_add_counter:
6435   \stex_smsmode_do:
6436 }{
6437   \__stex_sproof_remove_counter:
6438   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6439     \__stex_sproof_inc_counter:

```

```

6440 }
6441 \stex_if_smsmode:F{
6442   \clist_set:No \l_tmpa_clist \spftype
6443   \tl_set:Nn \l_tmpa_tl{\sproofend}
6444   \clist_map_inline:Nn \l_tmpa_clist {
6445     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6446       \tl_clear:N \l_tmpa_tl
6447     }
6448   }
6449   \l_tmpa_tl
6450   \end{stex_annotate_env}
6451 }
6452 }

```

`spfcase (env.)` similar to `spfcase`, takes a third argument.

```

6453 \newcommand\spfcasesketch[3] [] {
6454   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6455 }

```

32.2 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```

6456 \keys_define:nn { stex / just }{
6457   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6458   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6459   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6460   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6461 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.⁹

\spfjust

```

6462 \newcommand\spfjust[1] [] {}

```

(End definition for `\spfjust`. This function is documented on page 46.)

\premise

```

6463 \newcommand\stex_proof_premise:[2] [] {#2}

```

(End definition for `\premise`. This function is documented on page 46.)

\justarg the `\justarg` macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```

6464 \newcommand\justarg[2] [] {#2}
6465 \</package>

```

(End definition for `\justarg`. This function is documented on page 46.)

Some auxiliary code, and clean up to be executed at the end of the package.

⁹EdNOTE: need to do something about the premise in draft mode.

Chapter 33

STEX -Others Implementation

```
6466 <*package>
6467
6468 %%%%%%%%%% others.dtx %%%%%%%%%%
6469
6470 <@@=stex_others>
        Warnings and error messages
6471 % None

\MSC Math subject classifier

6472 \NewDocumentCommand \MSC {m} {
6473 % TODO
6474 }

(End definition for \MSC. This function is documented on page ??.)
        Patching tikzinput, if loaded

6475 \@ifpackageloaded{tikzinput}{
6476 \RequirePackage{stex-tikzinput}
6477 }{}
6478
6479 \bool_if:NT \c_stex_persist_mode_bool {
6480 \let__stex_notation_restore_notation_old:nnnnn
6481 \__stex_notation_restore_notation:nnnnn
6482 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6483 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6484 \ExplSyntaxOn
6485 }
6486 \def__stex_notation_restore_notation:nnnnn{
6487 \ExplSyntaxOff
6488 \catcode'\sim10
6489 \__stex_notation_restore_notation_new:nnnnn
6490 }
6491 \input{\jobname.sms}
6492 \let__stex_notation_restore_notation:nnnnn
6493 \__stex_notation_restore_notation_old:nnnnn
6494 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```



```

6495     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6496         \l_tmpa_str
6497     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6498         \c_stex_mathhub_main_manifest_prop
6499     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6500 }
6501 }
6502 </package>

```

Chapter 34

STEX -Metatheory Implementation

```
6503 <*package>
6504 <@@=stex_modules>
6505
6506 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6507
6508 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6509 \begingroup
6510 \stex_module_setup:nn{
6511   ns=\c_stex_metatheory_ns_str,
6512   meta=NONE
6513 }{Metatheory}
6514 \stex_reactivate_macro:N \symdecl
6515 \stex_reactivate_macro:N \notation
6516 \stex_reactivate_macro:N \symdef
6517 \ExplSyntaxOff
6518 \csname stex_suppress_html:n\endcsname{
6519   % is-a (a:A, a \in A, a is an A, etc.)
6520   \symdecl{isa}[args=ai]
6521   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6522   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6523   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6524
6525   % bind (\forall, \Pi, \lambda etc.)
6526   \symdecl{bind}[args=Bi,assoc=pre]
6527   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6528   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6529   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6530
6531   % implicit bind
6532   \symdecl{implicitbind}[args=Bi,assoc=pre]
6533   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6534   \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6535   \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6536
6537   % dummy variable
```

```

6538 \symdecl{dummyvar}
6539 \notation{dummyvar}[underscore]{\comp\_}
6540 \notation{dummyvar}[dot]{\comp\cdot}
6541 \notation{dummyvar}[dash]{\comp{\rm --}}
6542
6543 %fromto (function space, Hom-set, implication etc.)
6544 \symdecl{fromto}[args=ai]
6545 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6546 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6547
6548 % mapto (lambda etc.)
6549 \symdecl{mapto}[args=Bi]
6550 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6551 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6552 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6553
6554 % function/operator application
6555 \symdecl{apply}[args=ia]
6556 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6557 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6558
6559 % collection of propositions/booleans/truth values
6560 \symdecl{prop}[name=proposition]
6561 \notation{prop}[prop]{\comp{\rm prop}}
6562 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6563
6564 \symdecl{judgmentholds}[args=1]
6565 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6566
6567 % sequences
6568 \symdecl{seqtype}[args=1]
6569 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6570
6571 \symdecl{seqexpr}[args=a]
6572 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6573
6574 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
6575 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6576 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6577 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp
6578 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp
6579 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6580 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6581 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6582 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6583
6584 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\{#1\}_{#2}}
6585 \notation{sequence-index}[ui,prec=nobrackets]{\{#1\}^{\#2}}
6586
6587 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6588 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6589 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
6590
6591 % nat literals

```

```

6592 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6593
6594 % letin (''let'', local definitions, variable substitution)
6595 \symdecl{letin}[args=bii]
6596 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
6597 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
6598 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
6599
6600 % structures
6601 \symdecl*{module-type}[args=1]
6602 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6603 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6604 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6605
6606 % objects
6607 \symdecl{object}
6608 \notation{object}{\comp{\mathtt{OBJECT}}}
6609
6610 }
6611
6612 % The following are abbreviations in the sTeX corpus that are left over from earlier
6613 % developments. They will eventually be phased out.
6614
6615 \ExplSyntaxOn
6616 \stex_add_to_current_module:n{
6617   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6618   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6619   \def\livar{\csname sequence-index\endcsname[li]}
6620   \def\uivar{\csname sequence-index\endcsname[ui]}
6621   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6622   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6623 }
6624 \__stex_modules_end_module:
6625 \endgroup
6626 \</package>

```

Chapter 35

Tikzinput Implementation

```
6627 <@@=tikzinput>
6628 <*package>
6629
6630 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6631
6632 \ProvidesExplPackage{tikzinput}{2022/05/24}{3.1.0}{tikzinput package}
6633 \RequirePackage{l3keys2e}
6634
6635 \keys_define:nn { tikzinput } {
6636   image .bool_set:N = \c_tikzinput_image_bool,
6637   image .default:n = false ,
6638   unknown .code:n = {}
6639 }
6640
6641 \ProcessKeysOptions { tikzinput }
6642
6643 \bool_if:NTF \c_tikzinput_image_bool {
6644   \RequirePackage{graphicx}
6645
6646   \providecommand\usetikzlibrary[]{}
6647   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6648 }{
6649   \RequirePackage{tikz}
6650   \RequirePackage{standalone}
6651
6652   \newcommand \tikzinput [2] [] {
6653     \setkeys{Gin}{#1}
6654     \ifx \Gin@ewidth \Gin@exclamation
6655       \ifx \Gin@eheight \Gin@exclamation
6656         \input { #2 }
6657       \else
6658         \resizebox{!}{ \Gin@eheight }{
6659           \input { #2 }
6660         }
6661       \fi
6662     \else
6663       \ifx \Gin@eheight \Gin@exclamation
6664         \resizebox{ \Gin@ewidth }{!}{
```

```

6665         \input { #2 }
6666     }
6667     \else
6668         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6669             \input { #2 }
6670         }
6671     \fi
6672 \fi
6673 }
6674 }
6675
6676 \newcommand \ctikzinput [2] [] {
6677     \begin{center}
6678         \tikzinput [#1] {#2}
6679     \end{center}
6680 }
6681
6682 \@ifpackageloaded{stex}{
6683     \RequirePackage{stex-tikzinput}
6684 }{}
6685
6686 </package>
6687 <*stex>
6688 \ProvidesExplPackage{stex-tikzinput}{2022/05/24}{3.1.0}{stex-tikzinput}
6689 \RequirePackage{stex}
6690 \RequirePackage{tikzinput}
6691
6692 \newcommand\mhtikzinput[2] []{%
6693     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6694     \stex_in_repository:nn\Gin@mhrepos{
6695         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
6696     }
6697 }
6698 \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6699
6700 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6701     \pgfkeys@spdef\pgf@temp{#1}
6702     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6703     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6704     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6705     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6706     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6707     \catcode'\@=11
6708     \catcode'\|=12
6709     \catcode'\$=3
6710     \pgfutil@InputIfFileExists{#2}{-}{-}
6711     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6712     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6713     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6714 }
6715
6716
6717 \newcommand\libusetikzlibrary[1]{

```

```

6718 \prop_if_exist:NF \l_stex_current_repository_prop {
6719   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6720 }
6721 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6722   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6723 }
6724 \seq_clear:N \l__tikzinput_libinput_files_seq
6725 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6726 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6727
6728 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6729   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6730   \IfFileExists{ \l_tmpa_str }{
6731     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6732   }{}
6733   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6734   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6735 }
6736
6737 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6738 \IfFileExists{ \l_tmpa_str }{
6739   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6740 }{}
6741
6742 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6743   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6744 }{
6745   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6746     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6747       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6748     }
6749   }{
6750     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6751   }
6752 }
6753 }
6754 </stex>

```

Chapter 36

document-structure.sty Implementation

```
6755 <*package>
6756 <@@=document_structure>
6757 \ProvidesExplPackage{document-structure}{2022/05/24}{3.1.0}{Modular Document Structure}
6758 \RequirePackage{13keys2e}
```

36.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option xxx will just set the appropriate switches to true (otherwise they stay false).

```
6759
6760 \keys_define:nn{ document-structure }{
6761   class      .str_set_x:N = \c_document_structure_class_str,
6762   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6763   unknown    .code:n      = {
6764     \PassOptionsToClass{\CurrentOption}{stex}
6765     \PassOptionsToClass{\CurrentOption}{tikzinput}
6766   }
6767   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6768 }
6769 \ProcessKeysOptions{ document-structure }
6770 \str_if_empty:NT \c_document_structure_class_str {
6771   \str_set:Nn \c_document_structure_class_str {article}
6772 }
6773 \str_if_empty:NT \c_document_structure_topsect_str {
6774   \str_set:Nn \c_document_structure_topsect_str {section}
6775 }
```

Then we need to set up the packages by requiring the `sref` package to be loaded, and set up triggers for other languages

```
6776 \RequirePackage{xspace}
6777 \RequirePackage{comment}
6778 \RequirePackage{stex}
6779 \AddToHook{begindocument}{}
```



```

6780 \ltx@ifpackageloaded{babel}{
6781   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6782   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6783     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6784   }
6785 }{}
6786 }

```

`\section@level` Finally, we set the `\section@level` macro that governs sectioning. The default is two (corresponding to the `article` class), then we set the defaults for the standard classes `book` and `report` and then we take care of the levels passed in via the `topsect` option.

```

6787 \int_new:N \l_document_structure_section_level_int
6788 \str_case:NnF \c_document_structure_topsect_str {
6789   {part}}{
6790     \int_set:Nn \l_document_structure_section_level_int {0}
6791   }
6792   {chapter}{
6793     \int_set:Nn \l_document_structure_section_level_int {1}
6794   }
6795 }{
6796   \str_case:NnF \c_document_structure_class_str {
6797     {book}{
6798       \int_set:Nn \l_document_structure_section_level_int {0}
6799     }
6800     {report}{
6801       \int_set:Nn \l_document_structure_section_level_int {0}
6802     }
6803   }{
6804     \int_set:Nn \l_document_structure_section_level_int {2}
6805   }
6806 }

```

36.2 Document Structure

The structure of the document is given by the `sfragment` environment. The hierarchy is adjusted automatically according to the L^AT_EX class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.¹⁰

EdN:10

```

6807 \def\current@section@level{document}%
6808 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6809 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

(End definition for `\currentsectionlevel`. This function is documented on page 52.)

`\skipfragment`

```

6810 \cs_new_protected:Npn \skipfragment {

```

¹⁰EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6811 \ifcase\l_document_structure_section_level_int
6812 \or\stepcounter{part}
6813 \or\stepcounter{chapter}
6814 \or\stepcounter{section}
6815 \or\stepcounter{subsection}
6816 \or\stepcounter{subsubsection}
6817 \or\stepcounter{paragraph}
6818 \or\stepcounter{subparagraph}
6819 \fi
6820 }

```

(End definition for `\skipfragment`. This function is documented on page 51.)

`blindfragment (env.)`

```

6821 \newcommand\at@begin@blindsfragment[1]{
6822 \newenvironment{blindfragment}
6823 {
6824 \int_incr:N\l_document_structure_section_level_int
6825 \at@begin@blindsfragment\l_document_structure_section_level_int
6826 }{}

```

`\sfragment@nonum` convenience macro: `\sfragment@nonum{<level>}{<title>}` makes an unnumbered sectioning with title `<title>` at level `<level>`.

```

6827 \newcommand\sfragment@nonum[2]{
6828 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6829 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6830 }

```

(End definition for `\sfragment@nonum`. This function is documented on page ??.)

`\sfragment@num` convenience macro: `\sfragment@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `sfragment` environment and – if it is use it. But how to do that depends on whether the `rdfmata` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```

6831 \newcommand\sfragment@num[2]{
6832 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6833 \@nameuse{#1}{#2}
6834 }{
6835 \cs_if_exist:NTF\rdfmata@sectioning{
6836 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6837 }{
6838 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6839 }
6840 }
6841 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6842 }

```

(End definition for `\sfragment@num`. This function is documented on page ??.)

`sfragment (env.)`

```

6843 \keys_define:nn { document-structure / sfragment }{
6844 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6845 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6846 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6847 contributors   .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6848 srccite        .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6849 type          .tl_set:N     = \l__document_structure_sfragment_type_tl,
6850 short         .tl_set:N     = \l__document_structure_sfragment_short_tl,
6851 intro         .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6852 imports       .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6853 loadmodules    .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6854 }
6855 \cs_new_protected:Nn \__document_structure_sfragment_args:n {
6856   \str_clear:N \l__document_structure_sfragment_id_str
6857   \str_clear:N \l__document_structure_sfragment_date_str
6858   \clist_clear:N \l__document_structure_sfragment_creators_clist
6859   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6860   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6861   \tl_clear:N \l__document_structure_sfragment_type_tl
6862   \tl_clear:N \l__document_structure_sfragment_short_tl
6863   \tl_clear:N \l__document_structure_sfragment_imports_tl
6864   \tl_clear:N \l__document_structure_sfragment_intro_tl
6865   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6866   \keys_set:nn { document-structure / sfragment } { #1 }
6867 }

```

we define a switch for numbering lines and a hook for the beginning of groups: The `\at@begin@sfragment` macro allows customization. It is run at the beginning of the `sfragment`, i.e. after the section heading.

`\at@begin@sfragment`

```

6868 \newif\if@mainmatter\@mainmattertrue
6869 \newcommand\at@begin@sfragment[3][]{ }

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

6870 \keys_define:nn { document-structure / sectioning }{
6871   name      .str_set_x:N = \l__document_structure_sect_name_str  ,
6872   ref       .str_set_x:N = \l__document_structure_sect_ref_str   ,
6873   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6874   clear     .default:n   = {true}                                ,
6875   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
6876   num       .default:n   = {true}
6877 }
6878 \cs_new_protected:Nn \__document_structure_sect_args:n {
6879   \str_clear:N \l__document_structure_sect_name_str
6880   \str_clear:N \l__document_structure_sect_ref_str
6881   \bool_set_false:N \l__document_structure_sect_clear_bool
6882   \bool_set_false:N \l__document_structure_sect_num_bool
6883   \keys_set:nn { document-structure / sectioning } { #1 }
6884 }
6885 \newcommand\omdoc@sectioning[3][]{
6886   \__document_structure_sect_args:n {#1 }
6887   \let\omdoc@sect@name\l__document_structure_sect_name_str
6888   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6889   \if@mainmatter% numbering not overridden by frontmatter, etc.
6890     \bool_if:NTF \l__document_structure_sect_num_bool {
6891       \sfragment@num{#2}{#3}
6892     }{

```

```

6893     \sfragment@nonum{#2}{#3}
6894   }
6895   \def\current@section@level{\omdoc@sect@name}
6896   \else
6897     \sfragment@nonum{#2}{#3}
6898   \fi
6899 }% if@mainmatter

```

and another one, if redefines the `\addtocontentsline` macro of L^AT_EX to import the respective macros. It takes as an argument a list of module names.

```

6900 \newcommand\sfragment@redefine@addtocontents[1]{%
6901 %\edef\__document_structureimport{#1}%
6902 %\@for\@I:=\__document_structureimport\do{%
6903 %\edef\@path{\csname module@\@I @path\endcsname}%
6904 %\@ifundefined{tf@toc}\relax%
6905 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6906 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6907 %\def\addcontentsline##1##2##3{%
6908 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
6909 %\else% hyperref.sty not loaded
6910 %\def\addcontentsline##1##2##3{%
6911 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
6912 %\fi
6913 }% hypreref.sty loaded?

```

now the `sfragment` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registers the current level of sfragments in the `\sfragment@level` counter.

```

6914 \newenvironment{sfragment}[2][ ]% keys, title
6915 {
6916   \__document_structure_sfragment_args:n { #1 }%\sref@target%

```

If the `loadmodules` key is set on `\begin{sfragment}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

6917   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6918
6919   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6920     \sfragment@redefine@addtocontents{
6921       %\@ifundefined{module@id}\used@modules%
6922       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6923     }
6924   }

```

now we only need to construct the right sectioning depending on the value of `\section@level`.

```

6925
6926   \stex_document_title:n { #2 }
6927
6928   \int_incr:N\l__document_structure_section_level_int
6929   \ifcase\l__document_structure_section_level_int
6930     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6931     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6932     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6933     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

6934 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6935 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
6936 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{paragraph}{#1}
6937 \fi
6938 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
6939 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6940   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6941 }
6942 }% for customization
6943 {}

```

and finally, we localize the sections

```

6944 \newcommand\omdoc@part@kw{Part}
6945 \newcommand\omdoc@chapter@kw{Chapter}
6946 \newcommand\omdoc@section@kw{Section}
6947 \newcommand\omdoc@subsection@kw{Subsection}
6948 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6949 \newcommand\omdoc@paragraph@kw{paragraph}
6950 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

36.3 Front and Backmatter

Index markup is provided by the `omtext` package [Kohlhase:smmtf:git], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

6951 \providecommand\printindex{\IfFileExists{\jobname.ind}{\input{\jobname.ind}}{}}

```

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

some classes (e.g. `book.cls`) already have `\frontmatter`, `\mainmatter`, and `\backmatter` macros. As we want to define `frontmatter` and `backmatter` environments, we save their behavior (possibly defining it) in `orig@*matter` macros and make them undefined (so that we can define the environments).

```

6952 \cs_if_exist:NTF\frontmatter{
6953   \let\__document_structure_orig_frontmatter\frontmatter
6954   \let\frontmatter\relax
6955 }{
6956   \tl_set:Nn\__document_structure_orig_frontmatter{
6957     \clearpage
6958     \@mainmatterfalse
6959     \pagenumbering{roman}
6960   }
6961 }
6962 \cs_if_exist:NTF\backmatter{
6963   \let\__document_structure_orig_backmatter\backmatter
6964   \let\backmatter\relax
6965 }{
6966   \tl_set:Nn\__document_structure_orig_backmatter{
6967     \clearpage
6968     \@mainmatterfalse
6969     \pagenumbering{roman}
6970   }

```

```
6971 }
```

Using these, we can now define the `frontmatter` and `backmatter` environments

`frontmatter (env.)` we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, otherwise we define it.

```
6972 \newenvironment{frontmatter}{
6973   \_document_structure_orig_frontmatter
6974 }{
6975   \cs_if_exist:NTF\mainmatter{
6976     \mainmatter
6977   }{
6978     \clearpage
6979     \@mainmattertrue
6980     \pagenumbering{arabic}
6981   }
6982 }
```

`backmatter (env.)` As `backmatter` is at the end of the document, we do nothing for `\endbackmatter`.

```
6983 \newenvironment{backmatter}{
6984   \_document_structure_orig_backmatter
6985 }{
6986   \cs_if_exist:NTF\mainmatter{
6987     \mainmatter
6988   }{
6989     \clearpage
6990     \@mainmattertrue
6991     \pagenumbering{arabic}
6992   }
6993 }
```

finally, we make sure that page numbering is arabic and we have main matter as the default

```
6994 \@mainmattertrue\pagenumbering{arabic}
```

`\prematurestop` We initialize `\afterprematurestop`, and provide `\prematurestop@endsfragment` which looks up `\sfragment@level` and recursively ends enough `{sfragment}`s.

```
6995 \def \c__document_structure_document_str{document}
6996 \newcommand\afterprematurestop{}
6997 \def\prematurestop@endsfragment{
6998   \unless\ifx\@currenvir\c__document_structure_document_str
6999     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7000     \expandafter\prematurestop@endsfragment
7001   \fi
7002 }
7003 \providecommand\prematurestop{
7004   \message{Stopping~sTeX~processing~prematurely}
7005   \prematurestop@endsfragment
7006   \afterprematurestop
7007   \end{document}
7008 }
```

(End definition for `\prematurestop`. This function is documented on page 52.)

36.4 Global Variables

\setSGvar set a global variable

```
7009 \RequirePackage{etoolbox}
7010 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

(End definition for \setSGvar. This function is documented on page 52.)

\useSGvar use a global variable

```
7011 \newrobustcmd\useSGvar[1]{%
7012   \@ifundefined{sTeX@Gvar@#1}
7013   {\PackageError{document-structure}
7014     {The sTeX Global variable #1 is undefined}
7015     {set it with \protect\setSGvar}}
7016   \@nameuse{sTeX@Gvar@#1}}
```

(End definition for \useSGvar. This function is documented on page 52.)

\ifSGvar execute something conditionally based on the state of the global variable.

```
7017 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7018   \@ifundefined{sTeX@Gvar@#1}
7019   {\PackageError{document-structure}
7020     {The sTeX Global variable #1 is undefined}
7021     {set it with \protect\setSGvar}}
7022   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

(End definition for \ifSGvar. This function is documented on page 52.)

Chapter 37

NotesSlides – Implementation

37.1 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.

```
7023 \*cls)
7024 \@@=notesslides)
7025 \ProvidesExplClass{notesslides}{2022/05/24}{3.1.0}{notesslides Class}
7026 \RequirePackage{13keys2e}
7027
7028 \keys_define:nn{notesslides / cls}{
7029   class .str_set_x:N = \c__notesslides_class_str,
7030   notes .bool_set:N = \c__notesslides_notes_bool ,
7031   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7032   docopt .str_set_x:N = \c__notesslides_docopt_str,
7033   unknown .code:n = {
7034     \PassOptionsToPackage{\CurrentOption}{document-structure}
7035     \PassOptionsToClass{\CurrentOption}{beamer}
7036     \PassOptionsToPackage{\CurrentOption}{notesslides}
7037     \PassOptionsToPackage{\CurrentOption}{stex}
7038   }
7039 }
7040 \ProcessKeysOptions{ notesslides / cls }
7041
7042 \str_if_empty:NF \c__notesslides_class_str {
7043   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7044 }
7045
7046 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7047   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7048 }
7049 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7050   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7051 }
7052
7053 \RequirePackage{stex}
```



```

7054 \stex_html_backend:T {
7055   \bool_set_true:N\c__notesslides_notes_bool
7056 }
7057
7058 \bool_if:NTF \c__notesslides_notes_bool {
7059   \PassOptionsToPackage{notes=true}{notesslides}
7060   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7061 }{
7062   \PassOptionsToPackage{notes=false}{notesslides}
7063   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7064 }
7065 </cls>

```

now we do the same for the notesslides package.

```

7066 <*package>
7067 \ProvidesExplPackage{notesslides}{2022/05/24}{3.1.0}{notesslides Package}
7068 \RequirePackage{l3keys2e}
7069
7070 \keys_define:nn{notesslides / pkg}{
7071   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7072   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7073   notes            .bool_set:N = \c__notesslides_notes_bool ,
7074   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7075   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7076   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7077   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7078   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
7079   unknown          .code:n      = {
7080     \PassOptionsToClass{\CurrentOption}{stex}
7081     \PassOptionsToClass{\CurrentOption}{tikzinput}
7082   }
7083 }
7084 \ProcessKeysOptions{ notesslides / pkg }
7085
7086 \RequirePackage{stex}
7087 \stex_html_backend:T {
7088   \bool_set_true:N\c__notesslides_notes_bool
7089 }
7090
7091 \newif\ifnotes
7092 \bool_if:NTF \c__notesslides_notes_bool {
7093   \notesttrue
7094 }{
7095   \notesfalse
7096 }
7097

```

we give ourselves a macro \@@topsect that needs only be evaluated once, so that the \ifdefstring conditionals work below.

```

7098 \str_if_empty:NTF \c__notesslides_topsect_str {
7099   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7100 }{
7101   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7102 }
7103 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
7104 </package>
```

Depending on the options, we either load the `article`-based document-structure or the `beamer` class (and set some counters).

```
7105 <*cls>
7106 \bool_if:NTF \c__notesslides_notes_bool {
7107   \str_if_empty:NT \c__notesslides_class_str {
7108     \str_set:Nn \c__notesslides_class_str {article}
7109   }
7110   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
7111     {\c__notesslides_class_str}
7112 }{
7113   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7114   \newcounter{Item}
7115   \newcounter{paragraph}
7116   \newcounter{subparagraph}
7117   \newcounter{Hfootnote}
7118 }
7119 \RequirePackage{document-structure}
```

now it only remains to load the `notesslides` package that does all the rest.

```
7120 \RequirePackage{notesslides}
7121 </cls>
```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the \TeX packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the \TeX -specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```
7122 <*package>
7123 \bool_if:NT \c__notesslides_notes_bool {
7124   \RequirePackage{a4wide}
7125   \RequirePackage{marginnote}
7126   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7127   \RequirePackage{mdframed}
7128   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7129   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7130 }
7131 \RequirePackage{stex-tikzinput}
7132 \RequirePackage{comment}
7133 \RequirePackage{url}
7134 \RequirePackage{graphicx}
7135 \RequirePackage{pgf}
7136 \RequirePackage{bookmark}
```

37.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class.

```
7137 \bool_if:NT \c__notesslides_notes_bool {
7138   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7139 }
```

```

7140 \NewDocumentCommand \libusetheme {0{} m} {
7141   \libusepackage[#1]{beamertheme#2}
7142 }
7143

```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```

7144 \newcounter{slide}
7145 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7146 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

note (*env.*) 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 as a comment via the **comment** package.

```

7147 \bool_if:NTF \c__notesslides_notes_bool {
7148   \renewenvironment{note}{\ignorespaces}{}
7149 }{
7150   \excludecomment{note}
7151 }

```

We first set up the slide boxes in **article** mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```

7152 \bool_if:NT \c__notesslides_notes_bool {
7153   \newlength{\slideframewidth}
7154   \setlength{\slideframewidth}{1.5pt}

```

frame (*env.*) We first define the keys.

```

7155 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7156   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7157     \bool_set_true:N #1
7158   }{
7159     \bool_set_false:N #1
7160   }
7161 }
7162 \keys_define:nn{notesslides / frame}{
7163   label .str_set_x:N = \l__notesslides_frame_label_str,
7164   allowframebreaks .code:n = {
7165     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7166   },
7167   allowdisplaybreaks .code:n = {
7168     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7169   },
7170   fragile .code:n = {
7171     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7172   },
7173   shrink .code:n = {
7174     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7175   },
7176   squeeze .code:n = {
7177     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7178   },
7179   t .code:n = {

```

```

7180     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7181   },
7182   unknown    .code:n      = {}
7183 }
7184 \cs_new_protected:Nn \__notesslides_frame_args:n {
7185   \str_clear:N \l__notesslides_frame_label_str
7186   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7187   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7188   \bool_set_true:N \l__notesslides_frame_fragile_bool
7189   \bool_set_true:N \l__notesslides_frame_shrink_bool
7190   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7191   \bool_set_true:N \l__notesslides_frame_t_bool
7192   \keys_set:nn { notesslides / frame }{ #1 }
7193 }

```

We define the environment, read them, and construct the slide number and label.

```

7194 \renewenvironment{frame}[1][]{
7195   \__notesslides_frame_args:n{#1}
7196   \sffamily
7197   \stepcounter{slide}
7198   \def\@currentlabel{\theslide}
7199   \str_if_empty:NF \l__notesslides_frame_label_str {
7200     \label{\l__notesslides_frame_label_str}
7201   }

```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```

7202   \def\itemize@level{outer}
7203   \def\itemize@outer{outer}
7204   \def\itemize@inner{inner}
7205   \renewcommand\newpage{\addtocounter{framenum}{1}}
7206   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7207   \renewenvironment{itemize}{
7208     \ifx\itemize@level\itemize@outer
7209       \def\itemize@label{$\rhd$}
7210     \fi
7211     \ifx\itemize@level\itemize@inner
7212       \def\itemize@label{$\scriptstyle\rhd$}
7213     \fi
7214     \begin{list}
7215       {\itemize@label}
7216       {\setlength{\labelsep}{.3em}
7217        \setlength{\labelwidth}{.5em}
7218        \setlength{\leftmargin}{1.5em}
7219       }
7220     \edef\itemize@level{\itemize@inner}
7221   }{
7222     \end{list}
7223   }

```

We create the box with the `mdframed` environment from the `equinymous` package.

```

7224   \stex_html_backend:TF {
7225     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7226     \mdf@patchamsthm
7227   }{
7228     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7229     }
7230   }{
7231     \stex_html_backend:TF {
7232       \miko@slidelabel\egroup\end{stex_annotate_env}
7233     }\medskip\miko@slidelabel\end{mdframed}}
7234   }

```

Now, we need to redefine the frametitle (we are still in course notes mode).

`\frametitle`

```

7235   \renewcommand{\frametitle}[1]{
7236     \stex_document_title:n { #1 }
7237     {\Large\bf\sf\color{blue}{#1}}\medskip
7238   }
7239 }

```

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

EdN:11

`\pause` 11

```

7240 \bool_if:NT \c__notesslides_notes_bool {
7241   \newcommand\pause{ }
7242 }

```

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

`nparagraph (env.)`

```

7243 \bool_if:NTF \c__notesslides_notes_bool {
7244   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7245 }{
7246   \excludecomment{nparagraph}
7247 }

```

`nfragment (env.)`

```

7248 \bool_if:NTF \c__notesslides_notes_bool {
7249   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1][#2]}\end{sfragment}}
7250 }{
7251   \excludecomment{nfragment}
7252 }

```

`ndefinition (env.)`

```

7253 \bool_if:NTF \c__notesslides_notes_bool {
7254   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7255 }{
7256   \excludecomment{ndefinition}
7257 }

```

`nassertion (env.)`

```

7258 \bool_if:NTF \c__notesslides_notes_bool {
7259   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7260 }{
7261   \excludecomment{nassertion}
7262 }

```

¹¹EDNOTE: MK: fake it in notes mode for now

`nsproof (env.)`

```
7263 \bool_if:NTF \c__notesslides_notes_bool {
7264   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7265 }{
7266   \excludecomment{nproof}
7267 }
```

`nexample (env.)`

```
7268 \bool_if:NTF \c__notesslides_notes_bool {
7269   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7270 }{
7271   \excludecomment{nexample}
7272 }
```

`\inputref@*skip` We customize the hooks for in `\inputref`.

```
7273 \def\inputref@preskip{\smallskip}
7274 \def\inputref@postskip{\medskip}
```

*(End definition for \inputref@*skip. This function is documented on page ??.)*

`\inputref*`

```
7275 \let\orig@inputref\inputref
7276 \def\inputref{@ifstar\ninputref\orig@inputref}
7277 \newcommand\ninputref[2] []{
7278   \bool_if:NT \c__notesslides_notes_bool {
7279     \orig@inputref[#1]{#2}
7280   }
7281 }
```

(End definition for \inputref. This function is documented on page 54.)*

37.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

`\setslidelogo` The default logo is the `STEX` logo. Customization can be done by `\setslidelogo{<logo name>}`.

```
7282 \newlength{\slidelogoheight}
7283
7284 \RequirePackage{graphicx}
7285
7286 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7287 \providecommand\mhgraphics[2] []{
7288   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7289   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7290 }
7291
7292 \bool_if:NTF \c__notesslides_notes_bool {
7293   \setlength{\slidelogoheight}{.4cm}
7294 }{
7295   \setlength{\slidelogoheight}{.25cm}
7296 }
```

```

7297 \ifcsname slidelogo\endcsname\else
7298   \newsavebox{\slidelogo}
7299   \sbox{\slidelogo}{\sTeX}
7300 \fi
7301 \newrobustcmd{\setslidelogo}[2][]{
7302   \tl_if_empty:nTF{#1}{
7303     \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7304   }{
7305     \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7306   }
7307 }

```

(End definition for `\setslidelogo`. This function is documented on page 55.)

\author In notes mode, we redefine the `\author` macro so that it does not disregard the optional argument (as `beamerarticle` does). We want to use it to set the source later.

```

7308 \bool_if:NT \c__notesslides_notes_bool {
7309   \def\author{\@dblarg\@ns@author}
7310   \long\def\@ns@author[#1]#2{%
7311     \def\c__notesslides_shortauthor{#1}%
7312     \def\@author{#2}
7313   }
7314 }

```

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

\setsource `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name.

```

7315 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

(End definition for `\setsource`. This function is documented on page 55.)

\setlicensing Now, we set up the copyright and licensing. By default we use the Creative Commons Attribution-ShareAlike license 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.

```

7316 \def\copyrightnotice{%
7317   \footnotesize\copyright : \hspace{.3ex}%
7318   \ifcsname source\endcsname\source\else%
7319   \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7320   \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7321   ?source/author?\fi%
7322 \fi}
7323 \newsavebox{\cclogo}
7324 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7325 \newif\ifcchref\cchreffalse
7326 \AtBeginDocument{
7327   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7328 }
7329 \def\licensing{
7330   \ifcchref
7331     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7332   \else
7333     {\usebox{\cclogo}}

```

```

7334 \fi
7335 }
7336 \newrobustcmd{\setlicensing}[2][]{
7337   \def\@url{#1}
7338   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7339   \ifx\@url\@empty
7340     \def\licensing{\usebox{\cclogo}}
7341   \else
7342     \def\licensing{
7343       \ifcchref
7344         \href{#1}{\usebox{\cclogo}}
7345       \else
7346         {\usebox{\cclogo}}
7347     \fi
7348   }
7349 \fi
7350 }

```

(End definition for \setlicensing. This function is documented on page 55.)

EdN:12

```

\slidelabel Now, we set up the slide label for the article mode.12
7351 \newrobustcmd\miko@slidelabel{
7352   \vbox to \slidelogoheight{
7353     \vss\hbox to \slidewidth
7354       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7355   }
7356 }

```

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

37.4 Frame Images

\frameimage We have to make sure that the width is overwritten, for that we check the \Gin@ewidth macro from the graphicx package. We also add the label key.

```

7357 \def\Gin@mhrepos{}
7358 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7359 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7360 \newrobustcmd\frameimage[2][]{
7361   \stepcounter{slide}
7362   \bool_if:NT \c__notesslides_frameimages_bool {
7363     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7364     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7365     \begin{center}
7366       \bool_if:NTF \c__notesslides_fiboxed_bool {
7367         \fbox{
7368           \ifx\Gin@ewidth\@empty
7369             \ifx\Gin@mhrepos\@empty
7370               \mhgraphics[width=\slidewidth,#1]{#2}
7371             \else
7372               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7373             \fi
7374           \else% Gin@ewidth empty

```

¹²EdNOTE: see that we can use the themes for the slides some day. This is all fake.


```

7375         \ifx\Gin@mhrepos\@empty
7376         \mhgraphics[#1]{#2}
7377     \else
7378         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7379     \fi
7380 \fi% Gin@ewidth empty
7381 }
7382 }{
7383     \ifx\Gin@ewidth\@empty
7384     \ifx\Gin@mhrepos\@empty
7385         \mhgraphics[width=\slidewidth,#1]{#2}
7386     \else
7387         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7388     \fi
7389     \ifx\Gin@mhrepos\@empty
7390         \mhgraphics[#1]{#2}
7391     \else
7392         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7393     \fi
7394     \fi% Gin@ewidth empty
7395 }
7396 \end{center}
7397 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7398 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7399 }
7400 } % ifmks@sty@frameimages

```

(End definition for `\frameimage`. This function is documented on page 55.)

37.5 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```

7401 \stex_html_backend:F {
7402     \bool_if:NT \c__notesslides_sectocframes_bool {
7403         \str_if_eq:VnTF \__notesslidestopsect{part}{
7404             \newcounter{chapter}\counterwithin*{section}{chapter}
7405         }{
7406             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7407                 \newcounter{chapter}\counterwithin*{section}{chapter}
7408             }
7409         }
7410     }
7411 }

```

`\section@level` We set the `\section@level` counter that governs sectioning according to the class options. We also introduce the sectioning counters accordingly.

`\section@level`

```

7412 \def\part@prefix{}
7413 \@ifpackageloaded{document-structure}{
7414     \str_case:VnF \__notesslidestopsect {

```

```

7415 {part}{
7416   \int_set:Nn \l_document_structure_section_level_int {0}
7417   \def\thesection{\arabic{chapter}.\arabic{section}}
7418   \def\part@prefix{\arabic{chapter}.}
7419 }
7420 {chapter}{
7421   \int_set:Nn \l_document_structure_section_level_int {1}
7422   \def\thesection{\arabic{chapter}.\arabic{section}}
7423   \def\part@prefix{\arabic{chapter}.}
7424 }
7425 }{
7426   \int_set:Nn \l_document_structure_section_level_int {2}
7427   \def\part@prefix{}
7428 }
7429 }
7430
7431 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

(End definition for \section@level. This function is documented on page ??.)

The new counters are used in the `sfragment` environment that chooses the L^AT_EX sectioning macros according to `\section@level`.

`sfragment (env.)`

```

7432 \renewenvironment{sfragment}[2][]{
7433   \__document_structure_sfragment_args:n { #1 }
7434   \int_incr:N \l_document_structure_section_level_int
7435   \bool_if:NT \c__notesslides_sectocframes_bool {
7436     \stepcounter{slide}
7437     \begin{frame}[noframenumbering]
7438     \vfill\Large\centering
7439     \red{
7440       \ifcase\l_document_structure_section_level_int\or
7441         \stepcounter{part}
7442         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7443         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7444         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7445         \def\currentsectionlevel{\omdoc@part@kw}
7446       \or
7447         \stepcounter{chapter}
7448         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7449         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7450         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7451         \def\currentsectionlevel{\omdoc@chapter@kw}
7452       \or
7453         \stepcounter{section}
7454         \def\__notesslideslabel{\part@prefix\arabic{section}}
7455         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7456         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7457         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7458         \def\currentsectionlevel{\omdoc@section@kw}
7459       \or
7460         \stepcounter{subsection}
7461         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7462         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

7463         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7464         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7465         \def\currentsectionlevel{\omdoc@subsection@kw}
7466     \or
7467         \stepcounter{subsubsection}
7468         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7469         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7470         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7471         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7472         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7473     \or
7474         \stepcounter{paragraph}
7475         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7476         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7477         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7478         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7479         \def\currentsectionlevel{\omdoc@paragraph@kw}
7480     \else
7481         \def\__notesslideslabel{}
7482         \def\currentsectionlevel{\omdoc@paragraph@kw}
7483     \fi% end ifcase
7484     \__notesslideslabel\quad #2%
7485 }%
7486 \vfill%
7487 \end{frame}%
7488 }
7489 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7490     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7491 }
7492 }{}
7493 }

```

We set up a beamer template for theorems like ams style, but without a block environment.

```

7494 \def\inserttheorembodyfont{\normalfont}
7495 %\bool_if:NF \c__notesslides_notes_bool {
7496 % \defbeamertemplate{theorem begin}{miko}
7497 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7498 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7499 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7500 % \defbeamertemplate{theorem end}{miko}{\fi}

```

and we set it as the default one.

```

7501 % \setbeamertemplate{theorems}[miko]

```

The following fixes an error I do not understand, this has something to do with beamer compatibility, which has similar definitions but only up to 1.

```

7502 % \expandafter\def\csname Parent2\endcsname{}
7503 %}
7504
7505 \AddToHook{begindocument}{ % this does not work for some reason
7506     \setbeamertemplate{theorems}[ams style]
7507 }
7508 \bool_if:NT \c__notesslides_notes_bool {
7509     \renewenvironment{columns}[1][\fi]{}{}

```

```

7510     \par\noindent%
7511     \begin{minipage}%
7512     \slidewidth\centering\leavevmode%
7513   }{%
7514     \end{minipage}\par\noindent%
7515   }%
7516   \newsavebox\columnbox%
7517   \renewenvironment<>{column}[2][]{%
7518     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7519   }{%
7520     \end{minipage}\end{lrbox}\usebox\columnbox%
7521   }%
7522 }

7523 \bool_if:NTF \c__notesslides_noproblems_bool {
7524   \newenvironment{problems}{}{}
7525 }{
7526   \excludacomment{problems}
7527 }

```

37.6 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.

```

7528 \gdef\printexcursions{}
7529 \newcommand\excursionref[2]{% label, text
7530   \bool_if:NT \c__notesslides_notes_bool {
7531     \begin{sparagraph}[title=Excursion]
7532       #2 \sref[fallback=the appendix]{#1}.
7533     \end{sparagraph}
7534   }
7535 }
7536 \newcommand\activate@excursion[2][{}{
7537   \gappto\printexcursions{\inputref{#1}{#2}}
7538 }
7539 \newcommand\excursion[4][{}{ repos, label, path, text
7540   \bool_if:NT \c__notesslides_notes_bool {
7541     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7542   }
7543 }

```

(End definition for `\excursion`. This function is documented on page 56.)

\excursiongroup

```

7544 \keys_define:nn{notesslides / excursiongroup }{
7545   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7546   intro       .tl_set:N    = \l__notesslides_excursion_intro_tl,
7547   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7548 }
7549 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7550   \tl_clear:N \l__notesslides_excursion_intro_tl
7551   \str_clear:N \l__notesslides_excursion_id_str

```

```

7552 \str_clear:N \l__notesslides_excursion_mhrepos_str
7553 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7554 }
7555 \newcommand\excursionsgroup[1][]{
7556   \__notesslides_excursion_args:n{ #1 }
7557   \ifdefempty\printexcursions{}% only if there are excursions
7558   {\begin{note}
7559     \begin{sfragment}[#1]{Excursions}%
7560     \ifdefempty\l__notesslides_excursion_intro_tl}{
7561       \inputref[\l__notesslides_excursion_mhrepos_str]{
7562         \l__notesslides_excursion_intro_tl
7563       }
7564     }
7565     \printexcursions%
7566     \end{sfragment}
7567   \end{note}}
7568 }
7569 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7570 \end{package}

```

(End definition for \excursionsgroup. This function is documented on page 56.)

Chapter 38

The Implementation

38.1 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.

```
7571 <*package>
7572 <@@=problems>
7573 \ProvidesExplPackage{problem}{2022/05/24}{3.1.0}{Semantic Markup for Problems}
7574 \RequirePackage{l3keys2e,stex}
7575 \RequirePackage{amssymb}% for \Box
7576
7577 \keys_define:nn { problem / pkg }{
7578   notes      .default:n    = { true },
7579   notes      .bool_set:N   = \c__problems_notes_bool,
7580   gnotes     .default:n    = { true },
7581   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7582   hints      .default:n    = { true },
7583   hints      .bool_set:N   = \c__problems_hints_bool,
7584   solutions  .default:n    = { true },
7585   solutions  .bool_set:N   = \c__problems_solutions_bool,
7586   pts        .default:n    = { true },
7587   pts        .bool_set:N   = \c__problems_pts_bool,
7588   min        .default:n    = { true },
7589   min        .bool_set:N   = \c__problems_min_bool,
7590   boxed      .default:n    = { true },
7591   boxed      .bool_set:N   = \c__problems_boxed_bool,
7592   unknown    .code:n       = {}
7593 }
7594 \newif\ifsolutions
7595
7596 \ProcessKeysOptions{ problem / pkg }
7597 \bool_if:NTF \c__problems_solutions_bool {
7598   \solutionstrue
7599 }{
7600   \solutionsfalse
7601 }
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7602 \RequirePackage{comment}
```

The next package relies on the L^AT_EX3 kernel, which L^AT_EXML only partially supports. As it is purely presentational, we only load it when the `boxed` option is given and we run L^AT_EXML.

```
7603 \bool_if:NT \c__problems_boxed_bool { \RequirePackage{mdframed} }
```

`\prob@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
7604 \def\prob@problem@kw{Problem}
7605 \def\prob@solution@kw{Solution}
7606 \def\prob@hint@kw{Hint}
7607 \def\prob@note@kw{Note}
7608 \def\prob@grade@kw{Grading}
7609 \def\prob@pt@kw{pt}
7610 \def\prob@min@kw{min}
7611 \def\prob@correct@kw{Correct}
7612 \def\prob@wrong@kw{Wrong}
```

(End definition for `\prob@*@kw`. This function is documented on page ??.)

For the other languages, we set up triggers

```
7613 \AddToHook{begindocument}{
7614   \ltx@ifpackageloaded{babel}{
7615     \makeatletter
7616     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7617     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7618       \input{problem-ngerman.ldf}
7619     }
7620     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7621       \input{problem-finnish.ldf}
7622     }
7623     \clist_if_in:NnT \l_tmpa_clist {french}{
7624       \input{problem-french.ldf}
7625     }
7626     \clist_if_in:NnT \l_tmpa_clist {russian}{
7627       \input{problem-russian.ldf}
7628     }
7629     \makeatother
7630   }{}
7631 }
```

38.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
7632 \keys_define:nn{ problem / problem }{
7633   id      .str_set:x:N = \l__problems_prob_id_str,
7634   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7635   min     .tl_set:N    = \l__problems_prob_min_tl,
7636   title   .tl_set:N    = \l__problems_prob_title_tl,
7637   type    .tl_set:N    = \l__problems_prob_type_tl,
```

```

7638 imports .tl_set:N      = \l__problems_prob_imports_tl,
7639 name      .str_set_x:N   = \l__problems_prob_name_str,
7640 refnum    .int_set:N     = \l__problems_prob_refnum_int
7641 }
7642 \cs_new_protected:Nn \__problems_prob_args:n {
7643   \str_clear:N \l__problems_prob_id_str
7644   \str_clear:N \l__problems_prob_name_str
7645   \tl_clear:N \l__problems_prob_pts_tl
7646   \tl_clear:N \l__problems_prob_min_tl
7647   \tl_clear:N \l__problems_prob_title_tl
7648   \tl_clear:N \l__problems_prob_type_tl
7649   \tl_clear:N \l__problems_prob_imports_tl
7650   \int_zero_new:N \l__problems_prob_refnum_int
7651   \keys_set:nn { problem / problem }{ #1 }
7652   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7653     \let\l__problems_prob_refnum_int\undefined
7654   }
7655 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7656 \newcounter{problem}[section]
7657 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}

```

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

`\prob@label` We provide the macro `\prob@label` to redefine later to get context involved.

```

7658 \newcommand\prob@label[1]{\thesection.#1}

```

(End definition for `\prob@label`. This function is documented on page ??.)

`\prob@number` We consolidate the problem number into a reusable internal macro

```

7659 \newcommand\prob@number{
7660   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7661     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7662   }{
7663     \int_if_exist:NTF \l__problems_prob_refnum_int {
7664       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7665     }{
7666       \prob@label\theproblem
7667     }
7668   }
7669 }

```

(End definition for `\prob@number`. This function is documented on page ??.)

`\prob@title` We consolidate the problem title into a reusable internal macro as well. `\prob@title` takes three arguments the first is the fallback when no title is given at all, the second and third go around the title, if one is given.

```

7670 \newcommand\prob@title[3]{%
7671   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7672     #2 \l__problems_inclprob_title_tl #3
7673   }{
7674     \tl_if_exist:NTF \l__problems_prob_title_tl {

```



```

7675     #2 \l__problems_prob_title_tl #3
7676   }{
7677     #1
7678   }
7679 }
7680 }

```

(End definition for `\prob@title`. This function is documented on page ??.)

With these the problem header is a one-liner

`\prob@heading` We consolidate the problem header line into a separate internal macro that can be reused in various settings.

```

7681 \def\prob@heading{
7682   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7683   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7684 }

```

(End definition for `\prob@heading`. This function is documented on page ??.)

With this in place, we can now define the `problem` environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whether the respective options are set.

`sproblem (env.)`

```

7685 \newenvironment{sproblem}[1][]{
7686   \__problems_prob_args:n{#1}%\sref@target%
7687   \@in@omtexttrue% we are in a statement (for inline definitions)
7688   \stepcounter{problem}\record@problem
7689   \def\current@section@level{\prob@problem@kw}
7690
7691   \str_if_empty:NT \l__problems_prob_name_str {
7692     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7693     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7694     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7695   }
7696
7697   \stex_if_do_html:T{
7698     \tl_if_empty:NF \l__problems_prob_title_tl {
7699       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7700     }
7701   }
7702
7703   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7704
7705   \stex_reactivate_macro:N \STEXexport
7706   \stex_reactivate_macro:N \importmodule
7707   \stex_reactivate_macro:N \symdecl
7708   \stex_reactivate_macro:N \notation
7709   \stex_reactivate_macro:N \symdef
7710
7711   \stex_if_do_html:T{
7712     \begin{sproblem} {problem} {
7713       \l_stex_module_ns_str ? \l_stex_module_name_str
7714     }

```

```

7715
7716 \stex_annotate_invisible:nnn{header}{} {
7717   \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
7718   \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7719   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7720     \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7721   }
7722 }
7723 }
7724
7725 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7726
7727
7728 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7729   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7730 }{
7731   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7732 }
7733 \str_if_exist:NTF \l__problems_inclprob_id_str {
7734   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7735 }{
7736   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7737 }
7738
7739
7740 \stex_if_smsmode:F {
7741   \clist_set:No \l_tmpa_clist \sproblemtype
7742   \tl_clear:N \l_tmpa_tl
7743   \clist_map_inline:Nn \l_tmpa_clist {
7744     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7745       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7746     }
7747   }
7748   \tl_if_empty:NTF \l_tmpa_tl {
7749     \__problems_sproblem_start:
7750   }{
7751     \l_tmpa_tl
7752   }
7753 }
7754 \stex_ref_new_doc_target:n \sproblemid
7755 \stex_smsmode_do:
7756 }{
7757   \__stex_modules_end_module:
7758   \stex_if_smsmode:F{
7759     \clist_set:No \l_tmpa_clist \sproblemtype
7760     \tl_clear:N \l_tmpa_tl
7761     \clist_map_inline:Nn \l_tmpa_clist {
7762       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7763         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7764       }
7765     }
7766     \tl_if_empty:NTF \l_tmpa_tl {
7767       \__problems_sproblem_end:
7768     }{

```

```

7769     \l_tmpa_tl
7770   }
7771 }
7772 \stex_if_do_html:T{
7773   \end{stex_annotate_env}
7774 }
7775
7776 \smallskip
7777 }
7778
7779 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7780
7781
7782
7783 \cs_new_protected:Nn \__problems_sproblem_start: {
7784   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7785 }
7786 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7787
7788 \newcommand\stexpatchproblem[3][] {
7789   \str_set:Nx \l_tmpa_str{ #1 }
7790   \str_if_empty:NTF \l_tmpa_str {
7791     \tl_set:Nn \__problems_sproblem_start: { #2 }
7792     \tl_set:Nn \__problems_sproblem_end: { #3 }
7793   }{
7794     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7795     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7796   }
7797 }
7798
7799
7800 \bool_if:NT \c__problems_boxed_bool {
7801   \surroundwithhmdframed{problem}
7802 }

```

\record@problem This macro records information about the problems in the *.aux file.

```

7803 \def\record@problem{
7804   \protected@write\@auxout{}
7805   {
7806     \string\@problem{\prob@number}
7807     {
7808       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7809         \l__problems_inclprob_pts_tl
7810       }{
7811         \l__problems_prob_pts_tl
7812       }
7813     }%
7814     {
7815       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7816         \l__problems_inclprob_min_tl
7817       }{
7818         \l__problems_prob_min_tl
7819       }
7820     }

```

```

7821 }
7822 }

```

(End definition for \record@problem. This function is documented on page ??.)

\@problem This macro acts on a problem's record in the *.aux file. It does not have any functionality here, but can be redefined elsewhere (e.g. in the assignment package).

```

7823 \def\@problem#1#2#3{}

```

(End definition for \@problem. This function is documented on page ??.)

solution (env.) The solution environment is similar to the problem environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```

7824 \keys_define:nn { problem / solution }{
7825   id          .str_set_x:N = \l__problems_solution_id_str ,
7826   for         .str_set_x:N = \l__problems_solution_for_str ,
7827   type        .str_set_x:N = \l__problems_solution_type_str ,
7828   title       .tl_set:N    = \l__problems_solution_title_tl
7829 }
7830 \cs_new_protected:Nn \__problems_solution_args:n {
7831   \str_clear:N \l__problems_solution_id_str
7832   \str_clear:N \l__problems_solution_type_str
7833   \str_clear:N \l__problems_solution_for_str
7834   \tl_clear:N \l__problems_solution_title_tl
7835   \keys_set:nn { problem / solution }{ #1 }
7836 }

```

\startsolutions for the \startsolutions macro we use the \specialcomment macro from the comment package. Note that we use the \@startsolution macro in the start codes, that parses the optional argument.

```

7837 \box_new:N \l__problems_solution_box
7838 \newenvironment{solution}[1][{}{
7839   \__problems_solution_args:n{#1}
7840   \stex_html_backend:TF{
7841     \stex_if_do_html:T{
7842       \begin{stex_annotate_env}{solution}{}
7843       \str_if_empty:NF \l__problems_solution_type_str {
7844         \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
7845       }
7846       \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
7847     }
7848   }{
7849     \setbox\l__problems_solution_box\vbox\bgroup
7850     \par\smallskip\hrule\smallskip
7851     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
7852   }
7853 }{
7854   \stex_html_backend:TF{
7855     \stex_if_do_html:T{
7856       \end{stex_annotate_env}
7857     }
7858   }{
7859     \smallskip\hrule
7860     \egroup
7861     \bool_if:NT \c__problems_solutions_bool {

```

```

7862     \box\l__problems_solution_box
7863   }
7864 }
7865 }
7866
7867 \newcommand\startsolutions{
7868   \bool_set_true:N \c__problems_solutions_bool
7869   \solutionstrue
7870   % \specialcomment{solution}{\@startsolution}{
7871   %   \bool_if:NF \c__problems_boxed_bool {
7872   %     \hrule\medskip
7873   %   }
7874   %   \end{small}%
7875   % }
7876   % \bool_if:NT \c__problems_boxed_bool {
7877   %   \surroundwithmdframed{solution}
7878   % }
7879 }

```

(End definition for \startsolutions. This function is documented on page 58.)

\stopsolutions

```

7880 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

(End definition for \stopsolutions. This function is documented on page 58.)

exnote (env.)

```

7881 \bool_if:NTF \c__problems_notes_bool {
7882   \newenvironment{exnote}[1][]{
7883     \par\smallskip\hrule\smallskip
7884     \noindent\textbf{\prob@note@kw :~ }\small
7885   }{
7886     \smallskip\hrule
7887   }
7888 }{
7889   \excludecomment{exnote}
7890 }

```

hint (env.)

```

7891 \bool_if:NTF \c__problems_notes_bool {
7892   \newenvironment{hint}[1][]{
7893     \par\smallskip\hrule\smallskip
7894     \noindent\textbf{\prob@hint@kw :~ }\small
7895   }{
7896     \smallskip\hrule
7897   }
7898 }{
7899   \newenvironment{exhint}[1][]{
7900     \par\smallskip\hrule\smallskip
7901     \noindent\textbf{\prob@hint@kw :~ }\small
7902   }{
7903     \smallskip\hrule
7904   }
7905 }{
7906   \excludecomment{hint}

```

```

7906 \excludecomment{exhint}
7907 }

gnote (env.)

7908 \bool_if:NTF \c__problems_notes_bool {
7909   \newenvironment{gnote}[1][ ]{
7910     \par\smallskip\hrule\smallskip
7911     \noindent\textbf{\prob@gnote@kw :~ }\small
7912   }{
7913     \smallskip\hrule
7914   }
7915 }{
7916   \excludecomment{gnote}
7917 }

```

38.3 Multiple Choice Blocks

EdN:13

mcb (env.) ¹³

```

7918 \newenvironment{mcb}{
7919   \begin{enumerate}
7920 }{
7921   \end{enumerate}
7922 }

we define the keys for the mcc macro

7923 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7924   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7925     \bool_set_true:N #1
7926   }{
7927     \bool_set_false:N #1
7928   }
7929 }

7930 \keys_define:nn { problem / mcc }{
7931   id      .str_set:N = \l__problems_mcc_id_str ,
7932   feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7933   T       .default:n = { false } ,
7934   T       .bool_set:N = \l__problems_mcc_t_bool ,
7935   F       .default:n = { false } ,
7936   F       .bool_set:N = \l__problems_mcc_f_bool ,
7937   Ttext   .tl_set:N = \l__problems_mcc_Ttext_tl ,
7938   Ftext   .tl_set:N = \l__problems_mcc_Ftext_tl
7939 }

7940 \cs_new_protected:Nn \l__problems_mcc_args:n {
7941   \str_clear:N \l__problems_mcc_id_str
7942   \tl_clear:N \l__problems_mcc_feedback_tl
7943   \bool_set_false:N \l__problems_mcc_t_bool
7944   \bool_set_false:N \l__problems_mcc_f_bool
7945   \tl_clear:N \l__problems_mcc_Ttext_tl
7946   \tl_clear:N \l__problems_mcc_Ftext_tl
7947   \str_clear:N \l__problems_mcc_id_str
7948   \keys_set:nn { problem / mcc }{ #1 }
7949 }

```

¹³EdNOTE: MK: maybe import something better here from a dedicated MC package

\mcc

```
7950 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
7951 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
7952 \newcommand\mcc[2][{}]{
7953   \l__problems_mcc_args:n{ #1 }
7954   \item[{$\Box$}] #2
7955   \bool_if:NT \c__problems_solutions_bool{
7956     \\\
7957     \bool_if:NT \l__problems_mcc_t_bool {
7958       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7959     }
7960     \bool_if:NT \l__problems_mcc_f_bool {
7961       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7962     }
7963     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7964       \emph{\l__problems_mcc_feedback_tl}
7965     }
7966   }
7967 } %solutions
```

(End definition for \mcc. This function is documented on page 59.)

38.4 Including Problems

\includeproblem The \includeproblem command is essentially a glorified \input statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the inclprob keys after the input.

```
7968
7969 \keys_define:nn{ problem / inclproblem }{
7970   id      .str_set_x:N = \l__problems_inclprob_id_str,
7971   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7972   min     .tl_set:N    = \l__problems_inclprob_min_tl,
7973   title   .tl_set:N    = \l__problems_inclprob_title_tl,
7974   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
7975   type    .tl_set:N    = \l__problems_inclprob_type_tl,
7976   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
7977 }
7978 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7979   \str_clear:N \l__problems_prob_id_str
7980   \tl_clear:N \l__problems_inclprob_pts_tl
7981   \tl_clear:N \l__problems_inclprob_min_tl
7982   \tl_clear:N \l__problems_inclprob_title_tl
7983   \tl_clear:N \l__problems_inclprob_type_tl
7984   \int_zero_new:N \l__problems_inclprob_refnum_int
7985   \str_clear:N \l__problems_inclprob_mhrepos_str
7986   \keys_set:nn { problem / inclproblem }{ #1 }
7987   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7988     \let\l__problems_inclprob_pts_tl\undefined
7989   }
7990   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7991     \let\l__problems_inclprob_min_tl\undefined
7992   }
7993   \tl_if_empty:NT \l__problems_inclprob_title_tl {
```

```

7994 \let\l__problems_inclprob_title_tl\undefined
7995 }
7996 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7997 \let\l__problems_inclprob_type_tl\undefined
7998 }
7999 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8000 \let\l__problems_inclprob_refnum_int\undefined
8001 }
8002 }
8003
8004 \cs_new_protected:Nn \__problems_inclprob_clear: {
8005 \let\l__problems_inclprob_id_str\undefined
8006 \let\l__problems_inclprob_pts_tl\undefined
8007 \let\l__problems_inclprob_min_tl\undefined
8008 \let\l__problems_inclprob_title_tl\undefined
8009 \let\l__problems_inclprob_type_tl\undefined
8010 \let\l__problems_inclprob_refnum_int\undefined
8011 \let\l__problems_inclprob_mhrepos_str\undefined
8012 }
8013 \__problems_inclprob_clear:
8014
8015 \newcommand\includeproblem[2][ ]{
8016 \__problems_inclprob_args:n{ #1 }
8017 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8018 \stex_html_backend:TF {
8019 \str_clear:N \l_tmpa_str
8020 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8021 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8022 }
8023 \stex_annotate_invisible:nnn{includeproblem}{
8024 \l_tmpa_str / #2
8025 }{}}
8026 }{
8027 \begingroup
8028 \inputreftrue
8029 \tl_if_empty:nTF{ ##1 }{
8030 \input{#2}
8031 }{
8032 \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8033 }
8034 \endgroup
8035 }
8036 }
8037 \__problems_inclprob_clear:
8038 }

```

(End definition for `\includeproblem`. This function is documented on page 60.)

38.5 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```

8039 \AddToHook{enddocument}{

```



```

8040 \bool_if:NT \c__problems_pts_bool {
8041   \message{Total:~\arabic{pts}~points}
8042 }
8043 \bool_if:NT \c__problems_min_bool {
8044   \message{Total:~\arabic{min}~minutes}
8045 }
8046 }

```

The margin pars are reader-visible, so we need to translate

```

8047 \def\pts#1{
8048   \bool_if:NT \c__problems_pts_bool {
8049     \marginpar{#1~\prob@pt@kw}
8050   }
8051 }
8052 \def\min#1{
8053   \bool_if:NT \c__problems_min_bool {
8054     \marginpar{#1~\prob@min@kw}
8055   }
8056 }

```

\show@pts The `\show@pts` shows the points: if no points are given from the outside and also no points are given locally do nothing, else show and add. If there are outside points then we show them in the margin.

```

8057 \newcounter{pts}
8058 \def\show@pts{
8059   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8060     \bool_if:NT \c__problems_pts_bool {
8061       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8062       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8063     }
8064   }{
8065     \tl_if_exist:NT \l__problems_prob_pts_tl {
8066       \bool_if:NT \c__problems_pts_bool {
8067         \tl_if_empty:NT\l__problems_prob_pts_tl{
8068           \tl_set:Nn \l__problems_prob_pts_tl {0}
8069         }
8070         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8071         \addtocounter{pts}{\l__problems_prob_pts_tl}
8072       }
8073     }
8074   }
8075 }

```

(End definition for `\show@pts`. This function is documented on page ??.)

and now the same for the minutes

\show@min

```

8076 \newcounter{min}
8077 \def\show@min{
8078   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8079     \bool_if:NT \c__problems_min_bool {
8080       \marginpar{\l__problems_inclprob_min_tl\ min}
8081       \addtocounter{min}{\l__problems_inclprob_min_tl}
8082     }

```

```

8083   }{
8084     \tl_if_exist:NT \l__problems_prob_min_tl {
8085       \bool_if:NT \c__problems_min_bool {
8086         \tl_if_empty:NT\l__problems_prob_min_tl{
8087           \tl_set:Nn \l__problems_prob_min_tl {0}
8088         }
8089         \marginpar{\l__problems_prob_min_tl\ min}
8090         \addtocounter{min}{\l__problems_prob_min_tl}
8091       }
8092     }
8093   }
8094 }
8095 \end{package}

```

(End definition for \show@min. This function is documented on page ??.)

Chapter 39

Implementation: The hwexam Package

39.1 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.

```
8096 \*package>
8097 \ProvidesExplPackage{hwexam}{2022/05/24}{3.1.0}{homework assignments and exams}
8098 \RequirePackage{13keys2e}
8099
8100 \newif\iftest\testfalse
8101 \DeclareOption{test}{\testtrue}
8102 \newif\ifmultiple\multiplefalse
8103 \DeclareOption{multiple}{\multipletrue}
8104 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8105 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
8106 \RequirePackage{keyval}[1997/11/10]
8107 \RequirePackage{problem}
```

`\hwexam@*kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
8108 \newcommand\hwexam@assignment@kw{Assignment}
8109 \newcommand\hwexam@given@kw{Given}
8110 \newcommand\hwexam@due@kw{Due}
8111 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
8112 blank~for~extra~space}
8113 \def\hwexam@minutes@kw{minutes}
8114 \newcommand\correction@probs@kw{prob.}
8115 \newcommand\correction@pts@kw{total}
8116 \newcommand\correction@reached@kw{reached}
8117 \newcommand\correction@sum@kw{Sum}
8118 \newcommand\correction@grade@kw{grade}
8119 \newcommand\correction@forgrading@kw{To~be~used~for~grading,~do~not~write~here}
```

(End definition for `\hwexam@*@kw`. This function is documented on page ??.)

For the other languages, we set up triggers

```

8120 \AddToHook{begindocument}{
8121 \ltx@ifpackageloaded{babel}{
8122 \makeatletter
8123 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8124 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
8125 \input{hwexam-ngerman.ldf}
8126 }
8127 \clist_if_in:NnT \l_tmpa_clist {finnish}{
8128 \input{hwexam-finnish.ldf}
8129 }
8130 \clist_if_in:NnT \l_tmpa_clist {french}{
8131 \input{hwexam-french.ldf}
8132 }
8133 \clist_if_in:NnT \l_tmpa_clist {russian}{
8134 \input{hwexam-russian.ldf}
8135 }
8136 \makeatother
8137 }{}
8138 }
8139

```

39.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```

8140 \newcounter{assignment}
8141 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8142 \keys_define:nn { hwexam / assignment } {
8143 id .str_set:N = \l_@@_assign_id_str,
8144 number .int_set:N = \l_@@_assign_number_int,
8145 title .tl_set:N = \l_@@_assign_title_tl,
8146 type .tl_set:N = \l_@@_assign_type_tl,
8147 given .tl_set:N = \l_@@_assign_given_tl,
8148 due .tl_set:N = \l_@@_assign_due_tl,
8149 loadmodules .code:n = {
8150 \bool_set_true:N \l_@@_assign_loadmodules_bool
8151 }
8152 }
8153 \cs_new_protected:Nn \_@@_assignment_args:n {
8154 \str_clear:N \l_@@_assign_id_str
8155 \int_set:Nn \l_@@_assign_number_int {-1}
8156 \tl_clear:N \l_@@_assign_title_tl
8157 \tl_clear:N \l_@@_assign_type_tl
8158 \tl_clear:N \l_@@_assign_given_tl
8159 \tl_clear:N \l_@@_assign_due_tl
8160 \bool_set_false:N \l_@@_assign_loadmodules_bool
8161 \keys_set:nn { hwexam / assignment }{ #1 }
8162 }

```

The next three macros are intermediate functions that handle the case gracefully, where the respective token registers are undefined.

The `\given@due` macro prints information about the given and due status of the assignment. Its arguments specify the brackets.

```

8163 \newcommand\given@due[2]{
8164 \bool_lazy_all:nF {
8165 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8166 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8167 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8168 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8169 }{ #1 }
8170
8171 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8172 \tl_if_empty:NF \l_@@_assign_given_tl {
8173 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8174 }
8175 }{
8176 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8177 }
8178
8179 \bool_lazy_or:nnF {
8180 \bool_lazy_and_p:nn {
8181 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8182 }{
8183 \tl_if_empty_p:V \l_@@_assign_due_tl
8184 }
8185 }{
8186 \bool_lazy_and_p:nn {
8187 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8188 }{
8189 \tl_if_empty_p:V \l_@@_assign_due_tl
8190 }
8191 }{ ,~ }
8192
8193 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8194 \tl_if_empty:NF \l_@@_assign_due_tl {
8195 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8196 }
8197 }{
8198 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8199 }
8200
8201 \bool_lazy_all:nF {
8202 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8203 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8204 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8205 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8206 }{ #2 }
8207 }

```

`\assignment@title` This macro prints the title of an assignment, the local title is overwritten, if there is one from the `\inputassignment`. `\assignment@title` takes three arguments the first is the

fallback when no title is given at all, the second and third go around the title, if one is given.

```

8208 \newcommand\assignment@title[3]{
8209 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8210 \tl_if_empty:NTF \l_@@_assign_title_tl {
8211 #1
8212 }{
8213 #2\l_@@_assign_title_tl#3
8214 }
8215 }{
8216 #2\l_@@_inclasssign_title_tl#3
8217 }
8218 }

```

(End definition for \assignment@title. This function is documented on page ??.)

\assignment@number Like \assignment@title only for the number, and no around part.

```

8219 \newcommand\assignment@number{
8220 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8221 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8222 \arabic{assignment}
8223 } {
8224 \int_use:N \l_@@_assign_number_int
8225 }
8226 }{
8227 \int_use:N \l_@@_inclasssign_number_int
8228 }
8229 }

```

(End definition for \assignment@number. This function is documented on page ??.)

With them, we can define the central **assignment** environment. This has two forms (separated by \ifmultiple) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

assignment (env.) For the **assignment** environment we delegate the work to the **@assignment** environment that depends on whether **multiple** option is given.

```

8230 \newenvironment{assignment}[1][]{
8231 \_@@_assignment_args:n { #1 }
8232 %\sref@target
8233 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8234 \global\stepcounter{assignment}
8235 }{
8236 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8237 }
8238 \setcounter{problem}{0}
8239 \renewcommand\prob@label[1]{\assignment@number.##1}
8240 \def\current@section@level{\document@hwexamtype}
8241 %\sref@label{id}{\document@hwexamtype \thesection}
8242 \begin{@assignment}
8243 }{
8244 \end{@assignment}
8245 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

8246 \def\ass@title{
8247 {\protect\document@hwexamtype}\arabic{assignment}
8248 \assignment@title{}\;{}{}\;} -- \given@due{}\}
8249 }
8250 \ifmultiple
8251 \newenvironment{@assignment}{
8252 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8253 \begin{sfragment}[loadmodules]{\ass@title}
8254 }{
8255 \begin{sfragment}{\ass@title}
8256 }
8257 }{
8258 \end{sfragment}
8259 }

```

for the single-page case we make a title block from the same components.

```

8260 \else
8261 \newenvironment{@assignment}{
8262 \begin{center}\bf
8263 \Large@title\strut\
8264 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;{}{}\;{}\\}
8265 \large\given@due{--\;}\;{}{--}
8266 \end{center}
8267 }{}
8268 \fi% multiple

```

39.3 Including Assignments

\in*assignment This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points. Importantly, it resets the `inclassig` keys after the input.

```

8269 \keys_define:nn { hwexam / inclassignment } {
8270 %id .str_set_x:N = \l_@@_assign_id_str,
8271 number .int_set:N = \l_@@_inclassign_number_int,
8272 title .tl_set:N = \l_@@_inclassign_title_tl,
8273 type .tl_set:N = \l_@@_inclassign_type_tl,
8274 given .tl_set:N = \l_@@_inclassign_given_tl,
8275 due .tl_set:N = \l_@@_inclassign_due_tl,
8276 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8277 }
8278 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8279 \int_set:Nn \l_@@_inclassign_number_int {-1}
8280 \tl_clear:N \l_@@_inclassign_title_tl
8281 \tl_clear:N \l_@@_inclassign_type_tl
8282 \tl_clear:N \l_@@_inclassign_given_tl
8283 \tl_clear:N \l_@@_inclassign_due_tl
8284 \str_clear:N \l_@@_inclassign_mhrepos_str
8285 \keys_set:nn { hwexam / inclassignment }{ #1 }
8286 }
8287 \l_@@_inclassignment_args:n {}
8288
8289 \newcommand\inputassignment[2][]{

```

```

8290 \_@@_inclassignment_args:n { #1 }
8291 \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8292   \input{#2}
8293 }{
8294   \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8295     \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
8296   }
8297 }
8298 \_@@_inclassignment_args:n {}
8299 }
8300 \newcommand\includeassignment[2][]{
8301   \newpage
8302   \inputassignment[#1]{#2}
8303 }

```

(End definition for \in*assignment. This function is documented on page ??.)

39.4 Typesetting Exams

\quizheading

```

8304 \ExplSyntaxOff
8305 \newcommand\quizheading[1]{%
8306   \def\@tas{#1}%
8307   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8308   \ifx\@tas\@empty\else%
8309     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8310   \fi%
8311 }
8312 \ExplSyntaxOn

```

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

\testheading

```

8313
8314 \def\hwexamheader{\input{hwexam-default.header}}
8315
8316 \def\hwexamminutes{
8317   \tl_if_empty:NTF \testheading@duration {
8318     {\testheading@min}~\hwexam@minutes@kw
8319   }{
8320     \testheading@duration
8321   }
8322 }
8323
8324 \keys_define:nn { hwexam / testheading } {
8325   min .tl_set:N = \testheading@min,
8326   duration .tl_set:N = \testheading@duration,
8327   reqpts .tl_set:N = \testheading@reqpts,
8328   tools .tl_set:N = \testheading@tools
8329 }
8330 \cs_new_protected:Nn \_@@_testheading_args:n {
8331   \tl_clear:N \testheading@min
8332   \tl_clear:N \testheading@duration

```



```

8333 \tl_clear:N \testheading@reqpts
8334 \tl_clear:N \testheading@tools
8335 \keys_set:nn { hwexam / testheading }{ #1 }
8336 }
8337 \newenvironment{testheading}[1][ ]{
8338 \_@@_testheading_args:n{ #1 }
8339 \newcount\check@time\check@time=\testheading@min
8340 \advance\check@time by -\theassignment@totalmin
8341 \newif\if@bonuspoints
8342 \tl_if_empty:NTF \testheading@reqpts {
8343 \@bonuspointsfalse
8344 }{
8345 \newcount\bonus@pts
8346 \bonus@pts=\theassignment@totalpts
8347 \advance\bonus@pts by -\testheading@reqpts
8348 \edef\bonus@pts{\the\bonus@pts}
8349 \@bonuspointstrue
8350 }
8351 \edef\check@time{\the\check@time}
8352
8353 \makeatletter\hwexamheader\makeatother
8354 }{
8355 \newpage
8356 }

```

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

\testspace

```

8357 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8358 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8359 \newcommand\testemptypage[1][ ]{\iftest\begin{center}\hwexam@testemptypage@kw\end{center}\vfi}

```

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

\@problem This macro acts on a problem's record in the *.aux file. Here we redefine it (it was defined to do nothing in problem.sty) to generate the correction table.

```

8360 <@@=problems>
8361 \renewcommand\@problem[3]{
8362 \stepcounter{assignment@probs}
8363 \def\__problemspts{#2}
8364 \ifx\__problemspts\@empty\else
8365 \addtocounter{assignment@totalpts}{#2}
8366 \fi
8367 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8368 \xdef\correction@probs{\correction@probs & #1}%
8369 \xdef\correction@pts{\correction@pts & #2}
8370 \xdef\correction@reached{\correction@reached &}

```

```

8371 }
8372 <@@=hwexam>

```

(End definition for \@problem. This function is documented on page ??.)

`\correction@table` This macro generates the correction table

```

8373 \newcounter{assignment@probs}
8374 \newcounter{assignment@totalpts}
8375 \newcounter{assignment@totalmin}
8376 \def\correction@probs{\correction@probs@kw}
8377 \def\correction@pts{\correction@pts@kw}
8378 \def\correction@reached{\correction@reached@kw}
8379 \stepcounter{assignment@probs}
8380 \newcommand\correction@table{
8381 \resizebox{\textwidth}{!}{%
8382 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8383 &\multicolumn{\theassignment@probs}{c|}||%|
8384 {\footnotesize\correction@forgrading@kw} &\\ \hline
8385 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8386 \correction@pts & \theassignment@totalpts & \\ \hline
8387 \correction@reached & & \[.7cm]\hline
8388 \end{tabular}}
8389 </package>

```

(End definition for \correction@table. This function is documented on page ??.)

39.5 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}

```

Chapter 40

References

EdN:14

14

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T_EX distribution. The Comprehensive T_EX Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphicx.pdf>.
- [DCM03] The DCMi Usage Board. *DCMI Metadata Terms*. DCMi Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

¹⁴EdNOTE: we need an un-numbered version sfragment*

- [SIa] *sLaTeX/sTeX-IDE*. URL: <https://github.com/slatex/sTeX-IDE> (visited on 04/22/2022).
- [SIb] *sLaTeX/stexls-vscode-plugin*. URL: <https://github.com/slatex/stexls-vscode-plugin> (visited on 04/22/2022).
- [SLS] *sLaTeX/stexls*. URL: <https://github.com/slatex/stexls> (visited on 04/22/2022).
- [ST] *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: <https://ctan.org/pkg/stex> (visited on 04/22/2022).
- [sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: <https://github.com/sLaTeX/sTeX> (visited on 05/11/2020).
- [Tana] Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: <http://ctan.org/pkg/beamer> (visited on 01/07/2014).
- [Tanb] Till Tantau. *User Guide to the Beamer Class*. URL: <http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.
- [TL] *TeX Live*. URL: <http://www.tug.org/texlive/> (visited on 12/11/2012).