

The sTeX3 Package Collection *

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

2022-07-28

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 2022-07-28)

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	28
3.4.1	Multilinguality and Translations	28
3.4.2	Simple Inheritance and Namespaces	29
3.4.3	The mathstructure Environment	31
3.4.4	The copymodule Environment	34
3.4.5	The interpretmodule Environment	35
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>TeX</code> Statements	41
5.1	Definitions, Theorems, Examples, Paragraphs	41
5.2	Proofs	44
5.3	Highlighting and Presentation Customizations	49
6	Additional Packages	51
6.1	Tikzinput: Treating TIKZ code as images	51
6.2	Modular Document Structuring	52
6.2.1	Introduction	52
6.2.2	Package Options	52
6.2.3	Document Fragments	52
6.2.4	Ending Documents Prematurely	54
6.2.5	Global Document Variables	54
6.3	Slides and Course Notes	54
6.3.1	Introduction	54
6.3.2	Package Options	55
6.3.3	Notes and Slides	55
6.3.4	Customizing Header and Footer Lines	56
6.3.5	Frame Images	57
6.3.6	Excursions	58
6.4	Representing Problems and Solutions	59
6.4.1	Introduction	59
6.4.2	Problems and Solutions	59
6.4.3	Markup for Added-Value Services	61
	Multiple Choice Blocks	61
	Filling-In Concrete Solutions	62
6.4.4	Including Problems	63
6.5	Homeworks, Quizzes and Exams	64
6.5.1	Introduction	64
6.5.2	Package Options	64
6.5.3	Assignments	64
6.5.4	Including Assignments	65
6.5.5	Typesetting Exams	65
II	Documentation	67
7	<code>TeX</code>-Basics	68
7.1	Macros and Environments	68
7.1.1	HTML Annotations	68
7.1.2	Babel Languages	69
7.1.3	Auxiliary Methods	69
8	<code>TeX</code>-MathHub	70
8.1	Macros and Environments	70
8.1.1	Files, Paths, URIs	70
8.1.2	MathHub Archives	71
8.1.3	Using Content in Archives	72

9	sTeX-References	73
9.1	Macros and Environments	73
9.1.1	Setting Reference Targets	73
9.1.2	Using References	74
10	sTeX-Modules	75
10.1	Macros and Environments	75
10.1.1	The <code>smodule</code> environment	77
11	sTeX-Module Inheritance	79
11.1	Macros and Environments	79
11.1.1	SMS Mode	79
11.1.2	Imports and Inheritance	80
12	sTeX-Symbols	82
12.1	Macros and Environments	82
13	sTeX-Terms	84
13.1	Macros and Environments	84
14	sTeX-Structural Features	86
14.1	Macros and Environments	86
14.1.1	Structures	86
15	sTeX-Statements	87
15.1	Macros and Environments	87
16	sTeX-Proofs: Structural Markup for Proofs	88
17	sTeX-Metatheory	89
17.1	Symbols	89
III	Extensions	90
18	Tikzinput: Treating TIKZ code as images	91
18.1	Macros and Environments	91
19	document-structure: Semantic Markup for Open Mathematical Documents in L^AT_EX	92
20	NotesSlides – Slides and Course Notes	93
21	problem.sty: An Infrastructure for formatting Problems	94
22	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	95
IV	Implementation	96

23	STeX-Basics Implementation	97
23.1	The STeXDocument Class	97
23.2	Preliminaries	98
23.3	Messages and logging	98
23.4	HTML Annotations	99
23.5	Babel Languages	101
23.6	Persistence	103
23.7	Auxiliary Methods	103
24	STeX-MathHub Implementation	107
24.1	Generic Path Handling	107
24.2	PWD and kpsewhich	109
24.3	File Hooks and Tracking	110
24.4	MathHub Repositories	111
24.5	Using Content in Archives	116
25	STeX-References Implementation	120
25.1	Document URIs and URLs	120
25.2	Setting Reference Targets	122
25.3	Using References	124
26	STeX-Modules Implementation	127
26.1	The smodule environment	131
26.2	Invoking modules	137
27	STeX-Module Inheritance Implementation	139
27.1	SMS Mode	139
27.2	Inheritance	143
28	STeX-Symbols Implementation	149
28.1	Symbol Declarations	149
28.2	Notations	157
28.3	Variables	165
29	STeX-Terms Implementation	173
29.1	Symbol Invocations	173
29.2	Terms	180
29.3	Notation Components	184
29.4	Variables	186
29.5	Sequences	189
30	STeX-Structural Features Implementation	190
30.1	Imports with modification	191
30.2	The feature environment	199
30.3	Structure	199
31	STeX-Statements Implementation	210
31.1	Definitions	210
31.2	Assertions	216
31.3	Examples	219
31.4	Logical Paragraphs	222

32 The Implementation	227
32.1 Proofs	227
33 \TeX-Others Implementation	236
34 \TeX-Metatheory Implementation	238
35 Tikzinput Implementation	241
36 document-structure.sty Implementation	244
36.1 Package Options	244
36.2 Document Structure	245
36.3 Front and Backmatter	249
36.4 Global Variables	251
37 NotesSlides – Implementation	252
37.1 Class and Package Options	252
37.2 Notes and Slides	254
37.3 Header and Footer Lines	258
37.4 Frame Images	260
37.5 Sectioning	261
37.6 Excursions	264
38 The Implementation	266
38.1 Package Options	266
38.2 Problems and Solutions	267
38.3 Markup for Added Value Services	274
38.4 Multiple Choice Blocks	274
38.5 Filling in Concrete Solutions	275
38.6 Including Problems	275
38.7 Reporting Metadata	277
39 Implementation: The hwexam Package	279
39.1 Package Options	279
39.2 Assignments	280
39.3 Including Assignments	283
39.4 Typesetting Exams	284
39.5 Leftovers	286
40 References	287

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 sTeX Archives (Manual Setup)

Writing semantically annotated sTeX becomes much easier, if we can use well-designed libraries of already annotated content. sTeX provides such libraries as 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 sTeX archive as an **archive path** `<apath>` and a name `<archive>`. We can clone the sTeX archive by the following command-line instructions:

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

Note that 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 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 sTeX IDE

We are currently working on an sTeX IDE as an 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 sTeX 1 [\[SLS; Stb\]](#).

2.1.5 Manual Setup for Active Documents and Knowledge Management Services

Foregoing on the 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 sTeX/MMT content archives.

- **sTeX Archives** If we only care about L^AT_EX 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) 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.

- **RuSTeX** The MMT system will also set up 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 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         \realdivide[frac]{1}{
16           \realpower{2}{\svar{n}}
17         }
18       }}
19     \end{sdefinition}
20
21   \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
22     The \symname{geometricSeries} \symname{converges} towards $1$.
23   \end{sassertion}
24 \end{smodule}
25 \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 \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[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`, `\realddivide`, `\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 \LaTeX -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 \LaTeX 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). \LaTeX 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`
`\definiendum`

The `\define{geometricSeries} ...`

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.

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

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

<hr/> <code>\svar</code> <hr/>	The <code>\svar{n}</code> command marks up the <code>n</code> as a variable with name <code>n</code> and notation <code>n</code> .
<hr/> <code>\definiens</code> <hr/>	The <code>sdefinition</code> -environment additionally provides the <code>\definiens</code> -command, which allows for explicitly marking up its argument as the <i>definiens</i> of the symbol currently being defined.

2.2.1 OMDoc/xhtml Conversion

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

TODO VSCode Plugin

Using `Ru\TeX` [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>
```

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

```
<OMBIND>
  <OMID name="...?series?infinitesum"/>
  <OMV name="n"/>
```

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

```

<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 `mathhub=/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 `\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. `\def\mathhub{/path/to/mathhub}\usepackage{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 `~/.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/arithmetics,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:

<div style="border-bottom: 1px solid black; padding-bottom: 5px;">$\backslash\text{mhinput}$</div>	$\backslash\text{mhinput}$ [Some/Archive]{some/file} directly inputs the file some/file in the source-folder of Some/Archive.
---	---

<div style="border-bottom: 1px solid black; padding-bottom: 5px;">$\backslash\text{inputref}$</div>	$\backslash\text{inputref}$ [Some/Archive]{some/file} behaves like $\backslash\text{mhinput}$, but wraps the input in a <code>\begin{group} ... \end{group}</code> . When converting to <code>xhtml</code> , 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 $\backslash\text{inputref}$ is likely to be preferred over $\backslash\text{mhinput}$ because it leads to less duplication in the generated `xhtml`.

<div style="border-bottom: 1px solid black; padding-bottom: 5px;">$\backslash\text{ifinput}$</div>	Both $\backslash\text{mhinput}$ and $\backslash\text{inputref}$ set $\backslash\text{ifinput}$ 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.
---	---

<div style="border-bottom: 1px solid black; padding-bottom: 5px;">$\backslash\text{addmhbibresource}$</div>	$\backslash\text{addmhbibresource}$ [Some/Archive]{some/file} searches for a file like $\backslash\text{mhinput}$ does, but calls $\backslash\text{addbibresource}$ to the result and looks for the file in the archive root directory directly, rather than the <code>source</code> directory. Typical invocations are
--	---

- $\backslash\text{addmhbibresource}\{\text{lib}/\text{refs.bib}\}$, which specifies a bibliography in the `lib` folder in the local archive or
- $\backslash\text{addmhbibresource}[\text{HW}/\text{meta-inf}]\{\text{lib}/\text{refs.bib}\}$ 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` A new module is declared using the basic syntax

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

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

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

`title` (*token list*) to display in customizations.

`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 $\text{\texttt{sTeX}}$ module corresponds to an MMT/OMDOC *theory*. As such it
 \rightarrow gets assigned a module URI (*universal resource identifier*) of the form
 \rightsquigarrow `\T` \rightsquigarrow `<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 `\smodulename`, `\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 `<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 \TeX 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[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 $\text{\S}\text{\TeX}$ 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: · ; b: ·`

\hookrightarrow `\symbolname!` is translated to OMDOC/MMT as `<OMS name="...?symbolname"/>`
 \hookrightarrow 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
 \hookrightarrow to OMBIND-terms in OMDOC/MMT, rather than OMA.
 \rightsquigarrow `T` \rightsquigarrow

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.
- \rightarrow Correspondingly, the name “type” should be taken with a grain of salt, since
- \rightsquigarrow 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.

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

When $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ 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. $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ starts out with $p_d = \text{\texttt{\textcode{\neginfprec}}}$.
2. $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ encounters `\addition` with $p_{op} = 100$. Since $100 \not> \text{\texttt{\textcode{\neginfprec}}}$, it inserts no parentheses.
3. Next, $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ encounters the two arguments for `\addition`. Both have no specifically provided argument precedence, so $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ uses $p_d = p_{op} = 100$ for both and recurses.



4. Next, $\text{\S}\text{\TeX}$ encounters $\text{\textcolor{teal}{multiplication}\{b,\dots\}}$, whose notation has $p_{op} = 50$.
5. We compare to the current downward precedence p_d set by $\text{\textcolor{teal}{addition}}$, arriving at $p_{op} = 50 \not> 100 = p_d$, so $\text{\S}\text{\TeX}$ again inserts no parentheses.
6. Since the notation of $\text{\textcolor{teal}{multiplication}}$ has no explicitly set argument precedences, $\text{\S}\text{\TeX}$ uses the operator precedence for all arguments of $\text{\textcolor{teal}{multiplication}}$, hence sets $p_d = p_{op} = 50$ and recurses.
7. Next, $\text{\S}\text{\TeX}$ encounters the inner $\text{\textcolor{teal}{addition}\{c,\dots\}}$ whose notation has $p_{op} = 100$.
8. We compare to the current downward precedence p_d set by $\text{\textcolor{teal}{multiplication}}$, arriving at $p_{op} = 100 > 50 = p_d$ – which finally prompts $\text{\S}\text{\TeX}$ 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 $\text{\textcolor{blue}{symdecl}}$, $\text{\textcolor{blue}{notation}}$, $\text{\textcolor{blue}{symdef}}$ etc. are disabled outside of $\text{\textcolor{black}{smodule}}$ -environments.

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

$\text{\textcolor{blue}{svar}}$

So far, we have always used variables using $\text{\textcolor{blue}{svar}\{n\}}$, which marks-up n as a variable with name n . More generally, $\text{\textcolor{blue}{svar}\{foo\}\{<\textcolor{black}{texcode}>\}}$ marks-up the arbitrary $<\textcolor{black}{texcode}>$ 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.

$\text{\textcolor{blue}{vardef}}$

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

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  $\text{\textcolor{teal}{varf}}!:\text{\textcolor{teal}{funtype}\{\text{\textcolor{teal}{Nat}}\}\{\text{\textcolor{teal}{Nat}}\}}$ ,
12 by  $\text{\textcolor{teal}{addition}\{\text{\textcolor{teal}{varf}}!,\text{\textcolor{teal}{varn}}\}}$  we mean the function
13  $\text{\textcolor{teal}{fun}\{\text{\textcolor{teal}{varx}}\}\{\text{\textcolor{teal}{varf}}\{\text{\textcolor{teal}{addition}\{\text{\textcolor{teal}{varx}},\text{\textcolor{teal}{varn}}\}}\}}$ 

```


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 \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  $i$ th index of  $\text{\seqa!}$  is  $\text{\seqa}{i}$ .
```

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  $\text{\addition}\{\text{\seqa}\}$ 
```

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

Output:

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

3.4 Module Inheritance and Structures

The \LaTeX 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 \LaTeX) 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 \LaTeX we will see a very simple application of modules: managing multilinguality modularly.

3.4.1 Multilinguality and Translations

If we load the \LaTeX document class or package with the option `lang=<lang>`, \LaTeX will load the appropriate `babel` language for you – e.g. `lang=de` will load the `babel` language

ngerman. Additionally, it makes \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 \TeX package option is set that allows for inferring a language, \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:
 \TeX \rightarrow `\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`.
 \TeX \rightarrow 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 `lcm(a,b)` in *english*, but is called *kleinstes gemeinsames Vielfaches* in *german* and consequently denoted as `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   $\lcm{a,b}$ von zwei Zahlen $a,b$ ist...
6 \end{smodule}
```

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

`\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` 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)`
- `→` – a *dependent record type with manifest fields*, the fields of which are generated from (and correspond to) the constants in `<name>-structure`.
- `\instantiate` 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 $\varM!:=\mathstrut{\varM{universe},\varM{op}!,\varM{unit}}$
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 $\varMb!:=\mathstrut{\varMb{universe},\varMb{op}!,\varMb{unit}}$
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: $\rtimes a\{rplus c\{rtimes d\}}$
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 sTeX Metatheory)

The `stex-metattheory` package contains 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 sTeX module.

We can also see the `stex-metattheory` 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-metattheory` 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 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 sTeX collection due to the obiquity of the symbols involved. Note however, that the metatheory is for all practical purposes a “normal” sTeX module, and the symbols contained “normal” 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

`\symref`
`\symname`

We have already seen `\symname` and `\symref`, the latter being the more general.

`\symref{<symbolname>}{<code>}` marks-up `<code>` as referencing `<symbolname>`. Since quite often, the `<code>` should be (a variant of) the name of the symbol anyway, we also have `\symname{<symbolname>}`.

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 M \rightarrow As expected, the above example is translated to OMDoc/MMT as an
 \rightarrow M \rightarrow OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and
 \rightsquigarrow T \rightsquigarrow `<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:

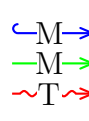

```
1 \begin{sexample}[
2   id=additionandmultiplication.ex,
3   for={addition,multiplication},
4   type={trivial,boring},
5   title={An Example}
6 ]
7   $\addition{2,3}$ is $5$, $\multiplication{2,3}$ is $6$.
8 \end{sexample}
```

Output:

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

`\definiendum`
`\definame`
`\Definame`

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

 The special `type=symdoc` for `sparagraph` is intended to be used for “informal definitions”, or encyclopedia-style descriptions for symbols.
 The MMT system can use those (in lieu of an actual `sdefinition` in scope) to 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   \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
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...

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.

Its central component is the `sproof`-environment, whose body consists of:

- *subproofs* via the `subproof`-environment,
- *proof steps* via the `\spfstep`, `\eqstep` `\assumption`, and `\conclude` macros, and
- *comments*, via normal text without special markup.

`sproof`, `subproof` and the various proof step macros take the following optional arguments:

`id` ($\langle string \rangle$) for referencing,

`method` ($\langle string \rangle$) the proof method (e.g. contradiction, induction,...)

`term` ($\langle token list \rangle$) the (ideally semantically-marked up) proposition that is derived/proven by this proof/subproof/proof step.

Additionally, they take one mandatory argument for the document text to be annotated, or (in the case of the environments) as an introductory description of the proof itself. Since the latter often contains the `term` to be derived as text, alternatively to providing it as an optional argument, the mandatory argument can use the `\yield`-macro to mark it up in the text.

The `sproof` and `subproof` environments additionally take two optional arguments:

`for` the symbol identifier/name corresponding to the `sassertion` to be proven. This too subsumes `\yield` and the `term`-argument.

`hide` In the pdf, this only shows the mandatory argument text and hides the body of the environment. In the HTML (as served by MMT), the bodies of all `proof` and `subproof` environments are *collapsible*, and `hide` collapses the body by default.

```

1 \begin{sassertion}[type=theorem,name=sqrt2irr]
2   \conclusion{\irrational{\arg{\realroot{2}}$ is \comp{irrational}}}.
3 \end{sassertion}
4
5 \begin{sproof}[for=sqrt2irr,method=contradiction]{By contradiction}
6   \assumption{Assume \yield{\rational{\arg{\realroot{2}}$ is
7     \comp{rational}}}}
8   \begin{subproof}[method=straightforward]{Then
9     \yield{\$eq{\ratfrac{\intpow{\vara}{2}}{\intpow{\varb}{2}}{2}$
10       for some $\inset{\vara,\varb}\PosInt$ with
11       \coprime{\arg{\vara},\arg{\varb}$ \comp{coprime}}}}
```

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**

```

12 \assumption{By assumption, \yield{there are
13 $\inset{\vara,\varb}\PosInt$ with
14 $\realroot{2}=\ratfrac{\vara}{\varb}$}}
15 \spfstep{wlog, we can assume \coprime{$\arg{\vara},\arg{\varb}$}
16 to be \comp{coprime}}
17 % a comment:
18 If not, reduce the fraction until numerator and denominator
19 are coprime, and let the resulting components be
20 $\vara$ and $\varb$
21 \spfstep{Then \yield{$\eq{\intpow{\ratfrac{\vara}{\varb}}{2}{2}$}}
22 \eqstep{\ratfrac{\intpow{\vara}{2}}{\intpow{\varb}{2}}}}
23 \end{subproof}
24 \begin{subproof}[term=\divides{2}{\vara},method=straightforward]{
25 Then $\vara$ is even}
26 \spfstep{Multiplying the equation by $\intpow{\varb}{2}$ yields
27 $\yield{\eq{\intpow{\vara}{2}}{\inttimes{2}{\intpow{\varb}{2}}}}$}
28 \spfstep[term=\divides{2}{\intpow{\vara}{2}}]{Hence
29 $\intpow{\vara}{2}$ is even}
30 \conclude[term=\divides{2}{\vara}]{Hence $\vara$ is even as well}
31 % another comment:
32 Hint: Think about the prime factorizations of $\vara$ and
33 $\intpow{\vara}{2}$
34 \end{subproof}
35 \begin{subproof}[term=\divides{2}{\varb},method=straightforward,]{
36 Then $\varb$ is also even}
37 \spfstep{Since $\vara$ is even, we have \yield{some $\varc$ $
38 such that $\eq{\inttimes{2}{\varc}}{\vara}$}}
39 \spfstep{Plugging into the above, we get
40 \yield{$\eq{\intpow{\inttimes{2}{\vara}}{2}
41 {\inttimes{2}{\intpow{\varb}{2}}}$}}
42 \eqstep{\inttimes{4}{\intpow{\vara}{2}}}
43 \spfstep{Dividing both sides by $2$ yields
44 \yield{$\eq{\intpow{\varb}{2}}{\inttimes{2}{\intpow{\vara}{2}}}$}}
45 \spfstep[term=\divides{2}{\intpow{\varb}{2}}]{Hence
46 $\intpow{\varb}{2}$ is even}
47 \conclude[term=\divides{2}{\varb}]{Hence $\varb$ is even}
48 % one more comment:
49 By the same argument as above
50 \end{subproof}
51 \conclude[term=\contradiction]{Contradiction to $\vara,\varb$ being
52 \symname{coprime}.}
53 \end{sproof}

```

which will produce:

Theorem 5.2.1. $\sqrt{2}$ is *irrational*.

Proof: By contradiction

1. Assume $\sqrt{2}$ is *rational*
2. Then $(\frac{a}{b})^2=2$ for some $a,b \in \mathbb{Z}^+$ with a,b *coprime*
 - 2.1. By assumption, there are $a,b \in \mathbb{Z}^+$ with $\sqrt{2} = \frac{a}{b}$
 - 2.2. wlog, we can assume a,b to be *coprime*

If not, reduce the fraction until numerator and denominator are coprime, and let the re-

sulting components be a and b

2.3. Then $(\frac{a}{b})^2=2$

$$= \frac{a^2}{b^2}$$

3. Then a is even

3.1. Multiplying the equation by b^2 yields $a^2=2b^2$

3.2. Hence a^2 is even

\Rightarrow Hence a is even as well

Hint: Think about the prime factorizations of a and a^2

4. Then b is also even

4.1. Since a is even, we have some c such that $2c=a$

4.2. Plugging into the above, we get $(2a)^2=2b^2$

$$= 4a^2$$

4.3. Dividing both sides by 2 yields $b^2=2a^2$

4.4. Hence b^2 is even

\Rightarrow Hence b is even

By the same argument as above

\Rightarrow Contradiction to a, b being coprime.

□

If we mark all subproofs with `hide`, we will obtain the following instead:

Theorem 5.2.2. $\sqrt{2}$ is irrational.

Proof: By contradiction

1. Assume $\sqrt{2}$ is rational

2. Then $(\frac{a}{b})^2=2$ for some $a, b \in \mathbb{Z}^+$ with a, b coprime

3. Then a is even

4. Then b is also even

\Rightarrow Contradiction to a, b being coprime.

□

However, the hidden subproofs will still be shown in the HTML, only in an expandable section which is collapsed by default.

The above style of writing proofs is usually called *structured proofs*. They have a huge advantage over the traditional purely prosaic style, in that (as the name suggests) the actual *structure* of the proof is made explicit, which almost always makes it considerably more comprehensible. We, among many others, encourage the general use of structured proofs.

Alas, most proofs are not written in this style, and we would do users a disservice by insisting on this style. For that reason, the `spfblock` environment turns all subproofs and proof step macros into presentationally neutral *inline* annotations, as in the induction step of the following example:

```
1 \begin{sproof}[id=simple-proof,method=induction]
2   {We prove that  $\sum_{i=1}^{n-1} n^{2i-1} = n^2$  by induction over  $n$ }
```

```

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

```

This yields the following result:

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

For the induction we have to consider three cases:

1. $n = 1$

then we compute $1 = 1^2$

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 $1 + 3 = 2^2 = 4$.

3. $n > 1$

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

We have to show that we can derive the assertion for $n = k + 1$ from this assumption,

i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.
 We obtain $\sum_{i=1}^{k+1} 2i - 1 = \sum_{i=1}^k 2i - 1 + 2(k + 1) - 1$ by [splitting the sum](#). Thus
 we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by [induction hypothesis](#). We can [simplify](#) the
 right-hand side to $k + 1^2$, which proves the assertion.
 \Rightarrow We have considered all the cases, so we have proven the assertion. □

proof The **proof** 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 **spfstep**, **spfcomment**, and **spfcases** environments that are used to markup the proof steps.

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

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

\spfstep Regular proof steps are marked up with the **\spfstep** macro, 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.

\yield See above

\spfjust This evidence is marked up with the **\spfjust** macro in the **stex-proofs** package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence (ideally, a semantically marked-up term).

\assumption The **\assumption** macro allows to mark up a (justified) assumption.

\justarg

subproof The **subproof** environment is used to mark up a subproof. This environment takes an optional **KeyVal** argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the **proof** environment). The **method** key can be used to give the name of the proof method executed to make this subproof.

`\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 \LaTeX 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 \LaTeX 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 \LaTeX 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}}

```

`\compemph`
`\varemp`
`\symrefemph`
`\defemph`

Apart from the environments, we can control how \TeX highlights variables, notation components, `\symrefs` and `\definiendums`, respectively.

To do so, we simply redefine these four macros. For example, to highlight notation components (i.e. everything in a `\comp`) in blue, as in this document, we can do `\def\compemph#1{\textcolor{blue}{#1}}`. By default, `\compemph` et al do nothing.

`\compemph@uri`
`\varemp@uri`
`\symrefemph@uri`
`\defemph@uri`

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

```

1 \protected\def\symrefemph@uri#1#2{
2   \pdftooltip{
3     \srefsymuri{#2}{\symrefemph{#1}}
4   }{
5     URI:~\detokenize{#2}
6   }
7 }

```

By default, `\compemph@uri` is simply defined as `\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 `ikzinput` package is determined by whether the `image` option is given. If it is not, then the `tikz` package is loaded, all other options are passed on to it and `\tikzinput{<file>}` inputs the TIKZ file `<file>.tex`; if not, only the `graphicx` package is loaded and `\tikzinput{<file>}` loads an image file `<file>.<ext>` generated from `<file>.tex`.

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

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

The `standalone` class is a minimal \LaTeX class that when loaded in a document that uses the `standalone` package: the preamble and the `document` environment are disregarded during loading, so they do not pose any problems. In effect, an `\input` of the file above only sees the `tikzpicture` environment, but the file itself is standalone in the sense that we can run \LaTeX over it separately, e.g. for generating an image file from it.

\tikzinput
\ctikzinput

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

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

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

`\mhtikzinput`
`\cmhtikzinput`

`\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.

`\libusetikzlibrary`

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

6.2 Modular Document Structuring

6.2.1 Introduction

The `document-structure` package supplies an infrastructure for writing OMDOC documents in \LaTeX . This includes a simple structure sharing mechanism for \TeX 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 \TeX 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 \TeX sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the \TeX 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 <code><name>.cls</code> instead of <code>article.cls</code>
<code>topsect=<sect></code>	The top-level sectioning level; the default for <code><sect></code> is <code>section</code>

6.2.3 Document Fragments

sfragment

The structure of the document is given by nested **sfragment** environments. In the \LaTeX 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 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.

`\currentsectionlevel`
`\CurrentSectionLevel`

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

6.2.4 Ending Documents Prematurely

`\prematurestop`
`\afterprematurestop`

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

`\prematurestop` is useful when one has a driver file, e.g. for a course taught multiple years and wants to generate course notes up to the current point in the lecture. Instead of commenting out the remaining parts, one can just move the `\prematurestop` macro. This is especially useful, if we need the rest of the file for processing, e.g. to generate a theory graph of the whole course with the already-covered parts marked up as an overview over the progress; see `import_graph.py` from the `lmhtools` utilities [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.

`\setSGvar`
`\useSGvar`

`\setSGvar{⟨vname⟩}{⟨text⟩}` to set the global variable `⟨vname⟩` to `⟨text⟩` and `\useSGvar{⟨vname⟩}` to reference it.

`\ifSGvar`

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

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:

<u>slides</u> <u>notes</u>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see subsection 6.3.3).
<u>sectocframes</u>	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.
<u>frameimages</u> <u>fiboxed</u>	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** Slides are represented with the `frame` environment just like in the `beamer` class, see [\[Tanb\]](#) for details.
- note** 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 `\notestfalse` 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*

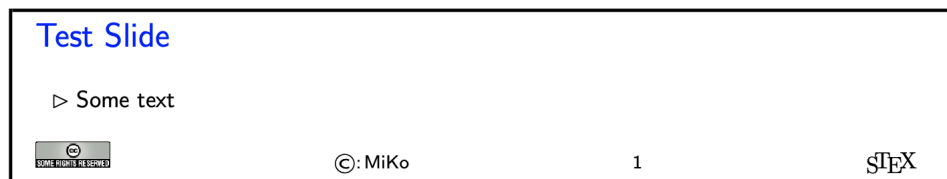
If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

`nexample`, `nsproof`, `nassertion`

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

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.

<code>\setslidelogo</code>	The default logo provided by the <code>notesslides</code> package is the \LaTeX logo it can be customized using <code>\setslidelogo{<logo name>}</code> .
----------------------------	--

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

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

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

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

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


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

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

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

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

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

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

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

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

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



When option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `sfragment` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made. This is a problem of the underlying document-structure package.

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

<hr/> <code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code> (should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?), <code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then the corresponding auxiliary parts of the problems are output, otherwise, they remain invisible.
<code>notes</code>	
<code>hints</code>	
<code>gnotes</code>	
<code>pts</code>	
<code>min</code>	
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so 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.
<code>test</code>	
<hr/>	
<code>problem</code>	The main environment provided by the <code>problempackage</code> is (surprise surprise) the <code>problem</code> environment. It is used to mark up problems and exercises. The environment takes an optional <code>KeyVal</code> argument with the keys <code>id</code> as an identifier that can be reference later, <code>pts</code> for the points to be gained from this exercise in homework or quiz situations, <code>min</code> for the estimated minutes needed to solve the problem, and finally <code>title</code> 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 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,exnote,gnote The `hint` and `exnote` environments can be used in a `problem` environment to give hints and to make notes that elaborate certain aspects of the problem. The `gnote` (grading notes) environment can be used to document situations that may arise in grading.

\startsolutions
\stopsolutions

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

\ifsolutions

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

6.4.3 Markup for Added-Value Services

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

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

Multiple Choice Blocks

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

`\mcc` `\mcc[⟨keyvals⟩]{⟨text⟩}` takes an optional key/value argument `⟨keyvals⟩` for choice 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 context. 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=Noooooooooooo,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).

Filling-In Concrete Solutions

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

`\fillinsol`

The `\fillinsol` macro takes⁶ an a single argument, which contains a concrete solution (i.e. a number, a string, ...), which generates a fill-in-box in test mode:

Example 43

Input:

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

Output:

Problem 6.4.4 (Fitting Elephants)

How many Elefants can you fit into a Volkswagen beetle?

and the actual solution in solutions mode:

Example 44

Input:

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

Output:

Problem 6.4.5 (Fitting Elephants)

How many Elefants can you fit into a Volkswagen beetle?

4!

Obviously, the argument of `\fillinsol` can be used for auto-grading. For concrete data like numbers, this is immediate, for more complex data like strings “soft comparisons” might be in order.⁷

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

⁷EDNOTE: For the moment we only assume a single concrete value as correct. In the future we will almost certainly want to extend the functionality to multiple answer classes that allow different feedback like in MCQ. This still needs a bit of design. Also we want to make the formatting of the answer in solutions/test mode configurable.

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

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

6.5.3 Assignments

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

6.5.4 Including Assignments

`\inputassignment`

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

6.5.5 Typesetting Exams

`\testspace`
`\testnewpage`
`\testemptypage`

`\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

`testheading`
`duration`
`min`
`reqpts`

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

```
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)

2022-07-28

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	6.4.4	6.4.5	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

EdN:8

8

⁸EDNOTE: 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 sTeX 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 XML
--------------------------	---

<code>\latexml_if_p: *</code>	L ^A T _E X3 conditionals for L ^A T _E XML.
<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_EXML 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</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> <code>\stex_path_to_string:N</code>	The inverse; turns a path into a string and stores it in the second argument variable, or 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> \star <code>\stex_path_if_absolute:N\underline{T}</code> \star	Checks whether the path provided is <i>absolute</i> , i.e. starts with an empty segment
--	---

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

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

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

8.1.2 MathHub Archives

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

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

\STEXreftitle**\STEXreftitle{<some title>}**

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**.

\stex_get_document_uri:

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.

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

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.

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

9.1.1 Setting Reference Targets

\stex_ref_new_doc_target:n**\stex_ref_new_doc_target:n{<id>}**

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

\stex_ref_new_sym_target:n**\stex_ref_new_sym_target:n{<uri>}**

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:

 $\backslash\text{c_stex_module_}\langle URI \rangle_prop$

A property list with the following fields:

name The *name* of the module,

ns the *namespace* in field **ns**,

file the *file* containing the module, as a sequence of path fragments

lang the module's *language*,

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.

 $\backslash\text{c_stex_module_}\langle URI \rangle_code$

The code to execute when this module is activated (i.e. imported), e.g. to set all the semantic macros, notations, etc.

 $\backslash\text{c_stex_module_}\langle URI \rangle_constants$

The names of all constants declared in the module

 $\backslash\text{c_stex_module_}\langle URI \rangle_constants$

The full URIs of all modules imported in this module

<hr/> <hr/> <code>\l_stex_current_module_str</code>	<code>\l_stex_current_module_str</code> always contains the URI of the current module (if existent).
<hr/> <hr/> <code>\l_stex_all_modules_seq</code>	Stores full URIs for all modules currently in scope.
<hr/> <hr/> <code>\stex_if_in_module_p: *</code> <code>\stex_if_in_module:TF *</code>	Conditional for whether we are currently in a module
<hr/> <hr/> <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.
<hr/> <hr/> <code>\stex_add_to_current_module:n</code> <code>\STEXexport</code>	Adds the provided tokens to the <code>_code</code> control sequence of the current module. <code>\stex_add_to_current_module:n</code> is used internally, <code>\STEXexport</code> is intended for users and additionally executes the provided code immediately.
<hr/> <hr/> <code>\stex_add_constant_to_current_module:n</code>	Adds the declaration with the provided name to the <code>_constants</code> control sequence of the current module.
<hr/> <hr/> <code>\stex_add_import_to_current_module:n</code>	Adds the module with the provided full URI to the <code>_imports</code> control sequence of the current module.
<hr/> <hr/> <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>
<hr/> <hr/> <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 `\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 `\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.

`\stex_activate_module:n`

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

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: *`
`\stex_if_smsmode: TF *`

Tests whether SMS mode is currently active.

<hr/> <hr/> <code>\stex_file_in_smsmode:nn</code>	<code>\stex_in_smsmode:nn</code> $\{\langle filename \rangle\}$ $\{\langle code \rangle\}$
	Executes $\langle code \rangle$ in SMS mode, followed by the content of $\langle filename \rangle$. $\langle code \rangle$ 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.

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

11.1.2 Imports and Inheritance

<hr/> <hr/> <code>\importmodule</code>	<code>\importmodule</code> $[\langle archive-ID \rangle]$ $\{\langle module-path \rangle\}$
	Imports a module by reading it from a file and “activating” it. $\text{\texttt{S\TeX}}$ determines the module and its containing file by passing its arguments on to <code>\stex_import_module_path:nn</code> .

<hr/> <hr/> <code>\usemodule</code>	<code>\importmodule</code> $[\langle archive-ID \rangle]$ $\{\langle module-path \rangle\}$
	Like <code>\importmodule</code> , but does not export its contents; i.e. including the current module will not activate the used module

`\stex_import_module_uri:nn`

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

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

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

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

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

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

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

2. Otherwise:

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

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

- (b) If $\langle path \rangle$ 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 `\stex_require_module:nn` on the `source` directory of the archive to find the file.

`\l_stex_import_name_str`
`\l_stex_import_archive_str`
`\l_stex_import_path_str`
`\l_stex_import_ns_str`

stores the result in these four variables.

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

Checks whether a module with URI $\langle ns \rangle?\langle name \rangle$ 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/> <code>\notation</code> <hr/>	$\text{\notation}[\langle args \rangle]\{\langle symbol \rangle\}\{\langle notations^+ \rangle\}$ <p>Introduces a new notation for $\langle symbol \rangle$, see <code>\stex_notation_do:nn</code></p>
<hr/> <hr/> <code>\stex_notation_do:nn</code>	$\text{\stex_notation_do:nn}\{\langle URI \rangle\}\{\langle notations^+ \rangle\}$ <p>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> <hr/>	$\text{\symdef}[\langle args \rangle]\{\langle symbol \rangle\}\{\langle notations^+ \rangle\}$ <p>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

<code>\STEXsymbol</code>	Uses <code>\stex_get_symbol:n</code> to find the symbol denoted by the first argument and passes the result on to <code>\stex_invoke_symbol:n</code>
--------------------------	--

<code>\symref</code>	<code>\symref{<symbol>}{<text>}</code> shortcut for <code>\STEXsymbol{<symbol>}! [<text>]</code>
----------------------	---

<code>\stex_invoke_symbol:n</code>	Executes a semantic macro. Outside of math mode or if followed by <code>*</code> , it continues to <code>\stex_term_custom:nn</code> . 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}`.

<code>\STEXInternalTermMathOMSiiii</code> <code>\STEXInternalTermMathOMAiiai</code> <code>\STEXInternalTermMathOMBiiii</code>	<code><URI><fragment><precedence><body></code>
---	--

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.

<code>\STEXInternalTermMathArgiii</code>	<code>\stex_term_arg:nnn<int><prec><body></code>
--	--

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

<hr/> <hr/>	<code>\STEXInternalTermMathAssocArgiiii</code>	<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/> <hr/>	<code>\infprec</code> <code>\neginfprec</code>	Maximal and minimal notation precedences.
<hr/> <hr/>	<code>\dobrackets</code>	<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/> <hr/>	<code>\withbrackets</code>	<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/> <hr/>	<code>\stex_term_custom:nn</code>	<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/> <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>	<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/> <hr/>	<code>\STEXinvisible</code>	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/> <hr/>	<code>\ellipses</code>	TODO

Chapter 14

ST_EX-Structural Features

Code related to structural features

14.1 Macros and Environments

14.1.1 Structures

`mathstructure` TODO

Chapter 15

sTeX-Statements

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

15.1 Macros and Environments

`symboldoc` `\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

sT_EX-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 68.)

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 68.)

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 68.)

`\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 68.)

`\stex_annotate:enx`

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

`\stex_annotate_invisible:n`

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

`\stex_annotate_invisible:nnn`

```

132 \ifcsname if@rustex\endcsname\else
133   \expandafter\newif\csname if@rustex\endcsname
134   \@rustexfalse
135 \fi
136 \ifcsname if@latexml\endcsname\else
137   \expandafter\newif\csname if@latexml\endcsname
138   \@latexmlfalse
139 \fi
140 \tl_if_exist:NF\stex@backend{
141   \if@rustex
142     \def\stex@backend{rustex}
143   \else
144     \if@latexml
145       \def\stex@backend{latexml}
146     \else

```

```

147     \cs_if_exist:NTF\HCode{
148         \def\stex@backend{tex4ht}
149     }{
150         \def\stex@backend{pdflatex}
151     }
152     \fi
153     \fi
154 }
155 \input{stex-backend-\stex@backend.cfg}
156
157 \newif\ifstexhtml
158 \stex_html_backend:TF\stexhtmltrue\stexhtmlfalse
159

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 69.)

23.5 Babel Languages

```

160 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

161 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_languages_prop { \tl_to_str:n {
162     en = english ,
163     de = ngerman ,
164     ar = arabic ,
165     bg = bulgarian ,
166     ru = russian ,
167     fi = finnish ,
168     ro = romanian ,
169     tr = turkish ,
170     fr = french
171 }}
172
173 \exp_args:NNx \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop { \tl_to_str:n {
174     english = en ,
175     ngerman = de ,
176     arabic = ar ,
177     bulgarian = bg ,
178     russian = ru ,
179     finnish = fi ,
180     romanian = ro ,
181     turkish = tr ,
182     french = fr
183 }}
184 % todo: chinese simplified (zhs)
185 %     chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 69.)

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

```

186 \cs_new_protected:Nn \stex_set_language:Nn {
187     \str_set:Nx \l_tmpa_str {#2}
188     \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {

```

```

189 \ifx\@onlypreamble\@notprerr
190 \ltx@ifpackageloaded{babel}{
191 \exp_args:No \selectlanguage #1
192 }{}
193 \else
194 \exp_args:No \str_if_eq:nnTF #1 {turkish} {
195 \RequirePackage[#1,shorthands=:!]{babel}
196 }{
197 \RequirePackage[#1]{babel}
198 }
199 \fi
200 }
201 }
202
203 \clist_if_empty:NF \c_stex_languages_clist {
204 \bool_set_false:N \l_tmpa_bool
205 \clist_clear:N \l_tmpa_clist
206 \clist_map_inline:Nn \c_stex_languages_clist {
207 \str_set:Nx \l_tmpa_str {#1}
208 \str_if_eq:nnT {#1}{tr}{
209 \bool_set_true:N \l_tmpa_bool
210 }
211 \prop_get:NoNTF \c_stex_languages_prop \l_tmpa_str \l_tmpa_str {
212 \clist_put_right:No \l_tmpa_clist \l_tmpa_str
213 } {
214 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
215 }
216 }
217 \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
218 \bool_if:NTF \l_tmpa_bool {
219 \RequirePackage[\clist_use:Nn \l_tmpa_clist,,shorthands=:!]{babel}
220 }{
221 \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
222 }
223 }
224
225 \AtBeginDocument{
226 \stex_html_backend:T {
227 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
228 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
229 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
230 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
231 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
232 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
233 \stex_debug:nn{basics} {Language~\l_tmpa_str~
234 inferred~from~file~name}
235 \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
236 }
237 }
238 }
239

```

23.6 Persistence

```

240 <@@=stex_persist>
241 \bool_if:NTF \c_stex_persist_mode_bool {
242   \def \stex_persist:n #1 {}
243   \def \stex_persist:x #1 {}
244 }{
245   \bool_if:NTF \c_stex_persist_write_mode_bool {
246     \iow_new:N \c__stex_persist_iow
247     \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
248     \AtEndDocument{
249       \iow_close:N \c__stex_persist_iow
250     }
251     \cs_new_protected:Nn \stex_persist:n {
252       \tl_set:Nn \l_tmpa_tl { #1 }
253       \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
254       \regex_replace_all:nnN { \ } { \~ } \l_tmpa_tl
255       \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
256     }
257     \cs_generate_variant:Nn \stex_persist:n {x}
258   }{
259     \def \stex_persist:n #1 {}
260     \def \stex_persist:x #1 {}
261   }
262 }

```

23.7 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

263 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
264   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
265   \def#1{
266     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
267   }
268 }

```

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

`\stex_reactivate_macro:N`

```

269 \cs_new_protected:Nn \stex_reactivate_macro:N {
270   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
271 }

```

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

`\ignorespacesandpars`

```

272 \protected\def\ignorespacesandpars{
273   \begingroup\catcode13=10\relax
274   \@ifnextchar\par{
275     \endgroup\expandafter\ignorespacesandpars\@gobble
276   }{
277     \endgroup
278   }
279 }

```

```

280
281 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
282   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
283   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
284   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
285
286   \tl_clear:N \_tmp_args_tl
287   \int_step_inline:nn \l_tmpa_int {
288     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
289   }
290
291   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
292   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
293     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
294     \exp_after:wN\exp_after:wN\exp_after:wN\exp_after:wN {
295       \exp_after:wN #2 \_tmp_args_tl
296     }
297   }}
298 }
299 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
300 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
301 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
302
303 \cs_new_protected:Nn \stex_copy_control_sequence_ii:NNN {
304   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
305   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
306   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
307
308   \tl_clear:N \_tmp_args_tl
309   \int_step_inline:nn \l_tmpa_int {
310     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{#####}\exp_not:n{##1}}}
311   }
312
313   \edef \_tmp_args_tl {
314     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
315     \exp_after:wN\exp_after:wN\exp_after:wN {
316       \exp_after:wN #2 \_tmp_args_tl
317     }
318   }
319
320   \exp_after:wN \def \exp_after:wN \_tmp_args_tl
321   \exp_after:wN ##\exp_after:wN 1 \exp_after:wN ##\exp_after:wN 2
322   \exp_after:wN { \_tmp_args_tl }
323
324   \edef \_tmp_args_tl {
325     \exp_after:wN \exp_not:n \exp_after:wN {
326       \_tmp_args_tl {####1}{####2}
327     }
328   }
329
330   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
331   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
332     \exp_after:wN\exp_not:n\exp_after:wN{\_tmp_args_tl}
333   }}

```

```

334 }
335
336 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {cNN}
337 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {NcN}
338 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {ccN}

```

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

\MMTrule

```

339 \NewDocumentCommand \MMTrule {m m}{
340   \seq_set_split:Nnn \l_tmpa_seq , {#2}
341   \int_zero:N \l_tmpa_int
342   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
343     \seq_if_empty:NF \l_tmpa_seq {
344       $\seq_map_inline:Nn \l_tmpa_seq {
345         \int_incr:N \l_tmpa_int
346         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
347       }$
348     }
349   }
350 }
351
352 \NewDocumentCommand \MMTinclude {m}{
353   \stex_annotate_invisible:nnn{import}{#1}{-}
354 }
355
356 \tl_new:N \g_stex_document_title
357 \cs_new_protected:Npn \STEXTtitle #1 {
358   \tl_if_empty:NT \g_stex_document_title {
359     \tl_gset:Nn \g_stex_document_title { #1 }
360   }
361 }
362 \cs_new_protected:Nn \stex_document_title:n {
363   \tl_if_empty:NT \g_stex_document_title {
364     \tl_gset:Nn \g_stex_document_title { #1 }
365     \stex_annotate_invisible:n{\noindent
366       \stex_annotate:nnn{doctitle}{-}{ #1 }
367     \par}
368   }
369 }
370 \AtBeginDocument {
371   \let \STEXTtitle \stex_document_title:n
372   \tl_if_empty:NF \g_stex_document_title {
373     \stex_annotate_invisible:n{\noindent
374       \stex_annotate:nnn{doctitle}{-}{ \g_stex_document_title }
375     \par}
376   }
377   \let \stex_maketitle:\maketitle
378   \def \maketitle{
379     \tl_if_empty:NF \@title {
380       \exp_args:No \stex_document_title:n \@title
381     }
382     \stex_maketitle:
383   }

```

```

384 }
385
386 \cs_new_protected:Nn \stex_par: {
387   \mode_if_vertical:F{
388     \if@minipage\else\if@nobreak\else\par\fi\fi
389   }
390 }
391
392 \</package>

```

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

Chapter 24

STEX -MathHub Implementation

```
393 <*package>
394
395 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
396
397 <@@=stex_path>
398
399 Warnings and error messages
400 \msg_new:nnn{stex}{error/norepository}{
401   No~archive~#1~found~in~#2
402 }
403 \msg_new:nnn{stex}{error/notinarchive}{
404   Not~currently~in~an~archive,~but~\detokenize{#1}~
405   needs~one!
406 }
407 \msg_new:nnn{stex}{error/nofile}{
408   \detokenize{#1}~could~not~find~file~#2
409 }
410 \msg_new:nnn{stex}{error/twofiles}{
411   \detokenize{#1}~found~two~candidates~for~#2
412 }
```

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`

```
411 \cs_new_protected:Nn \stex_path_from_string:Nn {
412   \str_set:Nx \l_tmpa_str { #2 }
413   \str_if_empty:NTF \l_tmpa_str {
414     \seq_clear:N #1
415   }{
416     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
417     \sys_if_platform_windows:T{
418       \seq_clear:N \l_tmpa_tl
```

```

419     \seq_map_inline:Nn #1 {
420       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
421       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
422     }
423     \seq_set_eq:NN #1 \l_tmpa_tl
424   }
425   \stex_path_canonicalize:N #1
426 }
427 }
428

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 70.)

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

```

429 \cs_new_protected:Nn \stex_path_to_string:NN {
430   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
431 }
432
433 \cs_new:Nn \stex_path_to_string:N {
434   \seq_use:Nn #1 /
435 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 70.)

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

```

436 \str_const:Nn \c__stex_path_dot_str {.}
437 \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.

```

438 \cs_new_protected:Nn \stex_path_canonicalize:N {
439   \seq_if_empty:NF #1 {
440     \seq_clear:N \l_tmpa_seq
441     \seq_get_left:NN #1 \l_tmpa_tl
442     \str_if_empty:NT \l_tmpa_tl {
443       \seq_put_right:Nn \l_tmpa_seq {}
444     }
445     \seq_map_inline:Nn #1 {
446       \str_set:Nn \l_tmpa_tl { ##1 }
447       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
448         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
449           \seq_if_empty:NNTF \l_tmpa_seq {
450             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
451               \c__stex_path_up_str
452             }
453           }{
454             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
455             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
456               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
457                 \c__stex_path_up_str
458               }
459             }{

```

```

460         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
461     }
462 }
463 }{
464     \str_if_empty:NF \l_tmpa_tl {
465         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
466     }
467 }
468 }
469 }
470 \seq_gset_eq:NN #1 \l_tmpa_seq
471 }
472 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 70.)

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

```

473 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
474     \seq_if_empty:NTF #1 {
475         \prg_return_false:
476     }{
477         \seq_get_left:NN #1 \l_tmpa_tl
478         \sys_if_platform_windows:TF{
479             \str_if_in:NnTF \l_tmpa_tl {:}{
480                 \prg_return_true:
481             }{
482                 \prg_return_false:
483             }
484         }{
485             \str_if_empty:NTF \l_tmpa_tl {
486                 \prg_return_true:
487             }{
488                 \prg_return_false:
489             }
490         }
491     }
492 }

```

(End definition for `\stex_path_if_absolute:N \underline{TF}` . This function is documented on page 70.)

24.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

493 \str_new:N\l_stex_kpsewhich_return_str
494 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
495     \catcode'\ =12
496     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
497     \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
498     \endgroup
499     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
500     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
501 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 70.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

502 \sys_if_platform_windows:TF{
503   \begingroup\escapechar=-1\catcode'\=12
504   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
505   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
506   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
507 }{
508   \stex_kpsewhich:n{-var-value~PWD}
509 }
510
511 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
512 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
513 \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 70.)

24.3 File Hooks and Tracking

514 `<@=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

```

515 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

516 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
517 \stex_path_from_string:Nn \c_stex_mainfile_seq
518   \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 70.)

`\g_stex_currentfile_seq`

```

519 \seq_gclear_new:N\g_stex_currentfile_seq

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 71.)

`\stex_filestack_push:n`

```

520 \cs_new_protected:Nn \stex_filestack_push:n {
521   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
522   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
523     \stex_path_from_string:Nn\g_stex_currentfile_seq{
524       \c_stex_pwd_str/#1
525     }

```

```

526 }
527 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
528 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
529 }

```

(End definition for `\stex_filestack_push:n`. This function is documented on page 71.)

`\stex_filestack_pop:`

```

530 \cs_new_protected:Nn \stex_filestack_pop: {
531   \seq_if_empty:NF\g__stex_files_stack{
532     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
533   }
534   \seq_if_empty:NTF\g__stex_files_stack{
535     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
536   }{
537     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
538     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
539   }
540 }

```

(End definition for `\stex_filestack_pop:`. This function is documented on page 71.)

Hooks for the current file:

```

541 \AddToHook{file/before}{
542   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
543 }
544 \AddToHook{file/after}{
545   \stex_filestack_pop:
546 }

```

24.4 MathHub Repositories

```

547 <@=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`

```

548 \str_if_empty:NTF\mathhub{
549   \sys_if_platform_windows:TF{
550     \begingroup\escapechar=-1\catcode'\=12
551     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
552     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
553     \exp_args:NNx\str_if_eq:ont\l_stex_kpsewhich_return_str{\c_percent_str MATHHUB\c_percent_str}
554     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
555   }{
556     \stex_kpsewhich:n{-var-value-MATHHUB}
557   }
558   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
559 }
560 \str_if_empty:NT \c_stex_mathhub_str {
561   \sys_if_platform_windows:TF{
562     \begingroup\escapechar=-1\catcode'\=12
563     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
564     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
565     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}

```

```

566   }{
567     \stex_kpsewhich:n{-var-value-HOME}
568   }
569   \ior_open:NnT \g_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
570     \begingroup\escapechar=-1\catcode'\=12
571     \ior_str_get:NN \g_tmpa_ior \l_tmpa_str
572     \sys_if_platform_windows:T{
573       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
574     }
575     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
576     \endgroup
577     \ior_close:N \g_tmpa_ior
578   }
579 }
580 \str_if_empty:NTF\c_stex_mathhub_str{
581   \msg_warning:nn{stex}{warning/nomathhub}
582 }{
583   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
584   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
585 }
586 }{
587   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
588   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
589     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
590       \c_stex_pwd_str/\mathhub
591     }
592   }
593   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
594   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
595 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 71.)

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

```

596 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
597   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
598     \str_set:Nx \l_tmpa_str { #1 }
599     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
600     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
601     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
602     \_stex_mathhub_find_manifest:N \l_tmpa_seq
603     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
604       \msg_error:nnxx{stex}{error/norepository}{#1}{
605         \stex_path_to_string:N \c_stex_mathhub_str
606       }
607       \input{Fatal-Error!}
608     } {
609       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
610     }
611   }
612 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

\l_stex_mathhub_manifest_file_seq

613 \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:

```

614 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
615   \seq_set_eq:NN\l_tmpa_seq #1
616   \bool_set_true:N\l_tmpa_bool
617   \bool_while_do:Nn \l_tmpa_bool {
618     \seq_if_empty:NTF \l_tmpa_seq {
619       \bool_set_false:N\l_tmpa_bool
620     }{
621       \file_if_exist:nTF{
622         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
623       }{
624         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
625         \bool_set_false:N\l_tmpa_bool
626       }{
627         \file_if_exist:nTF{
628           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
629         }{
630           \seq_put_right:Nn\l_tmpa_seq{META-INF}
631           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
632           \bool_set_false:N\l_tmpa_bool
633         }{
634           \file_if_exist:nTF{
635             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
636           }{
637             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
638             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
639             \bool_set_false:N\l_tmpa_bool
640           }{
641             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
642           }
643         }
644       }
645     }
646   }
647   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
648 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior

File variable used for MANIFEST-files

649 \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:

```

650 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
651   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
652   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}

```

```

653 \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
654   \str_set:Nn \l_tmpa_str {##1}
655   \exp_args:NNoo \seq_set_split:Nnn
656     \l_tmpb_seq \c_colon_str \l_tmpa_str
657   \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
658     \exp_args:NNe \str_set:Nn \l_tmpb_tl {
659       \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
660     }
661     \exp_args:No \str_case:nnTF \l_tmpa_tl {
662       {id} {
663         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
664           { id } \l_tmpb_tl
665       }
666       {narration-base} {
667         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
668           { narr } \l_tmpb_tl
669       }
670       {url-base} {
671         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
672           { docurl } \l_tmpb_tl
673       }
674       {source-base} {
675         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
676           { ns } \l_tmpb_tl
677       }
678       {ns} {
679         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
680           { ns } \l_tmpb_tl
681       }
682       {dependencies} {
683         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
684           { deps } \l_tmpb_tl
685       }
686     }{}{}
687   }{}
688 }
689 \ior_close:N \c__stex_mathhub_manifest_ior
690 \stex_persist:x {
691   \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
692     \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
693   }
694 }
695 }

```

(End definition for _stex_mathhub_parse_manifest:n.)

\stex_set_current_repository:n

```

696 \cs_new_protected:Nn \stex_set_current_repository:n {
697   \stex_require_repository:n { #1 }
698   \prop_set_eq:Nc \l_stex_current_repository_prop {
699     c_stex_mathhub_#1_manifest_prop
700   }
701 }

```

(End definition for \stex_set_current_repository:n. This function is documented on page 71.)

`\stex_require_repository:n`

```

702 \cs_new_protected:Nn \stex_require_repository:n {
703   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
704     \stex_debug:nn{mathhub}{Opening~archive:~#1}
705     \__stex_mathhub_do_manifest:n { #1 }
706   }
707 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 71.)

`\l_stex_current_repository_prop` Current MathHub repository

```

708 %\prop_new:N \l_stex_current_repository_prop
709 \bool_if:NF \c_stex_persist_mode_bool {
710   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
711   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
712     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
713   } {
714     \__stex_mathhub_parse_manifest:n { main }
715     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
716     \l_tmpa_str
717     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
718     \c_stex_mathhub_main_manifest_prop
719     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
720     \stex_debug:nn{mathhub}{Current~repository:~
721     \prop_item:Nn \l_stex_current_repository_prop {id}
722   }
723 }
724 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 71.)

`\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.

```

725 \cs_new_protected:Nn \stex_in_repository:nn {
726   \str_set:Nx \l_tmpa_str { #1 }
727   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
728   \str_if_empty:NTF \l_tmpa_str {
729     \prop_if_exist:NTF \l_stex_current_repository_prop {
730       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
731       \exp_args:Ne \l_tmpa_cs{
732         \prop_item:Nn \l_stex_current_repository_prop { id }
733       }
734     }{
735       \l_tmpa_cs{}
736     }
737   }{
738     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
739     \stex_require_repository:n \l_tmpa_str
740     \str_set:Nx \l_tmpa_str { #1 }
741     \exp_args:Nne \use:nn {
742       \stex_set_current_repository:n \l_tmpa_str
743       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
744     }{
745       \stex_debug:nn{mathhub}{switching~back~to:~

```

```

746     \prop_if_exist:NTF \l_stex_current_repository_prop {
747       \prop_item:Nn \l_stex_current_repository_prop { id } :~
748       \meaning\l_stex_current_repository_prop
749     }{
750       no~repository
751     }
752   }
753   \prop_if_exist:NTF \l_stex_current_repository_prop {
754     \stex_set_current_repository:n {
755       \prop_item:Nn \l_stex_current_repository_prop { id }
756     }
757   }{
758     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
759   }
760 }
761 }
762 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 71.)

24.5 Using Content in Archives

`\mhpath`

```

763 \def \mhpath #1 #2 {
764   \exp_args:Ne \tl_if_empty:nTF{#1}{
765     \c_stex_mathhub_str /
766     \prop_item:Nn \l_stex_current_repository_prop { id }
767     / source / #2
768   }{
769     \c_stex_mathhub_str / #1 / source / #2
770   }
771 }

```

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

`\inputref`

`\mhinput`

```

772 \newif \ifinputref \inputreffalse
773
774 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
775   \stex_in_repository:nn {#1} {
776     \ifinputref
777       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
778     \else
779       \inputreftrue
780       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
781       \inputreffalse
782     \fi
783   }
784 }
785 \NewDocumentCommand \mhinput { 0{} m }{
786   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
787 }
788

```

```

789 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
790   \stex_in_repository:nn {#1} {
791     \stex_html_backend:TF {
792       \str_clear:N \l_tmpa_str
793       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
794         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
795       }
796
797       \tl_if_empty:nTF{ ##1 }{
798         \IfFileExists{#2}{
799           \stex_annotate_invisible:nnn{inputref}{
800             \l_tmpa_str / #2
801           }{}
802         }{
803           \input{#2}
804         }
805       }{
806         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
807           \stex_annotate_invisible:nnn{inputref}{
808             \l_tmpa_str / #2
809           }{}
810         }{
811           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
812         }
813       }
814
815     }{
816       \begingroup
817       \inputreftrue
818       \tl_if_empty:nTF{ ##1 }{
819         \input{#2}
820       }{
821         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
822       }
823       \endgroup
824     }
825   }
826 }
827 \NewDocumentCommand \inputref { 0{} m}{
828   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
829 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 72.)

`\addmhbibresource`

```

830 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
831   \stex_in_repository:nn {#1} {
832     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
833   }
834 }
835 \newcommand\addmhbibresource[2][{}]{
836   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
837 }

```

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

`\libinput`

```
838 \cs_new_protected:Npn \libinput #1 {
839   \prop_if_exist:NF \l_stex_current_repository_prop {
840     \msg_error:nnn{stex}{error/notinarchive}\libinput
841   }
842   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
843     \msg_error:nnn{stex}{error/notinarchive}\libinput
844   }
845   \seq_clear:N \l__stex_mathhub_libinput_files_seq
846   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
847   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
848
849   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
850     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
851     \IfFileExists{ \l_tmpa_str }{
852       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
853     }{}
854     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
855     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
856   }
857
858   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
859   \IfFileExists{ \l_tmpa_str }{
860     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
861   }{}
862
863   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
864     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
865   }{
866     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
867       \input{ ##1 }
868     }
869   }
870 }
```

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

`\libusepackage`

```
871 \NewDocumentCommand \libusepackage {0{ } m} {
872   \prop_if_exist:NF \l_stex_current_repository_prop {
873     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
874   }
875   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
876     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
877   }
878   \seq_clear:N \l__stex_mathhub_libinput_files_seq
879   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
880   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
881
882   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
883     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
884     \IfFileExists{ \l_tmpa_str.sty }{
885       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
886     }{}
887   }
```

```

887 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
888 \seq_put_right:No \l_tmpa_seq \l_tmpa_str
889 }
890
891 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
892 \IfFileExists{ \l_tmpa_str.sty }{
893   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
894 }{}
895
896 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
897   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
898 }{
899   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
900     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
901       \usepackage[#1]{ ##1 }
902     }
903   }{
904     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
905   }
906 }
907 }

```

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

`\mhgraphics`
`\cmhgraphics`

```

908
909 \AddToHook{begindocument}{
910 \ltx@ifpackageloaded{graphicx}{
911   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
912   \providecommand\mhgraphics[2] [] {%
913     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
914     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
915   \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
916 }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 72.)

`\lstinputmhlisting`
`\clstinputmhlisting`

```

917 \ltx@ifpackageloaded{listings}{
918   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
919   \newcommand\lstinputmhlisting[2] [] {%
920     \def\lst@mhrepos{}\setkeys{lst}{#1}%
921     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
922   \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
923 }{}
924 }
925
926 </package>

```

(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 72.)

Chapter 25

STEX -References Implementation

```
927 <*package>
928
929 %%%%%%%%%%%%% stex-references.dtx %%%%%%%%%%%%%
930
931 <@@=stex_refs>
    Warnings and error messages
932
```

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

```
933 %\iow_new:N \c__stex_refs_refs_iow
934 \AtBeginDocument{
935 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
936 }
937 \AtEndDocument{
938 % \iow_close:N \c__stex_refs_refs_iow
939 }
```

`\STEXreftitle`

```
940 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
941
942 \NewDocumentCommand \STEXreftitle { m } {
943   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
944 }
```

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

25.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
945 \str_new:N \l_stex_current_docns_str
```

(End definition for `\l_stex_current_docns_str`. This variable is documented on page 73.)

`\stex_get_document_uri:`

```
946 \cs_new_protected:Nn \stex_get_document_uri: {
947   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
948   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
949   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
950   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
951   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
952
953   \str_clear:N \l_tmpa_str
954   \prop_if_exist:NT \l_stex_current_repository_prop {
955     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
956       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
957     }
958   }
959
960   \str_if_empty:NTF \l_tmpa_str {
961     \str_set:Nx \l_stex_current_docns_str {
962       file:/\stex_path_to_string:N \l_tmpa_seq
963     }
964   }{
965     \bool_set_true:N \l_tmpa_bool
966     \bool_while_do:Nn \l_tmpa_bool {
967       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
968       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
969         {source} { \bool_set_false:N \l_tmpa_bool }
970       }{}{
971         \seq_if_empty:NT \l_tmpa_seq {
972           \bool_set_false:N \l_tmpa_bool
973         }
974       }
975     }
976
977     \seq_if_empty:NTF \l_tmpa_seq {
978       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
979     }{
980       \str_set:Nx \l_stex_current_docns_str {
981         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
982       }
983     }
984   }
985 }
```

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

`\l_stex_current_docurl_str`

```
986 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 73.)

`\stex_get_document_url:`

```
987 \cs_new_protected:Nn \stex_get_document_url: {
988   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
989   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
990   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

991 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
992 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
993
994 \str_clear:N \l_tmpa_str
995 \prop_if_exist:NT \l_stex_current_repository_prop {
996   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
997     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
998       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
999     }
1000   }
1001 }
1002
1003 \str_if_empty:NTF \l_tmpa_str {
1004   \str_set:Nx \l_stex_current_docurl_str {
1005     file:/\stex_path_to_string:N \l_tmpa_seq
1006   }
1007 }{
1008   \bool_set_true:N \l_tmpa_bool
1009   \bool_while_do:Nn \l_tmpa_bool {
1010     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1011     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1012       {source} { \bool_set_false:N \l_tmpa_bool }
1013     }{}{
1014       \seq_if_empty:NT \l_tmpa_seq {
1015         \bool_set_false:N \l_tmpa_bool
1016       }
1017     }
1018   }
1019 }
1020 \seq_if_empty:NTF \l_tmpa_seq {
1021   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1022 }{
1023   \str_set:Nx \l_stex_current_docurl_str {
1024     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1025   }
1026 }
1027 }
1028 }

```

(End definition for `\stex_get_document_url`:. This function is documented on page [73](#).)

25.2 Setting Reference Targets

```

1029 \str_const:Nn \c__stex_refs_url_str{URL}
1030 \str_const:Nn \c__stex_refs_ref_str{REF}
1031 \str_new:N \l__stex_refs_curr_label_str
1032 % @currentlabel -> number
1033 % @currentlabelname -> title
1034 % @currentHref -> name.number <- id of some kind
1035 % \theH# -> \arabic{section}
1036 % \the# -> number
1037 % \hyper@makecurrent{#}
1038 \int_new:N \l__stex_refs_unnamed_counter_int

```


`\stex_ref_new_doc_target:n`

```

1039 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1040   \stex_get_document_uri:
1041   \str_clear:N \l__stex_refs_curr_label_str
1042   \str_set:Nx \l_tmpa_str { #1 }
1043   \str_if_empty:NT \l_tmpa_str {
1044     \int_incr:N \l__stex_refs_unnamed_counter_int
1045     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1046   }
1047   \str_set:Nx \l__stex_refs_curr_label_str {
1048     \l_stex_current_docns_str?\l_tmpa_str
1049   }
1050   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1051     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1052   }
1053   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1054     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1055   }
1056   \stex_if_smsmode:TF {
1057     \stex_get_document_url:
1058     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1059     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1060   }{
1061     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1062     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1063     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1064     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1065   }
1066 }

```

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

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

```

1067 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1068   \str_set:Nn \l_tmpa_str {#1?#2}
1069   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1070   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1071     \seq_new:c {g__stex_refs_labels_#2_seq}
1072   }
1073   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1074     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1075   }
1076 }

```

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

```

1077 \AtEndDocument{
1078   \def\stexauxadddocref#1 #2 {}{}
1079 }

```

`\stex_ref_new_sym_target:n`

```

1080 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1081   \stex_if_smsmode:TF {
1082     \str_if_exist:cF{sref_sym_#1_type}{
1083       \stex_get_document_url:
1084       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

1085     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1086   }
1087 }{
1088   \str_if_empty:NF \l__stex_refs_curr_label_str {
1089     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1090     \immediate\write\@auxout{
1091       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1092         \l__stex_refs_curr_label_str
1093       }
1094     }
1095   }
1096 }
1097 }

```

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

25.3 Using References

```

1098 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

1099
1100 \keys_define:nn { stex / sref } {
1101   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1102   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1103   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1104   post          .tl_set:N = \l__stex_refs_post_tl ,
1105 }
1106 \cs_new_protected:Nn \__stex_refs_args:n {
1107   \tl_clear:N \l__stex_refs_linktext_tl
1108   \tl_clear:N \l__stex_refs_fallback_tl
1109   \tl_clear:N \l__stex_refs_pre_tl
1110   \tl_clear:N \l__stex_refs_post_tl
1111   \str_clear:N \l__stex_refs_repo_str
1112   \keys_set:nn { stex / sref } { #1 }
1113 }

```

The actual macro:

```

1114 \NewDocumentCommand \sref { 0{} m}{
1115   \__stex_refs_args:n { #1 }
1116   \str_if_empty:NTF \l__stex_refs_indocument_str {
1117     \str_set:Nx \l_tmpa_str { #2 }
1118     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1119     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1120       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1121         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1122           \str_clear:N \l_tmpa_str
1123         }
1124       }{
1125         \str_clear:N \l_tmpa_str
1126       }
1127     }{
1128       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1129       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1130 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1131 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1132   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1133   \str_clear:N \l_tmpa_str
1134   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1135     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1136       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1137     }{
1138       \seq_map_break:n {
1139         \str_set:Nn \l_tmpa_str { ##1 }
1140       }
1141     }
1142   }
1143 }{
1144   \str_clear:N \l_tmpa_str
1145 }
1146 }
1147 \str_if_empty:NTF \l_tmpa_str {
1148   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1149 }{
1150   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1151     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1152       \cs_if_exist:cTF{autoref}{
1153         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1154       }{
1155         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1156       }
1157     }{
1158       \ltx@ifpackageloaded{hyperref}{
1159         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1160       }{
1161         \l__stex_refs_linktext_tl
1162       }
1163     }
1164   }{
1165     \ltx@ifpackageloaded{hyperref}{
1166       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1167     }{
1168       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1169     }
1170   }
1171 }
1172 }{
1173   % TODO
1174 }
1175 }

```

(End definition for `\sref`. This function is documented on page 74.)

`\srefsym`

```

1176 \NewDocumentCommand \srefsym { 0{} m}{
1177   \stex_get_symbol:n { #2 }
1178   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1179 }

```

```

1180
1181 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1182   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1183     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1184   }{
1185     \__stex_refs_args:n { #1 }
1186     \str_if_empty:NTF \l__stex_refs_indocument_str {
1187       \tl_if_exist:cTF{sref_sym_#2 _type}{
1188         % doc uri in \l_tmpb_str
1189         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1190         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1191           % reference
1192           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1193             \cs_if_exist:cTF{autoref}{
1194               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1195             }{
1196               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1197             }
1198           }{
1199             \ltx@ifpackageloaded{hyperref}{
1200               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1201             }{
1202               \l__stex_refs_linktext_tl
1203             }
1204           }
1205         }{
1206           % URL
1207           \ltx@ifpackageloaded{hyperref}{
1208             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1209           }{
1210             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1211           }
1212         }
1213       }{
1214         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1215       }
1216     }{
1217       % TODO
1218     }
1219   }
1220 }

```

(End definition for \srefsym. This function is documented on page 74.)

\srefsymuri

```

1221 \cs_new_protected:Npn \srefsymuri #1 #2 {
1222   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1223 }

```

(End definition for \srefsymuri. This function is documented on page 74.)

```

1224 </package>

```

Chapter 26

STEX -Modules Implementation

```
1225 <*package>
1226
1227 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1228
1229 <@@=stex_modules>
1230
1231 Warnings and error messages
1232 \msg_new:nnn{stex}{error/unknownmodule}{
1233   No~module~#1~found
1234 }
1235 \msg_new:nnn{stex}{error/syntax}{
1236   Syntax~error:~#1
1237 }
1238 \msg_new:nnn{stex}{error/siglanguage}{
1239   Module~#1~declares~signature~#2,~but~does~not~
1240   declare~its~language
1241 }
1242 \msg_new:nnn{stex}{warning/deprecated}{
1243   #1~is~deprecated;~please~use~#2~instead!
1244 }
1245 \msg_new:nnn{stex}{error/conflictingmodules}{
1246   Conflicting~imports~for~module~#1
1247 }
```

`\l_stex_current_module_str` The current module:

```
1247 \str_new:N \l_stex_current_module_str
```

(End definition for `\l_stex_current_module_str`. This variable is documented on page 76.)

`\l_stex_all_modules_seq` Stores all available modules

```
1248 \seq_new:N \l_stex_all_modules_seq
```

(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 76.)

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1249 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1250   \str_if_empty:NTF \l_stex_current_module_str
1251   \prg_return_false: \prg_return_true:
1252 }

```

(End definition for `\stex_if_in_module:TF`. This function is documented on page 76.)

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1253 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1254   \prop_if_exist:cTF { c_stex_module_#1_prop }
1255   \prg_return_true: \prg_return_false:
1256 }

```

(End definition for `\stex_if_module_exists:nTF`. This function is documented on page 76.)

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1257 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1258   \stex_add_to_current_module:n { #1 }
1259   \stex_do_up_to_module:n { #1 }
1260 }}
1261 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1262
1263 \cs_new_protected:Nn \stex_add_to_current_module:n {
1264   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1265 }
1266 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1267 \cs_new_protected:Npn \STEXexport {
1268   \ExplSyntaxOn
1269   \__stex_modules_export:n
1270 }
1271 \cs_new_protected:Nn \__stex_modules_export:n {
1272   \ignorespacesandpars#1\ExplSyntaxOff
1273   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1274   \stex_smsmode_do:
1275 }
1276 \let \stex_module_export_helper:n \use:n
1277 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 76.)

```

\stex_add_constant_to_current_module:n
1278 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1279   \str_set:Nx \l_tmpa_str { #1 }
1280   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1281 }

```

(End definition for `\stex_add_constant_to_current_module:n`. This function is documented on page 76.)

```

\stex_add_import_to_current_module:n
1282 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1283   \str_set:Nx \l_tmpa_str { #1 }
1284   \exp_args:Nno

```

```

1285 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1286 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1287 }
1288 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 76.)

`\stex_collect_imports:n`

```

1289 \cs_new_protected:Nn \stex_collect_imports:n {
1290 \seq_clear:N \l_stex_collect_imports_seq
1291 \__stex_modules_collect_imports:n {#1}
1292 }
1293 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1294 \seq_map_inline:cn {c_stex_module_#1_imports} {
1295 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1296 \__stex_modules_collect_imports:n { ##1 }
1297 }
1298 }
1299 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1300 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1301 }
1302 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 76.)

`\stex_do_up_to_module:n`

```

1303 \int_new:N \l__stex_modules_group_depth_int
1304 \cs_new_protected:Nn \stex_do_up_to_module:n {
1305 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1306 #1
1307 }{
1308 #1
1309 \expandafter \tl_gset:Nn
1310 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1311 \expandafter\expandafter\expandafter\endcsname
1312 \expandafter\expandafter\expandafter { \csname
1313 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1314 \aftergroup\__stex_modules_aftergroup_do:
1315 }
1316 }
1317 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1318 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1319 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1320 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1321 }}}
1322 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1323 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1324 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1325 }{
1326 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1327 \aftergroup\__stex_modules_aftergroup_do:
1328 }
1329 }
1330 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1331 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1332 }

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 76.)

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

1333

(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.

```
1334 \str_new:N \l_stex_module_ns_str
1335 \str_new:N \l_stex_module_subpath_str
1336 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1337   \seq_set_eq:NN \l_tmpa_seq #2
1338   % split off file extension
1339   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1340   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1341   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1342   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1343
1344   \bool_set_true:N \l_tmpa_bool
1345   \bool_while_do:Nn \l_tmpa_bool {
1346     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1347     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1348       {source} { \bool_set_false:N \l_tmpa_bool }
1349     }{}{
1350       \seq_if_empty:NT \l_tmpa_seq {
1351         \bool_set_false:N \l_tmpa_bool
1352       }
1353     }
1354   }
1355
1356   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1357   % \l_tmpa_seq <- sub-path relative to archive
1358   \str_if_empty:NTF \l_stex_module_subpath_str {
1359     \str_set:Nx \l_stex_module_ns_str {#1}
1360   }{
1361     \str_set:Nx \l_stex_module_ns_str {
1362       #1/\l_stex_module_subpath_str
1363     }
1364   }
1365 }
1366
1367 \cs_new_protected:Nn \stex_modules_current_namespace: {
1368   \str_clear:N \l_stex_module_subpath_str
1369   \prop_if_exist:NTF \l_stex_current_repository_prop {
1370     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1371     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1372   }{
1373     % split off file extension
1374     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1375     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```



```

1376 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1377 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1378 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1379 \str_set:Nx \l_stex_module_ns_str {
1380   file:/\stex_path_to_string:N \l_tmpa_seq
1381 }
1382 }
1383 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 77.)

26.1 The smodule environment

smodule arguments:

```

1384 \keys_define:nn { stex / module } {
1385   title      .tl_set:N      = \smodulename ,
1386   type       .str_set_x:N   = \smodulename ,
1387   id         .str_set_x:N   = \smoduleid ,
1388   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1389   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1390   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1391   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1392   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1393   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1394   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1395   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1396 }
1397
1398 \cs_new_protected:Nn \__stex_modules_args:n {
1399   \str_clear:N \smodulename
1400   \str_clear:N \smodulename
1401   \str_clear:N \smoduleid
1402   \str_clear:N \l_stex_module_ns_str
1403   \str_clear:N \l_stex_module_deprecate_str
1404   \str_clear:N \l_stex_module_lang_str
1405   \str_clear:N \l_stex_module_sig_str
1406   \str_clear:N \l_stex_module_creators_str
1407   \str_clear:N \l_stex_module_contributors_str
1408   \str_clear:N \l_stex_module_meta_str
1409   \str_clear:N \l_stex_module_srccite_str
1410   \keys_set:nn { stex / module } { #1 }
1411 }
1412
1413 % module parameters here? In the body?
1414

```

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

```

1415 \cs_new_protected:Nn \stex_module_setup:nn {
1416   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1417   \str_set:Nx \l_stex_module_name_str { #2 }
1418   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1419 \stex_if_in_module:TF {
1420   % Nested module
1421   \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1422   { ns } \l_stex_module_ns_str
1423   \str_set:Nx \l_stex_module_name_str {
1424     \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1425     { name } / \l_stex_module_name_str
1426   }
1427   \str_if_empty:NT \l_stex_module_lang_str {
1428     \str_set:Nx \l_stex_module_lang_str {
1429       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1430       { lang }
1431     }
1432   }
1433 }{
1434   % not nested:
1435   \str_if_empty:NT \l_stex_module_ns_str {
1436     \stex_modules_current_namespace:
1437     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1438       / {\l_stex_module_ns_str}
1439     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1440     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1441       \str_set:Nx \l_stex_module_ns_str {
1442         \stex_path_to_string:N \l_tmpa_seq
1443       }
1444     }
1445   }
1446 }

```

Next, we determine the language of the module:

```

1447 \str_if_empty:NT \l_stex_module_lang_str {
1448   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1449   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1450   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1451   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1452     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1453       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1454     }
1455   }
1456   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1457   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1458     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1459     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1460       inferred~from~file~name}
1461   }
1462 }
1463
1464 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1465   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1466 }}

```

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

```

1467 \str_if_empty:NTF \l_stex_module_sig_str {
1468   \exp_args:Nnx \prop_gset_from_keyval:cn {
1469     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1470   } {
1471     name      = \l_stex_module_name_str ,
1472     ns        = \l_stex_module_ns_str ,
1473     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1474     lang      = \l_stex_module_lang_str ,
1475     sig       = \l_stex_module_sig_str ,
1476     deprecate = \l_stex_module_deprecate_str ,
1477     meta      = \l_stex_module_meta_str
1478   }
1479   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1480   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1481   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1482   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1483   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1484 \str_if_empty:NT \l_stex_module_meta_str {
1485   \str_set:Nx \l_stex_module_meta_str {
1486     \c_stex_metatheory_ns_str ? Metatheory
1487   }
1488 }
1489 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1490   \bool_set_true:N \l_stex_in_meta_bool
1491   \exp_args:Nx \stex_add_to_current_module:n {
1492     \bool_set_true:N \l_stex_in_meta_bool
1493     \stex_activate_module:n {\l_stex_module_meta_str}
1494     \bool_set_false:N \l_stex_in_meta_bool
1495   }
1496   \stex_activate_module:n {\l_stex_module_meta_str}
1497   \bool_set_false:N \l_stex_in_meta_bool
1498 }
1499 }{
1500   \str_if_empty:NT \l_stex_module_lang_str {
1501     \msg_error:nnxx{stex}{error/siglanguage}{
1502       \l_stex_module_ns_str?\l_stex_module_name_str
1503     }{\l_stex_module_sig_str}
1504   }
1505   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1506   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1507     \stex_debug:nn{modules}{(already exists)}
1508   }{
1509     \stex_debug:nn{modules}{(needs loading)}
1510     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1511     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1512     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1513     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1514     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1515     \str_set:Nx \l_tmpa_str {
1516       \stex_path_to_string:N \l_tmpa_seq /

```

```

1517     \l_tmpa_str . \l_stex_module_sig_str .tex
1518 }
1519 \IfFileExists \l_tmpa_str {
1520     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1521         \str_clear:N \l_stex_current_module_str
1522         \seq_clear:N \l_stex_all_modules_seq
1523         \stex_debug:nn{modules}{Loading~signature}
1524     }
1525 }{
1526     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1527 }
1528 }
1529 \stex_if_smsmode:F {
1530     \stex_activate_module:n {
1531         \l_stex_module_ns_str ? \l_stex_module_name_str
1532     }
1533 }
1534 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1535 }
1536 \str_if_empty:NF \l_stex_module_deprecate_str {
1537     \msg_warning:nnxx{stex}{warning/deprecated}{
1538         Module~\l_stex_current_module_str
1539     }{
1540         \l_stex_module_deprecate_str
1541     }
1542 }
1543 \seq_put_right:Nx \l_stex_all_modules_seq {
1544     \l_stex_module_ns_str ? \l_stex_module_name_str
1545 }
1546 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1547 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 77.)

smodule The module environment.

`_stex_modules_begin_module:` implements `\begin{smodule}`

```

1548 \cs_new_protected:Nn \_stex_modules_begin_module: {
1549     \stex_reactivate_macro:N \STEXexport
1550     \stex_reactivate_macro:N \importmodule
1551     \stex_reactivate_macro:N \symdecl
1552     \stex_reactivate_macro:N \notation
1553     \stex_reactivate_macro:N \symdef
1554
1555     \stex_debug:nn{modules}{
1556         New~module:\\
1557         Namespace:~\l_stex_module_ns_str\\
1558         Name:~\l_stex_module_name_str\\
1559         Language:~\l_stex_module_lang_str\\
1560         Signature:~\l_stex_module_sig_str\\
1561         Metatheory:~\l_stex_module_meta_str\\
1562         File:~\stex_path_to_string:N \g_stex_currentfile_seq
1563     }
1564

```

```

1565 \stex_if_do_html:T{
1566   \begin{stex_annotate_env} {theory} {
1567     \l_stex_module_ns_str ? \l_stex_module_name_str
1568   }
1569
1570   \stex_annotate_invisible:nnn{header}{} {
1571     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1572     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1573     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1574       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1575     }
1576     \str_if_empty:NF \smoduletype {
1577       \stex_annotate:nnn{type}{\smoduletype}{}
1578     }
1579   }
1580 }
1581 % TODO: Inherit metatheory for nested modules?
1582 }
1583 \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}`

```

1584 \cs_new_protected:Nn \_stex_modules_end_module: {
1585   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module}
1586   \stex_reset_up_to_module:n \l_stex_current_module_str
1587   \stex_if_smsmode:T {
1588     \stex_persist:x {
1589       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1590         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1591       }
1592       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1593         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1594       }
1595       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1596         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1597       }
1598       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1599     }
1600     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1601     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1602   }
1603 }

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1604 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1605 \NewDocumentEnvironment { smodule } { 0 } { m } {
1606   \stex_module_setup:nn{#1}{#2}
1607   %\par
1608   \stex_if_smsmode:F{
1609     \tl_if_empty:NF \smoduletitle {
1610       \exp_args:No \stex_document_title:n \smoduletitle
1611     }

```

```

1612 \tl_clear:N \l_tmpa_tl
1613 \clist_map_inline:Nn \smodulotype {
1614   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1615     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1616   }
1617 }
1618 \tl_if_empty:NTF \l_tmpa_tl {
1619   \__stex_modules_smodule_start:
1620 }{
1621   \l_tmpa_tl
1622 }
1623 }
1624 \__stex_modules_begin_module:
1625 \str_if_empty:NF \smoduleid {
1626   \stex_ref_new_doc_target:n \smoduleid
1627 }
1628 \stex_smsmode_do:
1629 } {
1630   \__stex_modules_end_module:
1631   \stex_if_smsmode:F {
1632     \end{stex_annotate_env}
1633     \clist_set:Nn \l_tmpa_clist \smodulotype
1634     \tl_clear:N \l_tmpa_tl
1635     \clist_map_inline:Nn \l_tmpa_clist {
1636       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1637         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1638       }
1639     }
1640     \tl_if_empty:NTF \l_tmpa_tl {
1641       \__stex_modules_smodule_end:
1642     }{
1643       \l_tmpa_tl
1644     }
1645   }
1646 }

```

\stexpatchmodule

```

1647 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1648 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1649
1650 \newcommand\stexpatchmodule[3] [] {
1651   \str_set:Nx \l_tmpa_str{ #1 }
1652   \str_if_empty:NTF \l_tmpa_str {
1653     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1654     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1655   }{
1656     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1657     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1658   }
1659 }

```

(End definition for \stexpatchmodule. This function is documented on page 77.)

26.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1660 \NewDocumentCommand \STEXModule { m } {
1661   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1662   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1663   \tl_set:Nn \l_tmpa_tl {
1664     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1665   }
1666   \seq_map_inline:Nn \l_stex_all_modules_seq {
1667     \str_set:Nn \l_tmpb_str { ##1 }
1668     \str_if_eq:eeT { \l_tmpa_str } {
1669       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1670     } {
1671       \seq_map_break:n {
1672         \tl_set:Nn \l_tmpa_tl {
1673           \stex_invoke_module:n { ##1 }
1674         }
1675       }
1676     }
1677   }
1678   \l_tmpa_tl
1679 }
1680
1681 \cs_new_protected:Nn \stex_invoke_module:n {
1682   \stex_debug:nn{modules}{Invoking~module~#1}
1683   \peek_charcode_remove:NTF ! {
1684     \__stex_modules_invoke_uri:nN { #1 }
1685   } {
1686     \peek_charcode_remove:NTF ? {
1687       \__stex_modules_invoke_symbol:nn { #1 }
1688     } {
1689       \msg_error:nnx{stex}{error/syntax}{
1690         ?~or~!~expected~after~
1691         \c_backslash_str STEXModule{#1}
1692       }
1693     }
1694   }
1695 }
1696
1697 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1698   \str_set:Nn #2 { #1 }
1699 }
1700
1701 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1702   \stex_invoke_symbol:n{#1?#2}
1703 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 77.)

```

\stex_activate_module:n
1704 \bool_new:N \l_stex_in_meta_bool
1705 \bool_set_false:N \l_stex_in_meta_bool

```

```

1706 \cs_new_protected:Nn \stex_activate_module:n {
1707   \stex_debug:nn{modules}{Activating~module~#1}
1708   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1709     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1710     \use:c{ c_stex_module_#1_code }
1711   }
1712 }

```

(End definition for \stex_activate_module:n. This function is documented on page 78.)

```

1713 \</package>

```


Chapter 27

STEX -Module Inheritance Implementation

```
1714 <*package>
1715
1716 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1717
```

27.1 SMS Mode

```
1718 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1719 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1720 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1721 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1722
1723 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1724   \makeatletter
1725   \makeatother
1726   \ExplSyntaxOn
1727   \ExplSyntaxOff
1728   \rustexBREAK
1729 }
1730
1731 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1732   \symdef
1733   \importmodule
1734   \notation
1735   \symdecl
1736   \STEXexport
1737   \inlineass
1738   \inlinedef
1739   \inlineex
1740   \endinput
1741   \setnotation
```

```

1742 \copynotation
1743 \assign
1744 \renamedekl
1745 \donotcopy
1746 \instantiate
1747 \textsymdecl
1748 }
1749
1750 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1751   \tl_to_str:n {
1752     smodule,
1753     copymodule,
1754     interpretmodule,
1755     realization,
1756     sdefinition,
1757     sexample,
1758     sassertion,
1759     sparagraph,
1760     mathstructure
1761   }
1762 }

```

(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 79.)

`\stex_if_smsmode_p:`

`\stex_if_smsmode:TF`

```

1763 \bool_new:N \g__stex_smsmode_bool
1764 \bool_set_false:N \g__stex_smsmode_bool
1765 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1766   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1767 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 79.)

`_stex_smsmode_in_smsmode:nn`

```

1768 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1769   \vbox_set:Nn \l_tmpa_box {
1770     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1771     \bool_gset_true:N \g__stex_smsmode_bool
1772     #2
1773     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1774   }
1775   \box_clear:N \l_tmpa_box
1776 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1777 \quark_new:N \q__stex_smsmode_break
1778
1779 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1780   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1781   \stex_smsmode_do:
1782 }
1783

```

```

1784 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1785   \__stex_modules_args:n{#1}
1786   \stex_if_in_module:F {
1787     \str_if_empty:NF \l_stex_module_sig_str {
1788       \stex_modules_current_namespace:
1789       \str_set:Nx \l_stex_module_name_str { #2 }
1790       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1791         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1792         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1793         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1794         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1795         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1796         \str_set:Nx \l_tmpa_str {
1797           \stex_path_to_string:N \l_tmpa_seq /
1798           \l_tmpa_str . \l_stex_module_sig_str .tex
1799         }
1800         \IfFileExists \l_tmpa_str {
1801           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1802         }{
1803           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1804         }
1805       }
1806     }
1807   }
1808 }
1809
1810 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
1811   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
1812   \tl_if_empty:nTF{#1}{
1813     \prop_if_exist:NTF \l_stex_current_repository_prop
1814     {
1815       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
1816       \prg_return_true:
1817     } {
1818       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
1819       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
1820       \tl_if_empty:NT \l_tmpa_tl {
1821         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
1822       }
1823       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
1824       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
1825       \prg_return_true: \prg_return_false:
1826     }
1827   }\prg_return_true:
1828 }
1829
1830 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1831   \stex_filestack_push:n{#1}
1832   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1833   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1834   % ----- new -----
1835   \__stex_smsmode_in_smsmode:nn{#1}{
1836     \let\importmodule\__stex_smsmode_importmodule:
1837     \let\stex_module_setup:nn\__stex_smsmode_module:nn

```

```

1838 \let\__stex_modules_begin_module:\relax
1839 \let\__stex_modules_end_module:\relax
1840 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1841 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1842 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1843 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1844 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1845 \everyeof{\q__stex_smsmode_break\noexpand}
1846 \expandafter\expandafter\expandafter
1847 \stex_smsmode_do:
1848 \csname @ @ input\endcsname "#1"\relax
1849
1850 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1851   \stex_filestack_push:n{##1}
1852   \expandafter\expandafter\expandafter
1853   \stex_smsmode_do:
1854   \csname @ @ input\endcsname "##1"\relax
1855   \stex_filestack_pop:
1856 }
1857 }
1858 % ----- new -----
1859 \__stex_smsmode_in_smsmode:nn{#1} {
1860   #2
1861   % ----- new -----
1862   \begingroup
1863   \%stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1864   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1865     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
1866       \stex_import_module_uri:nn ##1
1867       \stex_import_require_module:nnnn
1868       \l_stex_import_ns_str
1869       \l_stex_import_archive_str
1870       \l_stex_import_path_str
1871       \l_stex_import_name_str \endgroup
1872     }
1873   }
1874   \endgroup
1875   \%stex_debug:nn{smsmode}{Actually~loading~file~#1}
1876   % ----- new -----
1877   \everyeof{\q__stex_smsmode_break\noexpand}
1878   \expandafter\expandafter\expandafter
1879   \stex_smsmode_do:
1880   \csname @ @ input\endcsname "#1"\relax
1881 }
1882 \stex_filestack_pop:
1883 }

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 80.)

`\stex_smsmode_do:` is executed on encountering `\` in smsmode. It checks whether the corresponding command is allowed and executes or ignores it accordingly:

```

1884 \cs_new_protected:Npn \stex_smsmode_do: {
1885   \stex_if_smsmode:T {
1886     \__stex_smsmode_do:w

```

```

1887 }
1888 }
1889 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1890   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1891     \expandafter\if\expandafter\relax\noexpand#1
1892     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1893   \else\expandafter\__stex_smsmode_do:w\fi
1894 }{
1895   \__stex_smsmode_do:w % #1
1896 }
1897 }
1898 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1899   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1900     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1901       #1\__stex_smsmode_do:w
1902     }{
1903       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1904         #1
1905       }{
1906         \cs_if_eq:NNTF \begin #1 {
1907           \__stex_smsmode_check_begin:n
1908         }{
1909           \cs_if_eq:NNTF \end #1 {
1910             \__stex_smsmode_check_end:n
1911           }{
1912             \__stex_smsmode_do:w
1913           }
1914         }
1915       }
1916     }
1917   }
1918 }
1919
1920 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1921   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1922     \begin{#1}
1923   }{
1924     \__stex_smsmode_do:w
1925   }
1926 }
1927 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1928   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1929     \end{#1}\__stex_smsmode_do:w
1930   }{
1931     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1932   }
1933 }

```

(End definition for `\stex_smsmode_do:.` This function is documented on page 80.)

27.2 Inheritance

```

1934 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```
1935 \cs_new_protected:Nn \stex_import_module_uri:nn {
1936   \str_set:Nx \l_stex_import_archive_str { #1 }
1937   \str_set:Nn \l_stex_import_path_str { #2 }
1938
1939   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1940   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1941   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1942
1943   \stex_modules_current_namespace:
1944   \bool_lazy_all:nTF {
1945     {\str_if_empty_p:N \l_stex_import_archive_str}
1946     {\str_if_empty_p:N \l_stex_import_path_str}
1947     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1948   }{
1949     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1950     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1951   }{
1952     \str_if_empty:NT \l_stex_import_archive_str {
1953       \prop_if_exist:NT \l_stex_current_repository_prop {
1954         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1955       }
1956     }
1957     \str_if_empty:NTF \l_stex_import_archive_str {
1958       \str_if_empty:NF \l_stex_import_path_str {
1959         \stex_path_from_string:Nn \l_tmpb_seq {
1960           \l_stex_module_ns_str / .. / \l_stex_import_path_str
1961         }
1962         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
1963         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file:///}
1964       }
1965     }{
1966       \stex_require_repository:n \l_stex_import_archive_str
1967       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1968       \l_stex_import_ns_str
1969       \str_if_empty:NF \l_stex_import_path_str {
1970         \str_set:Nx \l_stex_import_ns_str {
1971           \l_stex_import_ns_str / \l_stex_import_path_str
1972         }
1973       }
1974     }
1975   }
1976 }
```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 81.)

`\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`.

```
1977 \str_new:N \l_stex_import_name_str
1978 \str_new:N \l_stex_import_archive_str
1979 \str_new:N \l_stex_import_path_str
1980 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 81.)

```

\stex_import_require_module:nnnnn {\langle ns \rangle} {\langle archive-ID \rangle} {\langle path \rangle} {\langle name \rangle}
1981 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1982   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1983
1984     \stex_debug:nn{requiremodule}{Here:\~1:\~2:\~3:\~4}
1985
1986     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1987     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1988
1989     %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1990
1991     % archive
1992     \str_set:Nx \l_tmpa_str { #2 }
1993     \str_if_empty:NTF \l_tmpa_str {
1994       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1995       \seq_put_right:Nn \l_tmpa_seq {...}
1996     } {
1997       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1998       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1999       \seq_put_right:Nn \l_tmpa_seq { source }
2000     }
2001
2002     % path
2003     \str_set:Nx \l_tmpb_str { #3 }
2004     \str_if_empty:NTF \l_tmpb_str {
2005       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2006
2007       \ltx@ifpackageloaded{babel} {
2008         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2009           { \language } \l_tmpb_str {
2010           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2011         }
2012       } {
2013         \str_clear:N \l_tmpb_str
2014       }
2015
2016       \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2017       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2018         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2019       }{
2020         \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2021         \IfFileExists{ \l_tmpa_str.tex }{
2022           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2023         }{
2024           % try english as default
2025           \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2026           \IfFileExists{ \l_tmpa_str.en.tex }{
2027             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2028           }{
2029             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2030           }
2031         }
2032       }
2033

```

```

2034 } {
2035   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2036   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2037
2038   \ltx@ifpackageloaded{babel} {
2039     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2040       { \language } \l_tmpb_str {
2041       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2042     }
2043   } {
2044     \str_clear:N \l_tmpb_str
2045   }
2046
2047   \stex_path_canonicalize:N \l_tmpb_seq
2048   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2049
2050   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2051   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2052     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2053   }{
2054     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2055     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2056       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2057     }{
2058       % try english as default
2059       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2060       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2061         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2062       }{
2063         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2064         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2065           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2066         }{
2067           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2068           \IfFileExists{ \l_tmpa_str.tex }{
2069             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2070           }{
2071             % try english as default
2072             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2073             \IfFileExists{ \l_tmpa_str.en.tex }{
2074               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2075             }{
2076               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2077             }
2078           }
2079         }
2080       }
2081     }
2082   }
2083 }
2084
2085 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2086   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2087     \seq_clear:N \l_stex_all_modules_seq

```



```

2088     \str_clear:N \l_stex_current_module_str
2089     \str_set:Nx \l_tmpb_str { #2 }
2090     \str_if_empty:NF \l_tmpb_str {
2091       \stex_set_current_repository:n { #2 }
2092     }
2093     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2094   }
2095
2096   \stex_if_module_exists:nF { #1 ? #4 } {
2097     \msg_error:nnx{stex}{error/unknownmodule}{
2098       #1?#4~(in~file~\g__stex_importmodule_file_str)
2099     }
2100   }
2101 }
2102
2103 }
2104 \stex_activate_module:n { #1 ? #4 }
2105 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 81.)

`\importmodule`

```

2106 \NewDocumentCommand \importmodule { 0{} m } {
2107   \stex_import_module_uri:nn { #1 } { #2 }
2108   \stex_debug:nn{modules}{Importing~module:~
2109     \l_stex_import_ns_str ? \l_stex_import_name_str
2110   }
2111   \stex_import_require_module:nnnn
2112   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2113   { \l_stex_import_path_str } { \l_stex_import_name_str }
2114   \stex_if_smsmode:F {
2115     \stex_annotate_invisible:nnn
2116     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2117   }
2118   \exp_args:Nx \stex_add_to_current_module:n {
2119     \stex_import_require_module:nnnn
2120     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2121     { \l_stex_import_path_str } { \l_stex_import_name_str }
2122   }
2123   \exp_args:Nx \stex_add_import_to_current_module:n {
2124     \l_stex_import_ns_str ? \l_stex_import_name_str
2125   }
2126   \stex_smsmode_do:
2127   \ignorespacesandpars
2128 }
2129 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

2130 \NewDocumentCommand \usemodule { 0{} m } {
2131   \stex_if_smsmode:F {
2132     \stex_import_module_uri:nn { #1 } { #2 }
2133     \stex_import_require_module:nnnn
2134     { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```

```

2135     { \l_stex_import_path_str } { \l_stex_import_name_str }
2136     \stex_annotate_invisible:nnn
2137     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2138   }
2139   \stex_smsmode_do:
2140   \ignorespacesandpars
2141 }

```

(End definition for \usemodule. This function is documented on page 80.)

```

2142 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2143   \tl_if_empty:nF{#2}{
2144     \clist_set:Nn \l_tmpa_clist {#2}
2145     \clist_map_inline:Nn \l_tmpa_clist {
2146       \tl_if_head_eq_charcode:nNTF {##1} [{
2147         #1 ##1
2148       }{
2149         #1{##1}
2150       }
2151     }
2152   }
2153 }
2154 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2155
2156
2157 </package>

```

Chapter 28

STEX -Symbols Implementation

```
2158 <*package>
2159
2160 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2161
2162 Warnings and error messages
2163 \msg_new:nnn{stex}{error/wrongargs}{
2164   args~value~in~symbol~declaration~for~#1~
2165   needs~to~be~i,~a,~b~or~B,~but~#2~given
2166 }
2167 \msg_new:nnn{stex}{error/unknownsymbol}{
2168   No~symbol~#1~found!
2169 }
2170 \msg_new:nnn{stex}{error/seqlength}{
2171   Expected~#1~arguments;~got~#2!
2172 }
2173 \msg_new:nnn{stex}{error/unknownnotation}{
2174   Unknown~notation~#1~for~#2!
2175 }
```

28.1 Symbol Declarations

```
2175 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2176 \cs_new_protected:Nn \stex_all_symbols:n {
2177   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2178   \seq_map_inline:Nn \l_stex_all_modules_seq {
2179     \seq_map_inline:cn{c_stex_module_##1_constants}{
2180       \__stex_symdecl_all_symbols_cs{##1?####1}
2181     }
2182   }
2183 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page [83](#).)

`\STEXsymbol`

```
2184 \NewDocumentCommand \STEXsymbol { m } {
2185   \stex_get_symbol:n { #1 }
2186   \exp_args:No
2187   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2188 }
```

(End definition for `\STEXsymbol`. This function is documented on page 84.)

`symdecl` arguments:

```
2189 \keys_define:nn { stex / symdecl } {
2190   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2191   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2192   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2193   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2194   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2195   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2196   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2197   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2198   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2199   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2200   assoc     .choices:nn =
2201     {bin,binl,binr,pre,conj,pwconj}
2202     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2203 }
2204
2205 \bool_new:N \l_stex_symdecl_make_macro_bool
2206
2207 \cs_new_protected:Nn \__stex_symdecl_args:n {
2208   \str_clear:N \l_stex_symdecl_name_str
2209   \str_clear:N \l_stex_symdecl_args_str
2210   \str_clear:N \l_stex_symdecl_deprecate_str
2211   \str_clear:N \l_stex_symdecl_reorder_str
2212   \str_clear:N \l_stex_symdecl_assoctype_str
2213   \bool_set_false:N \l_stex_symdecl_local_bool
2214   \tl_clear:N \l_stex_symdecl_type_tl
2215   \tl_clear:N \l_stex_symdecl_definiens_tl
2216
2217   \keys_set:nn { stex / symdecl } { #1 }
2218 }
```

`\symdecl` Parses the optional arguments and passes them on to `\stex_symdecl_do:` (so that `\symdef` can do the same)

```
2219
2220 \NewDocumentCommand \symdecl { s m O{} } {
2221   \__stex_symdecl_args:n { #3 }
2222   \IfBooleanTF #1 {
2223     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2224   } {
2225     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2226   }
2227   \stex_symdecl_do:n { #2 }
2228   \stex_smsmode_do:
2229 }
```

```

2230
2231 \cs_new_protected:Nn \stex_symdecl_do:nn {
2232   \__stex_symdecl_args:n{#1}
2233   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2234   \stex_symdecl_do:n{#2}
2235 }
2236
2237 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 82.)

\stex_symdecl_do:n

```

2238 \cs_new_protected:Nn \stex_symdecl_do:n {
2239   \stex_if_in_module:F {
2240     % TODO throw error? some default namespace?
2241   }
2242
2243   \str_if_empty:NT \l_stex_symdecl_name_str {
2244     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2245   }
2246
2247   \prop_if_exist:cT { l_stex_symdecl_
2248     \l_stex_current_module_str ?
2249     \l_stex_symdecl_name_str
2250   _prop
2251 }{
2252   % TODO throw error (beware of circular dependencies)
2253 }
2254
2255 \prop_clear:N \l_tmpa_prop
2256 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2257 \seq_clear:N \l_tmpa_seq
2258 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2259 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2260
2261 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2262   \str_if_empty:NF \l_stex_module_deprecate_str {
2263     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2264   }
2265 }
2266 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2267
2268 \exp_args:No \stex_add_constant_to_current_module:n {
2269   \l_stex_symdecl_name_str
2270 }
2271
2272 % arity/args
2273 \int_zero:N \l_tmpb_int
2274
2275 \bool_set_true:N \l_tmpa_bool
2276 \str_map_inline:Nn \l_stex_symdecl_args_str {
2277   \token_case_meaning:NnF ##1 {
2278     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2279     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2280     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2281     {\tl_to_str:n a} {
2282         \bool_set_false:N \l_tmpa_bool
2283         \int_incr:N \l_tmpb_int
2284     }
2285     {\tl_to_str:n B} {
2286         \bool_set_false:N \l_tmpa_bool
2287         \int_incr:N \l_tmpb_int
2288     }
2289     ){
2290         \msg_error:nnxx{stex}{error/wrongargs}{
2291             \l_stex_current_module_str ?
2292             \l_stex_symdecl_name_str
2293         }{##1}
2294     }
2295 }
2296 \bool_if:NTF \l_tmpa_bool {
2297     % possibly numeric
2298     \str_if_empty:NTF \l_stex_symdecl_args_str {
2299         \prop_put:Nnn \l_tmpa_prop { args } {}
2300         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2301     }{
2302         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2303         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2304         \str_clear:N \l_tmpa_str
2305         \int_step_inline:nn \l_tmpa_int {
2306             \str_put_right:Nn \l_tmpa_str i
2307         }
2308         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2309     }
2310 } {
2311     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2312     \prop_put:Nnx \l_tmpa_prop { arity }
2313     { \str_count:N \l_stex_symdecl_args_str }
2314 }
2315 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2316
2317 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2318     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2319 }{
2320     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2321 }
2322
2323 % semantic macro
2324
2325 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2326     \exp_args:Nx \stex_do_up_to_module:n {
2327         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2328             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2329         }}
2330     }
2331 }
2332
2333 \stex_debug:nn{symbols}{New~symbol:~

```

```

2334 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2335 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2336 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2337 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2338 }
2339
2340 % circular dependencies require this:
2341 \stex_if_do_html:T {
2342   \stex_annotate_invisible:nnn {symdecl} {
2343     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2344   } {
2345     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2346       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2347     }
2348     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{ }
2349     \stex_annotate_invisible:nnn{macroname}{#1}{ }
2350     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2351       \stex_annotate_invisible:nnn{definiens}{ }
2352       {\l_stex_symdecl_definiens_tl$}
2353     }
2354     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2355       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{ }
2356     }
2357     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2358       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{ }
2359     }
2360   }
2361 }
2362 \prop_if_exist:cF {
2363   \l_stex_symdecl_
2364   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2365   _prop
2366 } {
2367   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2368   \__stex_symdecl_restore_symbol:nnnnnnn
2369   {\l_stex_symdecl_name_str}
2370   { \prop_item:Nn \l_tmpa_prop {args} }
2371   { \prop_item:Nn \l_tmpa_prop {arity} }
2372   { \prop_item:Nn \l_tmpa_prop {assocs} }
2373   { \prop_item:Nn \l_tmpa_prop {defined} }
2374   {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2375   {\l_stex_current_module_str}
2376 }
2377 }
2378 }
2379 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2380   \prop_clear:N \l_tmpa_prop
2381   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2382   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2383   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2384   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2385   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2386   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2387   \tl_if_empty:nF{#6}{

```

```

2388     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2389   }
2390   \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2391   \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2392 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 83.)

`\textsymdecl`

```

2393
2394 \keys_define:nn { stex / textsymdecl } {
2395   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2396   type      .tl_set:N    = \l__stex_symdecl_type_tl
2397 }
2398
2399 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2400   \str_clear:N \l__stex_symdecl_name_str
2401   \tl_clear:N \l__stex_symdecl_type_tl
2402   \keys_set:nn { stex / textsymdecl } { #1 }
2403 }
2404
2405 \NewDocumentCommand \textsymdecl {m O{} m} {
2406   \_stex_textsymdecl_args:n { #2 }
2407   \str_if_empty:NTF \l__stex_symdecl_name_str {
2408     \__stex_symdecl_args:n{name=#1,#2}
2409   }{
2410     \__stex_symdecl_args:n{#2}
2411   }
2412   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2413   \stex_symdecl_do:n{#1-sym}
2414   \stex_execute_in_module:n{
2415     \cs_set_nopar:cpn{#1name}{
2416       \ifvmode\hbox_unpack:N\c_empty_box\fi
2417       \ifmmode\hbox{#3}\else#3\fi\xspace
2418     }
2419     \cs_set_nopar:cpn{#1}{
2420       \ifmmode\csname#1-sym\expandafter\endcsname\else
2421       \ifvmode\hbox_unpack:N\c_empty_box\fi
2422       \symref{#1-sym}{#3}\expandafter\xspace
2423       \fi
2424     }
2425   }
2426   \stex_execute_in_module:x{
2427     \__stex_notation_restore_notation:nnnnn
2428     {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl
2429     }{0}
2430     {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{
2431       \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2432     }}}}
2433   {}
2434 }
2435 \stex_smsmode_do:
2436 }

```

(End definition for `\textsymdecl`. This function is documented on page 19.)

`\stex_get_symbol:n`

```
2437 \str_new:N \l_stex_get_symbol_uri_str
2438
2439 \cs_new_protected:Nn \stex_get_symbol:n {
2440   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2441     \tl_set:Nn \l_tmpa_tl { #1 }
2442     \__stex_symdecl_get_symbol_from_cs:
2443   }{
2444     % argument is a string
2445     % is it a command name?
2446     \cs_if_exist:cTF { #1 }{
2447       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2448       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2449       \str_if_empty:NTF \l_tmpa_str {
2450         \exp_args:Nx \cs_if_eq:NNTF {
2451           \tl_head:N \l_tmpa_tl
2452         } \stex_invoke_symbol:n {
2453           \__stex_symdecl_get_symbol_from_cs:
2454         }{
2455           \__stex_symdecl_get_symbol_from_string:n { #1 }
2456         }
2457       } {
2458         \__stex_symdecl_get_symbol_from_string:n { #1 }
2459       }
2460     }{
2461       % argument is not a command name
2462       \__stex_symdecl_get_symbol_from_string:n { #1 }
2463       % \l_stex_all_symbols_seq
2464     }
2465   }
2466   \str_if_eq:eeF {
2467     \prop_item:cn {
2468       l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2469     }{ deprecate }
2470   }{}{
2471     \msg_warning:nnxx{stex}{warning/deprecated}{
2472       Symbol~\l_stex_get_symbol_uri_str
2473     }{
2474       \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2475     }
2476   }
2477 }
2478
2479 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2480   \tl_set:Nn \l_tmpa_tl {
2481     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2482   }
2483   \str_set:Nn \l_tmpa_str { #1 }
2484
2485   %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2486
2487   \str_if_in:NnTF \l_tmpa_str ? {
2488     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2489     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```

```

2490 \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2491 }{
2492 \str_clear:N \l_tmpb_str
2493 }
2494 \str_if_empty:NTF \l_tmpb_str {
2495 \seq_map_inline:Nn \l_stex_all_modules_seq {
2496 \seq_map_inline:cn{c_stex_module_###1_constants}{
2497 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2498 \seq_map_break:n{\seq_map_break:n{
2499 \tl_set:Nn \l_tmpa_tl {
2500 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2501 }
2502 }}
2503 }
2504 }
2505 }
2506 }{
2507 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2508 \seq_map_inline:Nn \l_stex_all_modules_seq {
2509 \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2510 \seq_map_inline:cn{c_stex_module_###1_constants}{
2511 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2512 \seq_map_break:n{\seq_map_break:n{
2513 \tl_set:Nn \l_tmpa_tl {
2514 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2515 }
2516 }}
2517 }
2518 }
2519 }
2520 }
2521 }
2522
2523 \l_tmpa_tl
2524 }
2525
2526 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2527 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2528 { \tl_tail:N \l_tmpa_tl }
2529 \tl_if_single:NTF \l_tmpa_tl {
2530 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2531 \exp_after:wN \str_set:Nn \exp_after:wN
2532 \l_stex_get_symbol_uri_str \l_tmpa_tl
2533 }{
2534 % TODO
2535 % tail is not a single group
2536 }
2537 }{
2538 % TODO
2539 % tail is not a single group
2540 }
2541 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 83.)

28.2 Notations

```

2542 <@@=stex_notation>

notation arguments:
2543 \keys_define:nn { stex / notation } {
2544 % lang .tl_set_x:N = \l__stex_notation_lang_str ,
2545 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2546 prec .str_set_x:N = \l__stex_notation_prec_str ,
2547 op .tl_set:N = \l__stex_notation_op_tl ,
2548 primary .bool_set:N = \l__stex_notation_primary_bool ,
2549 primary .default:n = {true} ,
2550 unknown .code:n = \str_set:Nx
2551 \l__stex_notation_variant_str \l_keys_key_str
2552 }
2553
2554 \cs_new_protected:Nn \stex_notation_args:n {
2555 % \str_clear:N \l__stex_notation_lang_str
2556 \str_clear:N \l__stex_notation_variant_str
2557 \str_clear:N \l__stex_notation_prec_str
2558 \tl_clear:N \l__stex_notation_op_tl
2559 \bool_set_false:N \l__stex_notation_primary_bool
2560
2561 \keys_set:nn { stex / notation } { #1 }
2562 }

\notation

2563 \NewDocumentCommand \notation { s m O{}} {
2564 \stex_notation_args:n { #3 }
2565 \tl_clear:N \l_stex_symdecl_definiens_tl
2566 \stex_get_symbol:n { #2 }
2567 \tl_set:Nn \l_stex_notation_after_do_tl {
2568 \__stex_notation_final:
2569 \IfBooleanTF#1{
2570 \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2571 }{}
2572 \stex_smsmode_do:\ignorespacesandpars
2573 }
2574 \stex_notation_do:nnnnn
2575 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { args } }
2576 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
2577 { \l__stex_notation_variant_str }
2578 { \l__stex_notation_prec_str }
2579 }
2580 \stex_deactivate_macro:Nn \notation {module-environments}

(End definition for \notation. This function is documented on page 83.)

\stex_notation_do:nnnnn

2581 \seq_new:N \l__stex_notation_precedences_seq
2582 \tl_new:N \l__stex_notation_opprec_tl
2583 \int_new:N \l__stex_notation_currarg_int
2584 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2585
2586 \cs_new_protected:Nn \stex_notation_do:nnnnn {

```

```

2587 \let\STEXInternalCurrentSymbolStr\relax
2588 \seq_clear:N \l__stex_notation_precedences_seq
2589 \tl_clear:N \l__stex_notation_opprec_tl
2590 \str_set:Nx \l__stex_notation_args_str { #1 }
2591 \str_set:Nx \l__stex_notation_arity_str { #2 }
2592 \str_set:Nx \l__stex_notation_suffix_str { #3 }
2593 \str_set:Nx \l__stex_notation_prec_str { #4 }
2594
2595 % precedences
2596 \str_if_empty:NTF \l__stex_notation_prec_str {
2597   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2598     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2599   }{
2600     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2601   }
2602 } {
2603   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2604     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2605     \int_step_inline:nn { \l__stex_notation_arity_str } {
2606       \exp_args:NNo
2607       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2608     }
2609   }{
2610     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2611     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2612       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2613       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2614         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2615           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2616         \seq_map_inline:Nn \l_tmpa_seq {
2617           \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }
2618         }
2619       }
2620     }{
2621       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2622         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2623       }{
2624         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2625       }
2626     }
2627   }
2628 }
2629
2630 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2631 \int_step_inline:nn { \l__stex_notation_arity_str } {
2632   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2633     \exp_args:NNo
2634     \seq_put_right:No \l__stex_notation_precedences_seq {
2635       \l__stex_notation_opprec_tl
2636     }
2637   }
2638 }
2639 \tl_clear:N \l__stex_notation_dummyargs_tl
2640

```

```

2641 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2642   \exp_args:NNe
2643   \cs_set:Npn \l_stex_notation_macrocode_cs {
2644     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2645     { \l__stex_notation_suffix_str }
2646     { \l__stex_notation_opprec_tl }
2647     { \exp_not:n { #5 } }
2648   }
2649   \l_stex_notation_after_do_tl
2650 }{
2651   \str_if_in:NnTF \l__stex_notation_args_str b {
2652     \exp_args:Nne \use:nn
2653     {
2654       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2655       \cs_set:Npn \l__stex_notation_arity_str } { {
2656         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2657         { \l__stex_notation_suffix_str }
2658         { \l__stex_notation_opprec_tl }
2659         { \exp_not:n { #5 } }
2660       } }
2661   }{
2662     \str_if_in:NnTF \l__stex_notation_args_str B {
2663       \exp_args:Nne \use:nn
2664       {
2665         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2666         \cs_set:Npn \l__stex_notation_arity_str } { {
2667           \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2668           { \l__stex_notation_suffix_str }
2669           { \l__stex_notation_opprec_tl }
2670           { \exp_not:n { #5 } }
2671         } }
2672     }{
2673       \exp_args:Nne \use:nn
2674       {
2675         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2676         \cs_set:Npn \l__stex_notation_arity_str } { {
2677           \STEXInternalTermMathOMAiiii { \STEXInternalCurrentSymbolStr }
2678           { \l__stex_notation_suffix_str }
2679           { \l__stex_notation_opprec_tl }
2680           { \exp_not:n { #5 } }
2681         } }
2682     }
2683   }
2684
2685   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2686   \int_zero:N \l__stex_notation_currarg_int
2687   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2688   \__stex_notation_arguments:
2689 }
2690 }

```

(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

```

2691 \cs_new_protected:Nn \__stex_notation_arguments: {
2692   \int_incr:N \l__stex_notation_currarg_int
2693   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2694     \l_stex_notation_after_do_tl
2695   }{
2696     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2697     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2698     \str_if_eq:VnTF \l_tmpa_str a {
2699       \__stex_notation_argument_assoc:nn{a}
2700     }{
2701       \str_if_eq:VnTF \l_tmpa_str B {
2702         \__stex_notation_argument_assoc:nn{B}
2703       }{
2704         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2705         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2706           { \STEXInternalTermMathArgiii
2707             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2708             { \l_tmpb_str }
2709             { ####\int_use:N \l__stex_notation_currarg_int }
2710           }
2711         }
2712         \__stex_notation_arguments:
2713       }
2714     }
2715   }
2716 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```

2717 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2718
2719   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2720     {\l__stex_notation_arity_str}{
2721       #2
2722     }
2723   \int_zero:N \l_tmpa_int
2724   \tl_clear:N \l_tmpa_tl
2725   \str_map_inline:Nn \l__stex_notation_args_str {
2726     \int_incr:N \l_tmpa_int
2727     \tl_put_right:Nx \l_tmpa_tl {
2728       \str_if_eq:nnTF {##1}{a}{ {} }{
2729         \str_if_eq:nnTF {##1}{B}{ {} }{
2730           {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2731         }
2732       }
2733     }
2734   }
2735   \exp_after:wN\exp_after:wN\exp_after:wN \def
2736   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2737   \exp_after:wN\exp_after:wN\exp_after:wN ##
2738   \exp_after:wN\exp_after:wN\exp_after:wN 1
2739   \exp_after:wN\exp_after:wN\exp_after:wN ##
2740   \exp_after:wN\exp_after:wN\exp_after:wN 2

```

```

2741 \exp_after:wN\exp_after:wN\exp_after:wN {
2742   \exp_after:wN \exp_after:wN \exp_after:wN
2743   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2744     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2745   }
2746 }
2747
2748 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2749 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2750   \STEXInternalTermMathAssocArgiiii
2751   { #1\int_use:N \l__stex_notation_currarg_int }
2752   { \l_tmpa_str }
2753   { ####\int_use:N \l__stex_notation_currarg_int }
2754   { \l_tmpa_cs {####1} {####2} }
2755 } }
2756 \__stex_notation_arguments:
2757 }

```

(End definition for __stex_notation_argument_assoc:nn.)

__stex_notation_final: Called after processing all notation arguments

```

2758 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2759   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2760   \cs_set_nopar:Npn {#3}{#4}
2761   \tl_if_empty:nF {#5}{
2762     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2763   }
2764   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2765     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2766   }
2767 }
2768
2769 \cs_new_protected:Nn \__stex_notation_final: {
2770
2771   \stex_execute_in_module:x {
2772     \__stex_notation_restore_notation:nnnnn
2773     {\l_stex_get_symbol_uri_str}
2774     {\l__stex_notation_suffix_str}
2775     {\l__stex_notation_arity_str}
2776     {
2777       \exp_after:wN \exp_after:wN \exp_after:wN
2778       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2779       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
2780     }
2781     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2782   }
2783
2784   \stex_debug:nn{symbols}{
2785     Notation~\l__stex_notation_suffix_str
2786     ~for~\l_stex_get_symbol_uri_str^^J
2787     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2788     Argument~precedences:~
2789     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2790     Notation: \cs_meaning:c {

```

```

2791     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2792     \l__stex_notation_suffix_str
2793     _cs
2794 }
2795 }
2796 % HTML annotations
2797 \stex_if_do_html:T {
2798   \stex_annotate_invisible:nnn { notation }
2799   { \l_stex_get_symbol_uri_str } {
2800     \stex_annotate_invisible:nnn { notationfragment }
2801     { \l__stex_notation_suffix_str }{}
2802     \stex_annotate_invisible:nnn { precedence }
2803     { \l__stex_notation_prec_str }{}
2804
2805     \int_zero:N \l_tmpa_int
2806     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2807     \tl_clear:N \l_tmpa_tl
2808     \int_step_inline:nn { \l__stex_notation_arity_str }{
2809       \int_incr:N \l_tmpa_int
2810       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2811       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
2812       \str_if_eq:VnTF \l_tmpb_str a {
2813         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2814           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2815           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2816         } }
2817       }{
2818         \str_if_eq:VnTF \l_tmpb_str B {
2819           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2820             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2821             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2822           } }
2823         }{
2824           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2825             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2826           } }
2827         }
2828       }
2829     }
2830     \stex_annotate_invisible:nnn { notationcomp }{}{
2831       \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
2832       $ \exp_args:Nno \use:nn { \use:c {
2833         stex_notation_ \STEXInternalCurrentSymbolStr
2834         \c_hash_str \l__stex_notation_suffix_str _cs
2835       } } { \l_tmpa_tl } $
2836     }
2837     \tl_if_empty:NF \l__stex_notation_op_tl {
2838       \stex_annotate_invisible:nnn { notationopcomp }{}{
2839         $\l__stex_notation_op_tl$
2840       }
2841     }
2842   }
2843 }
2844 }

```


(End definition for _stex_notation_final:.)

\setnotation

```

2845 \keys_define:nn { stex / setnotation } {
2846   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2847   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2848   unknown .code:n      = \str_set:Nx
2849     \l__stex_notation_variant_str \l_keys_key_str
2850 }
2851
2852 \cs_new_protected:Nn \stex_setnotation_args:n {
2853   % \str_clear:N \l__stex_notation_lang_str
2854   \str_clear:N \l__stex_notation_variant_str
2855   \keys_set:nn { stex / setnotation } { #1 }
2856 }
2857
2858 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2859   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2860     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2861     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2862   }
2863 }
2864
2865 \cs_new_protected:Nn \stex_setnotation:n {
2866   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2867     { \l__stex_notation_variant_str }{
2868     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2869     \stex_debug:nn {notations}{
2870       Setting~default~notation~
2871       {\l__stex_notation_variant_str }~for~
2872       #1 \\
2873       \expandafter\meaning\csname
2874       l_stex_symdecl_#1_notations\endcsname
2875     }
2876   }{
2877     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2878   }
2879 }
2880
2881 \NewDocumentCommand \setnotation {m m} {
2882   \stex_get_symbol:n { #1 }
2883   \stex_setnotation_args:n { #2 }
2884   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2885   \stex_smsmode_do:\ignorespacesandpars
2886 }
2887
2888 \cs_new_protected:Nn \stex_copy_notations:nn {
2889   \stex_debug:nn {notations}{
2890     Copying~notations~from~#2~to~#1\\
2891     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2892   }
2893   \tl_clear:N \l_tmpa_tl
2894   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2895     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }

```

```

2896 }
2897 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
2898   \stex_debug:nn{Here}{Here:~##1}
2899   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2900   \edef \l_tmpa_tl {
2901     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2902     \exp_after:wN\exp_after:wN\exp_after:wN {
2903       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2904     }
2905   }
2906
2907   \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2908   \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2909   \exp_after:wN { \l_tmpa_tl }
2910
2911   \edef \l_tmpa_tl {
2912     \exp_after:wN \exp_not:n \exp_after:wN {
2913       \l_tmpa_tl {##### 1}{##### 2}
2914     }
2915   }
2916
2917   \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
2918
2919   \stex_execute_in_module:x {
2920     \__stex_notation_restore_notation:nnnnn
2921     {#1}{##1}
2922     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2923     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2924     {
2925       \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2926         \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2927       }
2928     }
2929   }\endgroup
2930 }
2931 }
2932
2933 \NewDocumentCommand \copynotation {m m} {
2934   \stex_get_symbol:n { #1 }
2935   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2936   \stex_get_symbol:n { #2 }
2937   \exp_args:Noo
2938   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2939   \stex_smsmode_do:\ignorespacesandpars
2940 }
2941

```

(End definition for \setnotation. This function is documented on page 19.)

\symdef

```

2942 \keys_define:nn { stex / symdef } {
2943   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2944   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2945   args      .str_set_x:N = \l_stex_symdecl_args_str ,

```

```

2946 type .tl_set:N = \l_stex_symdecl_type_tl ,
2947 def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2948 reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
2949 op .tl_set:N = \l__stex_notation_op_tl ,
2950 % lang .str_set_x:N = \l__stex_notation_lang_str ,
2951 variant .str_set_x:N = \l__stex_notation_variant_str ,
2952 prec .str_set_x:N = \l__stex_notation_prec_str ,
2953 assoc .choices:nn =
2954 {bin,binl,binr,pre,conj,pwconj}
2955 {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2956 unknown .code:n = \str_set:Nx
2957 \l__stex_notation_variant_str \l_keys_key_str
2958 }
2959
2960 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2961 \str_clear:N \l_stex_symdecl_name_str
2962 \str_clear:N \l_stex_symdecl_args_str
2963 \str_clear:N \l_stex_symdecl_assoctype_str
2964 \str_clear:N \l_stex_symdecl_reorder_str
2965 \bool_set_false:N \l_stex_symdecl_local_bool
2966 \tl_clear:N \l_stex_symdecl_type_tl
2967 \tl_clear:N \l_stex_symdecl_definiens_tl
2968 % \str_clear:N \l__stex_notation_lang_str
2969 \str_clear:N \l__stex_notation_variant_str
2970 \str_clear:N \l__stex_notation_prec_str
2971 \tl_clear:N \l__stex_notation_op_tl
2972
2973 \keys_set:nn { stex / symdef } { #1 }
2974 }
2975
2976 \NewDocumentCommand \symdef { m O{} } {
2977 \__stex_notation_symdef_args:n { #2 }
2978 \bool_set_true:N \l_stex_symdecl_make_macro_bool
2979 \stex_symdecl_do:n { #1 }
2980 \tl_set:Nn \l_stex_notation_after_do_tl {
2981 \__stex_notation_final:
2982 \stex_smsmode_do:\ignorespacesandpars
2983 }
2984 \str_set:Nx \l_stex_get_symbol_uri_str {
2985 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2986 }
2987 \exp_args:Nx \stex_notation_do:nnnnn
2988 { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2989 { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2990 { \l__stex_notation_variant_str }
2991 { \l__stex_notation_prec_str }
2992 }
2993 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

28.3 Variables

```

2994 <@@=stex_variables>

```

```

2995
2996 \keys_define:nn { stex / vardef } {
2997   name      .str_set_x:N = \l__stex_variables_name_str ,
2998   args      .str_set_x:N = \l__stex_variables_args_str ,
2999   type      .tl_set:N    = \l__stex_variables_type_tl ,
3000   def       .tl_set:N    = \l__stex_variables_def_tl ,
3001   op        .tl_set:N    = \l__stex_variables_op_tl ,
3002   prec      .str_set_x:N = \l__stex_variables_prec_str ,
3003   reorder   .str_set_x:N = \l__stex_variables_reorder_str ,
3004   assoc     .choices:nn  =
3005     {bin,binl,binr,pre,conj,pwconj}
3006     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3007   bind      .choices:nn  =
3008     {forall,exists}
3009     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3010 }
3011
3012 \cs_new_protected:Nn \__stex_variables_args:n {
3013   \str_clear:N \l__stex_variables_name_str
3014   \str_clear:N \l__stex_variables_args_str
3015   \str_clear:N \l__stex_variables_prec_str
3016   \str_clear:N \l__stex_variables_assoctype_str
3017   \str_clear:N \l__stex_variables_reorder_str
3018   \str_clear:N \l__stex_variables_bind_str
3019   \tl_clear:N \l__stex_variables_type_tl
3020   \tl_clear:N \l__stex_variables_def_tl
3021   \tl_clear:N \l__stex_variables_op_tl
3022
3023   \keys_set:nn { stex / vardef } { #1 }
3024 }
3025
3026 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
3027   \__stex_variables_args:n {#2}
3028   \str_if_empty:NT \l__stex_variables_name_str {
3029     \str_set:Nx \l__stex_variables_name_str { #1 }
3030   }
3031   \prop_clear:N \l_tmpa_prop
3032   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3033
3034   \int_zero:N \l_tmpb_int
3035   \bool_set_true:N \l_tmpa_bool
3036   \str_map_inline:Nn \l__stex_variables_args_str {
3037     \token_case_meaning:NnF ##1 {
3038       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3039       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3040       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3041       {\tl_to_str:n a} {
3042         \bool_set_false:N \l_tmpa_bool
3043         \int_incr:N \l_tmpb_int
3044       }
3045       {\tl_to_str:n B} {
3046         \bool_set_false:N \l_tmpa_bool
3047         \int_incr:N \l_tmpb_int
3048       }

```

```

3049     }{
3050         \msg_error:nnxx{stex}{error/wrongargs}{
3051             variable~\l__stex_variables_name_str
3052         }{##1}
3053     }
3054 }
3055 \bool_if:NTF \l_tmpa_bool {
3056     % possibly numeric
3057     \str_if_empty:NTF \l__stex_variables_args_str {
3058         \prop_put:Nnn \l_tmpa_prop { args } {}
3059         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3060     }{
3061         \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3062         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3063         \str_clear:N \l_tmpa_str
3064         \int_step_inline:nn \l_tmpa_int {
3065             \str_put_right:Nn \l_tmpa_str i
3066         }
3067         \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3068         \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3069     }
3070 } {
3071     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3072     \prop_put:Nnx \l_tmpa_prop { arity }
3073     { \str_count:N \l__stex_variables_args_str }
3074 }
3075 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3076 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3077
3078 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
3079
3080 \tl_if_empty:NF \l__stex_variables_op_tl {
3081     \cs_set:cpx {
3082         stex_var_op_notation_\l__stex_variables_name_str _cs
3083     } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3084 }
3085
3086 \tl_set:Nn \l_stex_notation_after_do_tl {
3087     \exp_args:Nne \use:nn {
3088         \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3089         \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3090     } {{
3091         \exp_after:wN \exp_after:wN \exp_after:wN
3092         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3093         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3094     }}
3095 \stex_if_do_html:T {
3096     \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3097         \stex_annotate_invisible:nnn { precedence }
3098         { \l__stex_variables_prec_str }{}
3099         \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l
3100         \stex_annotate_invisible:nnn{args}{ \l__stex_variables_args_str }{}
3101         \stex_annotate_invisible:nnn{macroname}{#1}{}
3102         \tl_if_empty:NF \l__stex_variables_def_tl {

```

```

3103     \stex_annotate_invisible:nnn{definiens}{}
3104     {${\l__stex_variables_def_tl$}
3105   }
3106   \str_if_empty:NF \l__stex_variables_assoctype_str {
3107     \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3108   }
3109   \str_if_empty:NF \l__stex_variables_reorder_str {
3110     \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3111   }
3112   \int_zero:N \l_tmpa_int
3113   \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3114   \tl_clear:N \l_tmpa_tl
3115   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3116     \int_incr:N \l_tmpa_int
3117     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3118     \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3119     \str_if_eq:VnTF \l_tmpb_str a {
3120       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3121         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3122         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3123       } }
3124     }{
3125       \str_if_eq:VnTF \l_tmpb_str B {
3126         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3127           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3128           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3129         } }
3130       }{
3131         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3132           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3133         } }
3134       }
3135     }
3136   }
3137   \stex_annotate_invisible:nnn { notationcomp }{}{
3138     \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3139     $ \exp_args:Nno \use:nn { \use:c {
3140       stex_var_notation_\l__stex_variables_name_str _cs
3141     } } { \l_tmpa_tl } $
3142   }
3143   \tl_if_empty:NF \l__stex_variables_op_tl {
3144     \stex_annotate_invisible:nnn { notationopcomp }{}{
3145       ${\l__stex_variables_op_tl$
3146     }
3147   }
3148   }
3149   \str_if_empty:NF \l__stex_variables_bind_str {
3150     \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3151   }
3152   }\ignorespacesandpars
3153 }
3154
3155 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3156 }

```

```

3157
3158 \cs_new:Nn \_stex_reset:N {
3159   \tl_if_exist:NTF #1 {
3160     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3161   }{
3162     \let \exp_not:N #1 \exp_not:N \undefined
3163   }
3164 }
3165
3166 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3167   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3168   \exp_args:Nnx \use:nn {
3169     % TODO
3170     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3171       #2
3172     }
3173   }{
3174     \_stex_reset:N \varnot
3175     \_stex_reset:N \vartype
3176     \_stex_reset:N \vardefi
3177   }
3178 }
3179
3180 \NewDocumentCommand \vardef { s } {
3181   \IfBooleanTF#1 {
3182     \__stex_variables_do_complex:nn
3183   }{
3184     \__stex_variables_do_simple:nnn
3185   }
3186 }
3187
3188 \NewDocumentCommand \svar { 0{} m }{
3189   \tl_if_empty:nTF {#1}{
3190     \str_set:Nn \l_tmpa_str { #2 }
3191   }{
3192     \str_set:Nn \l_tmpa_str { #1 }
3193   }
3194   \_stex_term_omv:nn {
3195     var://\l_tmpa_str
3196   }{
3197     \exp_args:Nnx \use:nn {
3198       \def\comp{\_varcomp}
3199       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3200       \comp{ #2 }
3201     }{
3202       \_stex_reset:N \comp
3203       \_stex_reset:N \STEXInternalCurrentSymbolStr
3204     }
3205   }
3206 }
3207
3208
3209
3210 \keys_define:nn { stex / varseq } {

```

```

3211 name .str_set_x:N = \l__stex_variables_name_str ,
3212 args .int_set:N = \l__stex_variables_args_int ,
3213 type .tl_set:N = \l__stex_variables_type_tl ,
3214 mid .tl_set:N = \l__stex_variables_mid_tl ,
3215 bind .choices:nn =
3216 {forall,exists}
3217 {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3218 }
3219
3220 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3221 \str_clear:N \l__stex_variables_name_str
3222 \int_set:Nn \l__stex_variables_args_int 1
3223 \tl_clear:N \l__stex_variables_type_tl
3224 \str_clear:N \l__stex_variables_bind_str
3225
3226 \keys_set:nn { stex / varseq } { #1 }
3227 }
3228
3229 \NewDocumentCommand \varseq {m O{} m m m}{
3230 \__stex_variables_seq_args:n { #2 }
3231 \str_if_empty:NT \l__stex_variables_name_str {
3232 \str_set:Nx \l__stex_variables_name_str { #1 }
3233 }
3234 \prop_clear:N \l_tmpa_prop
3235 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3236
3237 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3238 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3239 \msg_error:nnxx{stex}{error/seqlength}
3240 {\int_use:N \l__stex_variables_args_int}
3241 {\seq_count:N \l_tmpa_seq}
3242 }
3243 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3244 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3245 \msg_error:nnxx{stex}{error/seqlength}
3246 {\int_use:N \l__stex_variables_args_int}
3247 {\seq_count:N \l_tmpb_seq}
3248 }
3249 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3250 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3251
3252 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3253 \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3254
3255 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3256 \int_step_inline:nn \l__stex_variables_args_int {
3257 \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3258 }
3259 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3260 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3261 \tl_if_empty:NF \l__stex_variables_mid_tl {
3262 \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3263 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3264 }

```



```

3265 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3266 \int_step_inline:nn \l__stex_variables_args_int {
3267   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3268 }
3269 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3270 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3271
3272
3273 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3274
3275 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3276
3277 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3278
3279 \int_step_inline:nn \l__stex_variables_args_int {
3280   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3281     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3282   }}
3283 }
3284
3285 \tl_set:Nx \l_tmpa_tl {
3286   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3287     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3288   }
3289 }
3290
3291 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3292
3293 \exp_args:Nno \use:nn {
3294   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3295   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3296
3297   \stex_debug:nn{sequences}{New~Sequence:~
3298     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3299     \prop_to_keyval:N \l_tmpa_prop
3300   }
3301   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3302     \tl_if_empty:NF \l__stex_variables_type_tl {
3303       \stex_annotate:nnn {type}{\l__stex_variables_type_tl$}
3304     }
3305     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3306     \str_if_empty:NF \l__stex_variables_bind_str {
3307       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3308     }
3309     \stex_annotate:nnn{startindex}{\l__stex_variables_startindex}{\l__stex_variables_startindex}
3310     \stex_annotate:nnn{endindex}{\l__stex_variables_endindex}{\l__stex_variables_endindex}
3311   }
3312   \tl_clear:N \l_tmpa_tl
3313   \int_step_inline:nn \l__stex_variables_args_int {
3314     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3315       \stex_annotate:nnn{argmarker}{##1}{}
3316     } }
3317   }
3318   \stex_annotate_invisible:nnn { notationcomp }{}{

```

```

3319     \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3320     $ \exp_args:Nno \use:nn { \use:c {
3321         stex_varseq_\l__stex_variables_name_str _cs
3322     } } { \l_tmpa_tl } $
3323 }
3324 \stex_annotate_invisible:nnn { notationopcomp }{}{
3325     $ \prop_item:Nn \l_tmpa_prop { notation } $
3326 }
3327
3328 }}
3329
3330 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3331 \ignorespacesandpars
3332 }
3333
3334 \end{package}

```

Chapter 29

STEX -Terms Implementation

```
3335 <*package>
3336
3337 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3338
3339 <@@=stex_terms>
3340
3341   Warnings and error messages
3342 \msg_new:nnn{stex}{error/nonotation}{
3343   Symbol~#1~invoked,~but~has~no~notation#2!
3344 }
3345 \msg_new:nnn{stex}{error/notationarg}{
3346   Error~in~parsing~notation~#1
3347 }
3348 \msg_new:nnn{stex}{error/noop}{
3349   Symbol~#1~has~no~operator~notation~for~notation~#2
3350 }
3351 \msg_new:nnn{stex}{error/notallowed}{
3352   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3353 }
3354 \msg_new:nnn{stex}{error/doubleargument}{
3355   Argument~#1~of~symbol~#2~already~assigned
3356 }
3357 \msg_new:nnn{stex}{error/overarity}{
3358   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3359 }
```

29.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3359
3360
3361 \bool_new:N \l_stex_allow_semantic_bool
3362 \bool_set_true:N \l_stex_allow_semantic_bool
3363
```

```

3364 \cs_new_protected:Nn \stex_invoke_symbol:n {
3365   \ifvmode\indent\fi
3366   \bool_if:NTF \l_stex_allow_semantic_bool {
3367     \str_if_eq:eeF {
3368       \prop_item:cn {
3369         l_stex_symdecl_#1_prop
3370       }{ deprecate }
3371     }{}{
3372       \msg_warning:nxxx{stex}{warning/deprecated}{
3373         Symbol~#1
3374       }{
3375         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3376       }
3377     }
3378     \if_mode_math:
3379       \exp_after:wN \__stex_terms_invoke_math:n
3380     \else:
3381       \exp_after:wN \__stex_terms_invoke_text:n
3382     \fi: { #1 }
3383   }{
3384     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3385   }
3386 }
3387
3388 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3389   \peek_charcode_remove:NTF ! {
3390     \__stex_terms_invoke_op_custom:nn {#1}
3391   }{
3392     \__stex_terms_invoke_custom:nn {#1}
3393   }
3394 }
3395
3396 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3397   \peek_charcode_remove:NTF ! {
3398     % operator
3399     \peek_charcode_remove:NTF * {
3400       % custom op
3401       \__stex_terms_invoke_op_custom:nn {#1}
3402     }{
3403       % op notation
3404       \peek_charcode:NTF [ {
3405         \__stex_terms_invoke_op_notation:nw {#1}
3406       }{
3407         \__stex_terms_invoke_op_notation:nw {#1}[]
3408       }
3409     }
3410   }{
3411     \peek_charcode_remove:NTF * {
3412       \__stex_terms_invoke_custom:nn {#1}
3413       % custom
3414     }{
3415       % normal
3416       \peek_charcode:NTF [ {
3417         \__stex_terms_invoke_notation:nw {#1}

```

```

3418     }{
3419         \__stex_terms_invoke_notation:nw {#1}[]
3420     }
3421 }
3422 }
3423 }
3424
3425
3426 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3427     \exp_args:Nnx \use:nn {
3428         \def\comp{\_comp}
3429         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3430         \bool_set_false:N \l_stex_allow_semantic_bool
3431         \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3432             \comp{ #2 }
3433         }
3434     }{
3435         \stex_reset:N \comp
3436         \stex_reset:N \STEXInternalCurrentSymbolStr
3437         \bool_set_true:N \l_stex_allow_semantic_bool
3438     }
3439 }
3440
3441 \keys_define:nn { stex / terms } {
3442     % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3443     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3444     unknown .code:n = \str_set:Nx
3445         \l_stex_notation_variant_str \l_keys_key_str
3446 }
3447
3448 \cs_new_protected:Nn \__stex_terms_args:n {
3449     % \str_clear:N \l_stex_notation_lang_str
3450     \str_clear:N \l_stex_notation_variant_str
3451
3452     \keys_set:nn { stex / terms } { #1 }
3453 }
3454
3455 \cs_new_protected:Nn \stex_find_notation:nn {
3456     \__stex_terms_args:n { #2 }
3457     \seq_if_empty:cTF {
3458         l_stex_symdecl_ #1 _notations
3459     } {
3460         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3461     } {
3462         \str_if_empty:NTF \l_stex_notation_variant_str {
3463             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3464         }{
3465             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3466                 \l_stex_notation_variant_str
3467             }{
3468                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3469             }{
3470                 \msg_error:nnxx{stex}{error/nonotation}{#1}{
3471                     ~\l_stex_notation_variant_str

```

```

3472     }
3473   }
3474 }
3475 }
3476 }
3477
3478 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3479   \exp_args:Nnx \use:nn {
3480     \def\comp{\_comp}
3481     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3482     \stex_find_notation:nn { #1 }{ #2 }
3483     \bool_set_false:N \l_stex_allow_semantic_bool
3484     \cs_if_exist:cTF {
3485       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3486     }{
3487       \_stex_term_oms:nnn { #1 }{
3488         #1 \c_hash_str \l_stex_notation_variant_str
3489       }{
3490         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3491       }
3492     }{
3493       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3494         \cs_if_exist:cTF {
3495           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3496         }{
3497           \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3498             \_stex_reset:N \comp
3499             \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3500             \_stex_reset:N \STEXInternalCurrentSymbolStr
3501             \bool_set_true:N \l_stex_allow_semantic_bool
3502           }
3503           \def\comp{\_comp}
3504           \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3505           \bool_set_false:N \l_stex_allow_semantic_bool
3506           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3507         }{
3508           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3509             ~\l_stex_notation_variant_str
3510           }
3511         }
3512       }{
3513         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3514       }
3515     }
3516   }{
3517     \_stex_reset:N \comp
3518     \_stex_reset:N \STEXInternalCurrentSymbolStr
3519     \bool_set_true:N \l_stex_allow_semantic_bool
3520   }
3521 }
3522
3523 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3524   \stex_find_notation:nn { #1 }{ #2 }
3525   \cs_if_exist:cTF {

```

```

3526     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3527   }{
3528     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3529       \_stex_reset:N \comp
3530       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3531       \_stex_reset:N \STEXInternalCurrentSymbolStr
3532       \bool_set_true:N \l_stex_allow_semantic_bool
3533     }
3534     \def\comp{\_comp}
3535     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3536     \bool_set_false:N \l_stex_allow_semantic_bool
3537     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3538   }{
3539     \msg_error:nnxx{stex}{error/nonotation}{#1}{
3540       ~\l_stex_notation_variant_str
3541     }
3542   }
3543 }
3544
3545 \prop_new:N \l__stex_terms_custom_args_prop
3546
3547 \cs_new_protected:Nn \__stex_terms_custom_comp:n { \bool_set_false:N \l_stex_allow_semantic_bool }
3548
3549 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3550   \exp_args:Nnx \use:nn {
3551     \def\comp{\_stex_terms_custom_comp:n}
3552     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3553     \prop_clear:N \l__stex_terms_custom_args_prop
3554     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3555     \prop_get:cnN {
3556       l_stex_symdecl_#1 _prop
3557     } { args } \l_tmpa_str
3558     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3559     \tl_set:Nn \arg { \_stex_terms_arg: }
3560     \str_if_empty:NTF \l_tmpa_str {
3561       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3562     }{
3563       \str_if_in:NnTF \l_tmpa_str b {
3564         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3565       }{
3566         \str_if_in:NnTF \l_tmpa_str B {
3567           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3568         }{
3569           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3570         }
3571       }
3572     }
3573     % TODO check that all arguments exist
3574   }{
3575     \_stex_reset:N \STEXInternalCurrentSymbolStr
3576     \_stex_reset:N \arg
3577     \_stex_reset:N \comp
3578     \_stex_reset:N \l__stex_terms_custom_args_prop
3579     %\bool_set_true:N \l_stex_allow_semantic_bool

```

```

3580 }
3581 }
3582
3583 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3584   \tl_if_empty:nTF {#2}{
3585     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3586     \bool_set_true:N \l_tmpa_bool
3587     \bool_do_while:Nn \l_tmpa_bool {
3588       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3589         \int_incr:N \l_tmpa_int
3590       }{
3591         \bool_set_false:N \l_tmpa_bool
3592       }
3593     }
3594   }{
3595     \int_set:Nn \l_tmpa_int { #2 }
3596   }
3597   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3598   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3599     \msg_error:nnxxx{stex}{error/overarity}
3600     {\int_use:N \l_tmpa_int}
3601     {\STEXInternalCurrentSymbolStr}
3602     {\str_count:N \l_tmpa_str}
3603   }
3604   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3605   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3606     \bool_lazy_any:nF {
3607       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3608       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3609     }{
3610       \msg_error:nnxx{stex}{error/doubleargument}
3611       {\int_use:N \l_tmpa_int}
3612       {\STEXInternalCurrentSymbolStr}
3613     }
3614   }
3615   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3616   \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
3617     \bool_set_true:N \l_stex_allow_semantic_bool
3618     \use:nn
3619   }
3620   {
3621     \IfBooleanTF#1{
3622       \stex_annotate_invisible:n { %TODO
3623         \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3624       }
3625     }{ %TODO
3626       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3627     }}
3628     {\bool_set_false:N \l_stex_allow_semantic_bool}
3629   }
3630
3631
3632 \cs_new_protected:Nn \_stex_term_arg:nn {
3633   \bool_set_true:N \l_stex_allow_semantic_bool

```



```

3634 \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3635 \bool_set_false:N \l_stex_allow_semantic_bool
3636 }
3637
3638 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
3639 \exp_args:Nnx \use:nn
3640 { \int_set:Nn \l__stex_terms_downprec { #2 }
3641 \stex_term_arg:nn { #1 }{ #3 }
3642 }
3643 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3644 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 84.)

`\STEXInternalTermMathAssocArgiiii`

```

3645 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4 {
3646 \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3647 \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#1}{#2}}
3648 \tl_if_empty:nTF { #3 }{
3649 \STEXInternalTermMathArgiii{#1}{#2}{ }
3650 }{
3651 \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3652 \expandafter\if\expandafter\relax\noexpand#3
3653 \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3654 \else
3655 \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3656 \fi
3657 \l_tmpa_tl
3658 }{
3659 \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3660 }
3661 }
3662 }
3663
3664 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3665 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3666 \str_if_empty:NNTF \l_tmpa_str {
3667 \exp_args:Nx \cs_if_eq:NNTF {
3668 \tl_head:N #1
3669 } \stex_invoke_sequence:n {
3670 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3671 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3672 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\l_tmpa_str _prop}{notation}}
3673 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3674 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3675 \exp_not:n{\exp_args:Nnx \use:nn} {
3676 \exp_not:n {
3677 \def\comp{\_varcomp}
3678 \str_set:Nn \STEXInternalCurrentSymbolStr
3679 } {varseq://\l_tmpa_str}
3680 \exp_not:n{ ##1 }
3681 }{
3682 \exp_not:n {
3683 \stex_reset:N \comp

```

```

3684         \_stex_reset:N \STEXInternalCurrentSymbolStr
3685     }
3686 }
3687 }}}
3688 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3689 \seq_reverse:N \l_tmpa_seq
3690 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3691 \seq_map_inline:Nn \l_tmpa_seq {
3692     \exp_args:NNno \exp_args:Nno \tl_set:No \l_tmpa_tl {
3693         \exp_args:Nno
3694         \l_tmpa_cs { ##1 } \l_tmpa_tl
3695     }
3696 }
3697 \tl_set:Nx \l_tmpa_tl {
3698     \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3699         \exp_args:No \exp_not:n \l_tmpa_tl
3700     }
3701 }
3702 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3703 }{
3704     \_stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3705 }
3706 } {
3707     \_stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3708 }
3709 }
3710 }
3711
3712 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nn {
3713     \clist_set:Nn \l_tmpa_clist{ #2 }
3714     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3715         \tl_set:Nn \l_tmpa_tl { \_stex_term_arg:nn{A#1}{ #2 } }
3716     }{
3717         \clist_reverse:N \l_tmpa_clist
3718         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3719         \tl_set:Nx \l_tmpa_tl { \_stex_term_arg:nn{A#1}{
3720             \exp_args:No \exp_not:n \l_tmpa_tl
3721         }}
3722         \clist_map_inline:Nn \l_tmpa_clist {
3723             \exp_args:NNno \exp_args:Nno \tl_set:No \l_tmpa_tl {
3724                 \exp_args:Nno
3725                 \l_tmpa_cs { \_stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3726             }
3727         }
3728     }
3729     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3730 }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 85.)

29.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3731 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3732 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3733 \int_new:N \l__stex_terms_downprec
3734 \int_set_eq:NN \l__stex_terms_downprec \infprec

(End definition for \infprec, \neginfprec, and \l__stex_terms_downprec. These variables are docu-
mented on page 85.)

Bracketing:

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3735 \tl_set:Nn \l__stex_terms_left_bracket_str (
3736 \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
3737 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3738   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3739     \bool_set_false:N \l__stex_terms_brackets_done_bool
3740     #2
3741   } {
3742     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3743       \bool_if:NTF \l__stex_inarray_bool { #2 }{
3744         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3745         \dobrackets { #2 }
3746       }
3747     }{ #2 }
3748   }
3749 }

(End definition for \__stex_terms_maybe_brackets:nn.)

\dobrackets
3750 \bool_new:N \l__stex_terms_brackets_done_bool
3751 %\RequirePackage{scalerel}
3752 \cs_new_protected:Npn \dobrackets #1 {
3753   %\ThisStyle{\if D\m@switch
3754   %   \exp_args:Nnx \use:nn
3755   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3756   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3757   % \else
3758   \exp_args:Nnx \use:nn
3759   {
3760     \bool_set_true:N \l__stex_terms_brackets_done_bool
3761     \int_set:Nn \l__stex_terms_downprec \infprec
3762     \l__stex_terms_left_bracket_str
3763     #1
3764   }
3765   {
3766     \bool_set_false:N \l__stex_terms_brackets_done_bool
3767     \l__stex_terms_right_bracket_str
3768     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3769   }
3770   %\fi}
3771 }

```

(End definition for `\dobrackets`. This function is documented on page 85.)

`\withbrackets`

```

3772 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3773   \exp_args:Nnx \use:nn
3774   {
3775     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3776     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3777     #3
3778   }
3779   {
3780     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3781     {\l__stex_terms_left_bracket_str}
3782     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3783     {\l__stex_terms_right_bracket_str}
3784   }
3785 }

```

(End definition for `\withbrackets`. This function is documented on page 85.)

`\STEXinvisible`

```

3786 \cs_new_protected:Npn \STEXinvisible #1 {
3787   \stex_annotate_invisible:n { #1 }
3788 }

```

(End definition for `\STEXinvisible`. This function is documented on page 85.)

OMDoc terms:

`\STEXInternalTermMathOMSiiii`

```

3789 \cs_new_protected:Nn \_stex_term_oms:nnn {
3790   \stex_annotate:nnn{ OMID }{ #2 }{
3791     #3
3792   }
3793 }
3794
3795 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
3796   \__stex_terms_maybe_brackets:nn { #3 }{
3797     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3798   }
3799 }

```

(End definition for `\STEXInternalTermMathOMSiiii`. This function is documented on page 84.)

`_stex_term_math_omv:nn`

```

3800 \cs_new_protected:Nn \_stex_term_omv:nn {
3801   \stex_annotate:nnn{ OMV }{ #1 }{
3802     #2
3803   }
3804 }

```

(End definition for `_stex_term_math_omv:nn`. This function is documented on page ??.)

`\STEXInternalTermMathOMAi`

```

3805 \cs_new_protected:Nn \stex_term_oma:nnn {
3806   \stex_annotate:nnn{ OMA }{ #2 }{
3807     #3
3808   }
3809 }
3810
3811 \cs_new_protected:Npn \STEXInternalTermMathOMAi #1#2#3#4 {
3812   \__stex_terms_maybe_brackets:nn { #3 }{
3813     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3814   }
3815 }

```

(End definition for `\STEXInternalTermMathOMAi`. This function is documented on page 84.)

`\STEXInternalTermMathOMBi`

```

3816 \cs_new_protected:Nn \stex_term_ombind:nnn {
3817   \stex_annotate:nnn{ OMBIND }{ #2 }{
3818     #3
3819   }
3820 }
3821
3822 \cs_new_protected:Npn \STEXInternalTermMathOMBi #1#2#3#4 {
3823   \__stex_terms_maybe_brackets:nn { #3 }{
3824     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3825   }
3826 }

```

(End definition for `\STEXInternalTermMathOMBi`. This function is documented on page 84.)

`\symref`

`\symname`

```

3827 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3828
3829 \keys_define:nn { stex / symname } {
3830   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3831   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3832   root     .tl_set_x:N      = \l__stex_terms_root_tl
3833 }
3834
3835 \cs_new_protected:Nn \stex_symname_args:n {
3836   \tl_clear:N \l__stex_terms_post_tl
3837   \tl_clear:N \l__stex_terms_pre_tl
3838   \tl_clear:N \l__stex_terms_root_str
3839   \keys_set:nn { stex / symname } { #1 }
3840 }
3841
3842 \NewDocumentCommand \symref { m m }{
3843   \let\compemph_uri_prev:\compemph_uri
3844   \let\compemph_uri\symrefemph_uri
3845   \STEXsymbol{#1}!{ #2 }
3846   \let\compemph_uri\compemph_uri_prev:
3847 }
3848
3849 \NewDocumentCommand \synonym { 0{ } m m }{

```

```

3850 \stex_symname_args:n { #1 }
3851 \let\compemph_uri_prev:\compemph@uri
3852 \let\compemph@uri\symrefemph@uri
3853 % TODO
3854 \STEXsymbol{#2}!\{ \l__stex_terms_pre_tl #3 \l__stex_terms_post_tl }
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
3866 \let\compemph_uri_prev:\compemph@uri
3867 \let\compemph@uri\symrefemph@uri
3868 \exp_args:NNx \use:nn
3869 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3870 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3871 } }
3872 \let\compemph@uri\compemph_uri_prev:
3873 }
3874
3875 \NewDocumentCommand \Symname { 0{ } m }{
3876 \stex_symname_args:n { #1 }
3877 \stex_get_symbol:n { #2 }
3878 \str_set:Nx \l_tmpa_str {
3879 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3880 }
3881 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3882 \let\compemph_uri_prev:\compemph@uri
3883 \let\compemph@uri\symrefemph@uri
3884 \exp_args:NNx \use:nn
3885 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3886 \exp_after:wN \stex_capitalize:n \l_tmpa_str
3887 \l__stex_terms_post_tl
3888 } }
3889 \let\compemph@uri\compemph_uri_prev:
3890 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 84.)

29.3 Notation Components

```

3891 <@@=stex_notationcomps>

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\symrefemph@uri
\varemp
\varemp@uri

```

```

3892 \cs_new_protected:Npn \_comp #1 {
3893 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3894 \stex_html_backend:TF {
3895 \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3896 }{

```

```

3897     \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3898   }
3899 }
3900 }
3901
3902 \cs_new_protected:Npn \_varcomp #1 {
3903   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3904     \stex_html_backend:TF {
3905       \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3906     }{
3907       \exp_args:Nnx \varempemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3908     }
3909   }
3910 }
3911
3912 \def\comp{\_comp}
3913
3914 \cs_new_protected:Npn \compemph@uri #1 #2 {
3915   \compemph{ #1 }
3916 }
3917
3918
3919 \cs_new_protected:Npn \compemph #1 {
3920   #1
3921 }
3922
3923 \cs_new_protected:Npn \defemph@uri #1 #2 {
3924   \defemph{#1}
3925 }
3926
3927 \cs_new_protected:Npn \defemph #1 {
3928   \textbf{#1}
3929 }
3930
3931 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3932   \symrefemph{#1}
3933 }
3934
3935 \cs_new_protected:Npn \symrefemph #1 {
3936   \emph{#1}
3937 }
3938
3939 \cs_new_protected:Npn \varempemph@uri #1 #2 {
3940   \varempemph{#1}
3941 }
3942
3943 \cs_new_protected:Npn \varempemph #1 {
3944   #1
3945 }

```

(End definition for `\comp` and others. These functions are documented on page 85.)

\ellipses

```

3946 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 85.)

```

\parray
\prmatrix 3947 \bool_new:N \l_stex_inpparray_bool
\parrayline 3948 \bool_set_false:N \l_stex_inpparray_bool
\parraylineh 3949 \NewDocumentCommand \parray { m m } {
\parraycell 3950 \begin{group}
3951 \bool_set_true:N \l_stex_inpparray_bool
3952 \begin{array}{#1}
3953 #2
3954 \end{array}
3955 \end{group}
3956 }
3957
3958 \NewDocumentCommand \prmatrix { m } {
3959 \begin{group}
3960 \bool_set_true:N \l_stex_inpparray_bool
3961 \begin{matrix}
3962 #1
3963 \end{matrix}
3964 \end{group}
3965 }
3966
3967 \def \maybepline {
3968 \bool_if:NT \l_stex_inpparray_bool {\hline}
3969 }
3970
3971 \def \parrayline #1 #2 {
3972 #1 #2 \bool_if:NT \l_stex_inpparray_bool {\}
3973 }
3974
3975 \def \pmrow #1 { \parrayline{}{ #1 } }
3976
3977 \def \parraylineh #1 #2 {
3978 #1 #2 \bool_if:NT \l_stex_inpparray_bool {\hline}
3979 }
3980
3981 \def \parraycell #1 {
3982 #1 \bool_if:NT \l_stex_inpparray_bool {&}
3983 }

```

(End definition for `\parray` and others. These functions are documented on page ??.)

29.4 Variables

3984 `\@@=stex_variables`

`\stex_invoke_variable:n` Invokes a variable

```

3985 \cs_new_protected:Nn \stex_invoke_variable:n {
3986 \if_mode_math:
3987 \exp_after:wN \__stex_variables_invoke_math:n
3988 \else:
3989 \exp_after:wN \__stex_variables_invoke_text:n

```



```

3990 \fi: {#1}
3991 }
3992
3993 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3994 \peek_charcode_remove:NTF ! {
3995 \__stex_variables_invoke_op_custom:nn {#1}
3996 }{
3997 \__stex_variables_invoke_custom:nn {#1}
3998 }
3999 }
4000
4001
4002 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
4003 \peek_charcode_remove:NTF ! {
4004 \peek_charcode_remove:NTF ! {
4005 \peek_charcode:NTF [ {
4006 % TODO throw error
4007 }{
4008 \__stex_variables_invoke_op_custom:nn
4009 }
4010 }{
4011 \__stex_variables_invoke_op:n { #1 }
4012 }
4013 }{
4014 \peek_charcode_remove:NTF * {
4015 \__stex_variables_invoke_custom:nn { #1 }
4016 }{
4017 \__stex_variables_invoke_math_ii:n { #1 }
4018 }
4019 }
4020 }
4021
4022 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4023 \exp_args:Nnx \use:nn {
4024 \def\comp{\_varcomp}
4025 \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4026 \bool_set_false:N \l_stex_allow_semantic_bool
4027 \stex_term_omv:nn {var://#1}{
4028 \comp{ #2 }
4029 }
4030 }{
4031 \stex_reset:N \comp
4032 \stex_reset:N \STEXInternalCurrentSymbolStr
4033 \bool_set_true:N \l_stex_allow_semantic_bool
4034 }
4035 }
4036
4037 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4038 \cs_if_exist:cTF {
4039 stex_var_op_notation_ #1 _cs
4040 }{
4041 \exp_args:Nnx \use:nn {
4042 \def\comp{\_varcomp}
4043 \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }

```

```

4044     \stex_term_omv:nn { var://#1 }{
4045       \use:c{stex_var_op_notation_ #1 _cs }
4046     }
4047   }{
4048     \stex_reset:N \comp
4049     \stex_reset:N \STEXInternalCurrentSymbolStr
4050   }
4051 }{
4052   \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
4053     \__stex_variables_invoke_math_ii:n {#1}
4054   }{
4055     \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
4056   }
4057 }
4058 }
4059
4060 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4061   \cs_if_exist:cTF {
4062     stex_var_notation_#1_cs
4063   }{
4064     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4065       \stex_reset:N \comp
4066       \stex_reset:N \STEXInternalSymbolAfterInvokationTL
4067       \stex_reset:N \STEXInternalCurrentSymbolStr
4068       \bool_set_true:N \l_stex_allow_semantic_bool
4069     }
4070     \def\comp{\_varcomp}
4071     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4072     \bool_set_false:N \l_stex_allow_semantic_bool
4073     \use:c{stex_var_notation_#1_cs}
4074   }{
4075     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
4076   }
4077 }
4078
4079 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4080   \exp_args:Nnx \use:nn {
4081     \def\_varcomp{
4082       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4083       \prop_clear:N \l__stex_terms_custom_args_prop
4084       \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4085       \prop_get:cnN {
4086         l_stex_variable_#1_prop
4087       }{ args } \l_tmpa_str
4088       \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4089       \tl_set:Nn \arg { \__stex_terms_arg: }
4090       \str_if_empty:NTF \l_tmpa_str {
4091         \stex_term_omv:nn {var://#1}{\ignorespaces#2}
4092       }{
4093         \str_if_in:NnTF \l_tmpa_str b {
4094           \stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4095         }{
4096           \str_if_in:NnTF \l_tmpa_str B {
4097             \stex_term_ombind:nnn {var://#1}{\ignorespaces#2}

```

```

4098     }{
4099       \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4100     }
4101   }
4102 }
4103 % TODO check that all arguments exist
4104 }{
4105   \_stex_reset:N \STEXInternalCurrentSymbolStr
4106   \_stex_reset:N \arg
4107   \_stex_reset:N \comp
4108   \_stex_reset:N \l__stex_terms_custom_args_prop
4109   %\bool_set_true:N \l_stex_allow_semantic_bool
4110 }
4111 }

```

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

29.5 Sequences

```

4112 <@@=stex_sequences>
4113
4114 \cs_new_protected:Nn \stex_invoke_sequence:n {
4115   \peek_charcode_remove:NTF ! {
4116     \_stex_term_omv:nn {varseq://#1}{
4117       \exp_args:Nnx \use:nn {
4118         \def\comp{\_varcomp}
4119         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4120         \prop_item:cn{stex_varseq_#1_prop}{notation}
4121       }{
4122         \_stex_reset:N \comp
4123         \_stex_reset:N \STEXInternalCurrentSymbolStr
4124       }
4125     }
4126   }{
4127     \bool_set_false:N \l_stex_allow_semantic_bool
4128     \def\comp{\_varcomp}
4129     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4130     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4131       \_stex_reset:N \comp
4132       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4133       \_stex_reset:N \STEXInternalCurrentSymbolStr
4134       \bool_set_true:N \l_stex_allow_semantic_bool
4135     }
4136     \use:c { stex_varseq_#1_cs }
4137   }
4138 }
4139 </package>

```

Chapter 30

STEX -Structural Features Implementation

```
4140 ⟨*package⟩
4141
4142 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4143
4144 Warnings and error messages
4145 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4146   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4147 }
4148 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
4149   Symbol~#1~not~assigned~in~interpretmodule~#2
4150 }
4151 \msg_new:nnn{stex}{error/unknownstructure}{
4152   No~structure~#1~found!
4153 }
4154 \msg_new:nnn{stex}{error/unknownfield}{
4155   No~field~#1~in~instance~#2~found!\#3
4156 }
4157 }
4158 \msg_new:nnn{stex}{error/keyval}{
4159   Invalid~key=value~pair:#1
4160 }
4161 \msg_new:nnn{stex}{error/instantiate/missing}{
4162   Assignments~missing~in~instantiate:~#1
4163 }
4164 \msg_new:nnn{stex}{error/incompatible}{
4165   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4166 }
4167 }
4168
```

30.1 Imports with modification

```

4169 <@@=stex_copymodule>
4170 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4171   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4172     \tl_set:Nn \l_tmpa_tl { #1 }
4173     \__stex_copymodule_get_symbol_from_cs:
4174   }{
4175     % argument is a string
4176     % is it a command name?
4177     \cs_if_exist:cTF { #1 }{
4178       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4179       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4180       \str_if_empty:NTF \l_tmpa_str {
4181         \exp_args:Nx \cs_if_eq:NNTF {
4182           \tl_head:N \l_tmpa_tl
4183         } \stex_invoke_symbol:n {
4184           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4185         }{
4186           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4187         }
4188       } {
4189         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4190       }
4191     }{
4192       % argument is not a command name
4193       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4194       % \l_stex_all_symbols_seq
4195     }
4196   }
4197 }
4198
4199 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4200   \str_set:Nn \l_tmpa_str { #1 }
4201   \bool_set_false:N \l_tmpa_bool
4202   \bool_if:NF \l_tmpa_bool {
4203     \tl_set:Nn \l_tmpa_tl {
4204       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4205     }
4206     \str_set:Nn \l_tmpa_str { #1 }
4207     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4208     \seq_map_inline:Nn #2 {
4209       \str_set:Nn \l_tmpb_str { ##1 }
4210       \str_if_eq:eeT { \l_tmpa_str } {
4211         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4212       } {
4213         \seq_map_break:n {
4214           \tl_set:Nn \l_tmpa_tl {
4215             \str_set:Nn \l_stex_get_symbol_uri_str {
4216               ##1
4217             }
4218           }
4219         }
4220       }

```

```

4221     }
4222     \l_tmpa_tl
4223   }
4224 }
4225
4226 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4227   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4228     { \tl_tail:N \l_tmpa_tl }
4229   \tl_if_single:NTF \l_tmpa_tl {
4230     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4231       \exp_after:wN \str_set:Nn \exp_after:wN
4232         \l_stex_get_symbol_uri_str \l_tmpa_tl
4233       \__stex_copymodule_get_symbol_check:n { #1 }
4234     }{
4235       % TODO
4236       % tail is not a single group
4237     }
4238   }{
4239     % TODO
4240     % tail is not a single group
4241   }
4242 }
4243
4244 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4245   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4246     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4247       :~\seq_use:Nn #1 {,~}
4248     }
4249   }
4250 }
4251
4252 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4253   % import module
4254   \stex_import_module_uri:nn { #1 } { #2 }
4255   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4256   \stex_import_require_module:nnnn
4257     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4258     { \l_stex_import_path_str } { \l_stex_import_name_str }
4259
4260   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4261   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4262
4263   % fields
4264   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4265   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4266     \seq_map_inline:cn {c_stex_module_###1_constants}{
4267       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4268         ###1 ? #####1
4269       }
4270     }
4271   }
4272
4273   % setup prop
4274   \seq_clear:N \l_tmpa_seq

```

```

4275 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4276   name      = \l_stex_current_copymodule_name_str ,
4277   module    = \l_stex_current_module_str ,
4278   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4279   includes  = \l_tmpa_seq %,
4280 % fields    = \l_tmpa_seq
4281 }
4282 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4283   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4284 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4285 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4286
4287 \stex_if_do_html:T {
4288   \begin{stex_annotate_env} {#4} {
4289     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4290   }
4291   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4292 }
4293 }
4294
4295 \cs_new_protected:Nn \stex_copymodule_end:n {
4296   % apply to every field
4297   \def \l_tmpa_cs ##1 ##2 {#1}
4298
4299   \tl_clear:N \__stex_copymodule_module_tl
4300   \tl_clear:N \__stex_copymodule_exec_tl
4301
4302   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4303   \seq_clear:N \__stex_copymodule_fields_seq
4304
4305   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4306     \seq_map_inline:cn {c_stex_module_##1_constants}{
4307
4308       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4309       \l_tmpa_cs{##1}{####1}
4310
4311       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4312         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4313         \stex_if_do_html:T {
4314           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4315             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4316           }
4317         }
4318       }{
4319         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4320       }
4321
4322       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4323       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4324       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4325
4326       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4327         \stex_if_do_html:T {
4328           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4329         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4330     }
4331 }
4332 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4333 }
4334
4335 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4336 \tl_put_right:Nx \__stex_copymodule_module_tl {
4337     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4338     \prop_set_from_keyval:cn {
4339         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4340     }{
4341         \prop_to_keyval:N \l_tmpa_prop
4342     }
4343 }
4344
4345 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4346     \stex_if_do_html:T {
4347         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4348             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4349         }
4350     }
4351     \tl_put_right:Nx \__stex_copymodule_module_tl {
4352         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4353             \stex_invoke_symbol:n {
4354                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4355             }
4356         }
4357     }
4358 }
4359
4360 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4361
4362 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4363     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4364 }
4365
4366 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4367     \stex_if_do_html:TF{
4368         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4369     }{
4370         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4371     }
4372 }
4373 }
4374 }
4375
4376
4377 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4378 \tl_put_left:Nx \__stex_copymodule_module_tl {
4379     \prop_set_from_keyval:cn {
4380         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4381     }{
4382         \prop_to_keyval:N \l_stex_current_copymodule_prop

```



```

4383     }
4384   }
4385
4386   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4387     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4388   }
4389
4390   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4391   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4392   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4393
4394   \__stex_copymodule_exec_tl
4395   \stex_if_do_html:T {
4396     \end{stex_annotate_env}
4397   }
4398 }
4399
4400 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4401   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4402   \stex_deactivate_macro:Nn \symdecl {module~environments}
4403   \stex_deactivate_macro:Nn \symdef {module~environments}
4404   \stex_deactivate_macro:Nn \notation {module~environments}
4405   \stex_reactivate_macro:N \assign
4406   \stex_reactivate_macro:N \renamedec1
4407   \stex_reactivate_macro:N \donotcopy
4408   \stex_smsmode_do:
4409 }{
4410   \stex_copymodule_end:n {}
4411 }
4412
4413 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4414   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4415   \stex_deactivate_macro:Nn \symdecl {module~environments}
4416   \stex_deactivate_macro:Nn \symdef {module~environments}
4417   \stex_deactivate_macro:Nn \notation {module~environments}
4418   \stex_reactivate_macro:N \assign
4419   \stex_reactivate_macro:N \renamedec1
4420   \stex_reactivate_macro:N \donotcopy
4421   \stex_smsmode_do:
4422 }{
4423   \stex_copymodule_end:n {
4424     \tl_if_exist:cF {
4425       l__stex_copymodule_copymodule_##1?##2_def_tl
4426     }{
4427       \str_if_eq:eeF {
4428         \prop_item:cn{
4429           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4430       }{ true }{
4431         \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4432           ##1?##2
4433         }{\l_stex_current_copymodule_name_str}
4434       }
4435     }
4436   }

```

```

4437 }
4438
4439 \iffalse \begin{stex_annotate_env} \fi
4440 \NewDocumentEnvironment {realization} { 0 } { m } {
4441   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4442   \stex_deactivate_macro:Nn \symdecl {module~environments}
4443   \stex_deactivate_macro:Nn \symdef {module~environments}
4444   \stex_deactivate_macro:Nn \notation {module~environments}
4445   \stex_reactivate_macro:N \donotcopy
4446   \stex_reactivate_macro:N \assign
4447   \stex_smsmode_do:
4448 } {
4449   \stex_import_module_uri:nn { #1 } { #2 }
4450   \tl_clear:N \__stex_copymodule_exec_tl
4451   \tl_set:Nx \__stex_copymodule_module_tl {
4452     \stex_import_require_module:nnnn
4453     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4454     { \l_stex_import_path_str } { \l_stex_import_name_str }
4455   }
4456
4457   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4458     \seq_map_inline:cn {c_stex_module_##1_constants}{
4459       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4460       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4461         \stex_if_do_html:T {
4462           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4463             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4464               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4465             }
4466           }
4467         }
4468         \tl_put_right:Nx \__stex_copymodule_module_tl {
4469           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4470         }
4471       }
4472     }
4473
4474     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4475
4476     \__stex_copymodule_exec_tl
4477     \stex_if_do_html:T {\end{stex_annotate_env}}
4478   }
4479
4480   \NewDocumentCommand \donotcopy { m } {
4481     \str_clear:N \l_stex_import_name_str
4482     \str_set:Nn \l_tmpa_str { #1 }
4483     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4484     \seq_map_inline:Nn \l_stex_all_modules_seq {
4485       \str_set:Nn \l_tmpb_str { ##1 }
4486       \str_if_eq:eeT { \l_tmpa_str } {
4487         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4488       } {
4489         \seq_map_break:n {
4490           \stex_if_do_html:T {

```

```

4491         \stex_if_smsmode:F {
4492             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4493                 \stex_annotate:nnn{domain}{##1}{}}
4494         }
4495     }
4496 }
4497 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4498 }
4499 }
4500 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4501     \str_set:Nn \l_tmpb_str { #####1 }
4502     \str_if_eq:eeT { \l_tmpa_str } {
4503         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4504     } {
4505         \seq_map_break:n {\seq_map_break:n {
4506             \stex_if_do_html:T {
4507                 \stex_if_smsmode:F {
4508                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4509                         \stex_annotate:nnn{domain}{
4510                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4511                         }{}
4512                     }
4513                 }
4514             }
4515             \str_set:Nx \l_stex_import_name_str {
4516                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4517             }
4518         }}
4519     }
4520 }
4521 }
4522 \str_if_empty:NTF \l_stex_import_name_str {
4523     % TODO throw error
4524 }{
4525     \stex_collect_imports:n {\l_stex_import_name_str }
4526     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4527         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4528         \seq_map_inline:cn {c_stex_module_###1_constants}{
4529             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4530             \bool_lazy_any:nT {
4531                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4532                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4533                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4534             }{
4535                 % TODO throw error
4536             }
4537         }
4538     }
4539     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4540     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4541     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4542 }
4543 \stex_smsmode_do:
4544 }

```

```

4545
4546 \NewDocumentCommand \assign { m m }{
4547   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4548   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4549   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4550   \stex_smsmode_do:
4551 }
4552
4553 \keys_define:nn { stex / renamedecl } {
4554   name          .str_set_x:N = \l_stex_renamedecl_name_str
4555 }
4556 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4557   \str_clear:N \l_stex_renamedecl_name_str
4558   \keys_set:nn { stex / renamedecl } { #1 }
4559 }
4560
4561 \NewDocumentCommand \renamedecl { O{} m m }{
4562   \__stex_copymodule_renamedecl_args:n { #1 }
4563   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4564   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4565   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4566   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4567     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4568       \l_stex_get_symbol_uri_str
4569     } }
4570   } {
4571     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4572       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4573       \prop_set_eq:cc {l_stex_symdecl_
4574         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4575       _prop
4576       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4577       \seq_set_eq:cc {l_stex_symdecl_
4578         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4579       _notations
4580       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4581       \prop_put:cnx {l_stex_symdecl_
4582         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4583       _prop
4584       }{ name }{ \l_stex_renamedecl_name_str }
4585       \prop_put:cnx {l_stex_symdecl_
4586         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4587       _prop
4588       }{ module }{ \l_stex_current_module_str }
4589       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4590         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4591       }
4592       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4593         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4594       } }
4595     }
4596   \stex_smsmode_do:
4597 }
4598

```

```

4599 \stex_deactivate_macro:Nn \assign {copymodules}
4600 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4601 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4602
4603

```

30.2 The feature environment

structural@feature

```

4604 <@@=stex_features>
4605
4606 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4607   \stex_if_in_module:F {
4608     \msg_set:nnn{stex}{error/nomodule}{
4609       Structural~Feature~has~to~occur~in~a~module:\\
4610       Feature~#2~of~type~#1\\
4611       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4612     }
4613     \msg_error:nn{stex}{error/nomodule}
4614   }
4615
4616   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4617
4618   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4619
4620   \stex_if_do_html:T {
4621     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4622     \stex_annotate_invisible:nnn{header}{\{ #3 }
4623   }
4624   }{
4625     \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4626     \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4627     \stex_debug:nn{features}{
4628       Feature: \l_stex_last_feature_str
4629     }
4630     \stex_if_do_html:T {
4631       \end{stex_annotate_env}
4632     }
4633   }

```

30.3 Structure

structure

```

4634 <@@=stex_structures>
4635 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4636   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4637     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4638   }
4639   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4640   {#1}{#2}
4641 }
4642

```

```

4643 \keys_define:nn { stex / features / structure } {
4644   name          .str_set_x:N = \l__stex_structures_name_str ,
4645 }
4646
4647 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4648   \str_clear:N \l__stex_structures_name_str
4649   \keys_set:nn { stex / features / structure } { #1 }
4650 }
4651
4652 \NewDocumentEnvironment{mathstructure}{m O{}}{
4653   \__stex_structures_structure_args:n { #2 }
4654   \str_if_empty:NT \l__stex_structures_name_str {
4655     \str_set:Nx \l__stex_structures_name_str { #1 }
4656   }
4657   \stex_suppress_html:n {
4658     \bool_set_true:N \l_stex_symdecl_make_macro_bool
4659     \exp_args:Nx \stex_symdecl_do:nn {
4660       name = \l__stex_structures_name_str ,
4661       def = {\STEXsymbol{module-type}}{
4662         \STEXInternalTermMathOMSiiii {
4663           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4664             { ns } ?
4665           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4666             { name } / \l__stex_structures_name_str - structure
4667         }{}{0}{}
4668       }}
4669     }{ #1 }
4670   }
4671   \exp_args:Nnnx
4672   \begin{structural_feature_module}{ structure }
4673     { \l__stex_structures_name_str }{}
4674   \stex_smsmode_do:
4675 }{
4676   \end{structural_feature_module}
4677   \_stex_reset_up_to_module:n \l_stex_last_feature_str
4678   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4679   \seq_clear:N \l_tmpa_seq
4680   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4681     \seq_map_inline:cn{c_stex_module_##1_constants}{
4682       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4683     }
4684   }
4685   \exp_args:Nnno
4686   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4687   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4688   \stex_add_structure_to_current_module:nn
4689     \l__stex_structures_name_str
4690     \l_stex_last_feature_str
4691
4692   \stex_execute_in_module:x {
4693     \tl_set:cn { #1 }{
4694       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4695     }
4696   }

```

```

4697 }
4698
4699 \cs_new:Nn \stex_invoke_structure:nn {
4700   \stex_invoke_symbol:n { #1?#2 }
4701 }
4702
4703 \cs_new_protected:Nn \stex_get_structure:n {
4704   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4705     \tl_set:Nn \l_tmpa_tl { #1 }
4706     \__stex_structures_get_from_cs:
4707   }{
4708     \cs_if_exist:cTF { #1 }{
4709       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4710       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4711       \str_if_empty:NTF \l_tmpa_str {
4712         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4713           \__stex_structures_get_from_cs:
4714         }{
4715           \__stex_structures_get_from_string:n { #1 }
4716         }
4717       }{
4718         \__stex_structures_get_from_string:n { #1 }
4719       }
4720     }{
4721       \__stex_structures_get_from_string:n { #1 }
4722     }
4723   }
4724 }
4725
4726 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4727   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4728     { \tl_tail:N \l_tmpa_tl }
4729   \str_set:Nx \l_tmpa_str {
4730     \exp_after:wN \use_i:nn \l_tmpa_tl
4731   }
4732   \str_set:Nx \l_tmpb_str {
4733     \exp_after:wN \use_ii:nn \l_tmpa_tl
4734   }
4735   \str_set:Nx \l_stex_get_structure_str {
4736     \l_tmpa_str ? \l_tmpb_str
4737   }
4738   \str_set:Nx \l_stex_get_structure_module_str {
4739     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4740   }
4741 }
4742
4743 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4744   \tl_set:Nn \l_tmpa_tl {
4745     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4746   }
4747   \str_set:Nn \l_tmpa_str { #1 }
4748   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4749
4750   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

4751 \prop_if_exist:cT {c_stex_module_##1_structures} {
4752 \prop_map_inline:cn {c_stex_module_##1_structures} {
4753 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4754 \prop_map_break:n{\seq_map_break:n{
4755 \tl_set:Nn \l_tmpa_tl {
4756 \str_set:Nn \l_stex_get_structure_str {##1?####1}
4757 \str_set:Nn \l_stex_get_structure_module_str {####2}
4758 }
4759 }}
4760 }
4761 }
4762 }
4763 }
4764 \l_tmpa_tl
4765 }

```

\instantiate

```

4766
4767 \keys_define:nn { stex / instantiate } {
4768 name .str_set_x:N = \l__stex_structures_name_str
4769 }
4770 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4771 \str_clear:N \l__stex_structures_name_str
4772 \keys_set:nn { stex / instantiate } { #1 }
4773 }
4774
4775 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4776 \beginingroup
4777 \stex_get_structure:n {#3}
4778 \__stex_structures_instantiate_args:n { #2 }
4779 \str_if_empty:NT \l__stex_structures_name_str {
4780 \str_set:Nn \l__stex_structures_name_str { #1 }
4781 }
4782 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4783 \seq_clear:N \l__stex_structures_fields_seq
4784 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4785 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4786 \seq_map_inline:cn {c_stex_module_##1_constants}{
4787 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4788 }
4789 }
4790
4791 \tl_if_empty:nF{#5}{
4792 \seq_set_split:Nnn \l_tmpa_seq , {#5}
4793 \prop_clear:N \l_tmpa_prop
4794 \seq_map_inline:Nn \l_tmpa_seq {
4795 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4796 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4797 \msg_error:nnn{stex}{error/keyval}{##1}
4798 }
4799 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4800 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4801 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4802 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```



```

4803     \exp_args:Nxx \str_if_eq:nnF
4804     {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
4805     {\prop_item:cn{l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{args}}{
4806     \msg_error:nnxxxx{stex}{error/incompatible}
4807     {l__stex_structures_dom_str}
4808     {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
4809     {\l_stex_get_symbol_uri_str}
4810     {\prop_item:cn{l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{args}}
4811   }
4812   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4813 }
4814 }
4815
4816 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4817   \str_set:Nx \l_tmpa_str {field:l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4818   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4819
4820   \stex_add_constant_to_current_module:n {\l_tmpa_str}
4821   \stex_execute_in_module:x {
4822     \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4823     name   = \l_tmpa_str ,
4824     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4825     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4826     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4827   }
4828   \seq_clear:c {l_stex_symdecl\_l_stex_current_module_str?\l_tmpa_str _notations}
4829 }
4830
4831 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4832   \stex_find_notation:nn{##1}{}
4833   \stex_execute_in_module:x {
4834     \seq_put_right:cn {l_stex_symdecl\_l_stex_current_module_str?\l_tmpa_str _notation
4835   }
4836
4837   \stex_copy_control_sequence_ii:ccN
4838   {stex_notation\_l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4839   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4840   \l_tmpa_tl
4841   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4842
4843
4844   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4845     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4846     \stex_execute_in_module:x {
4847       \tl_set:cn
4848       {stex_op_notation\_l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4849       { \exp_args:No \exp_not:n \l_tmpa_cs}
4850     }
4851   }
4852
4853 }
4854
4855 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4856 }

```

```

4857
4858 \stex_execute_in_module:x {
4859   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4860     domain = \l_stex_get_structure_module_str ,
4861     \prop_to_keyval:N \l_tmpa_prop
4862   }
4863   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4864 }
4865 \stex_debug:nn{instantiate}{
4866   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4867   \prop_to_keyval:N \l_tmpa_prop
4868 }
4869 \exp_args:Nxx \stex_symdecl_do:nn {
4870   type={\STEXsymbol{module-type}}{
4871     \STEXInternalTermMathOMSiiii {
4872       \l_stex_get_structure_module_str
4873     }{}{0}{}
4874   }}
4875 }{\l__stex_structures_name_str}
4876 % {
4877   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4878   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4879   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4880 % }
4881 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4882 \endgroup
4883 \stex_smsmode_do:\ignorespacesandpars
4884 }
4885
4886 \cs_new_protected:Nn \stex_symbol_or_var:n {
4887   \cs_if_exist:cTF{#1}{
4888     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4889     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4890     \str_if_empty:NTF \l_tmpa_str {
4891       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4892       \stex_invoke_variable:n {
4893         \bool_set_true:N \l_stex_symbol_or_var_bool
4894         \bool_set_false:N \l_stex_instance_or_symbol_bool
4895         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4896         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4897         \str_set:Nx \l_stex_get_symbol_uri_str {
4898           \exp_after:wN \use:n \l_tmpa_tl
4899         }
4900       }{ % TODO \stex_invoke_varinstance:n
4901         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
4902           \bool_set_true:N \l_stex_symbol_or_var_bool
4903           \bool_set_true:N \l_stex_instance_or_symbol_bool
4904           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4905           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4906           \str_set:Nx \l_stex_get_symbol_uri_str {
4907             \exp_after:wN \use:n \l_tmpa_tl
4908           }
4909         }{
4910           \bool_set_false:N \l_stex_symbol_or_var_bool

```

```

4911         \stex_get_symbol:n{#1}
4912     }
4913 }
4914 }{
4915     \__stex_structures_symbolorvar_from_string:n{ #1 }
4916 }
4917 }{
4918     \__stex_structures_symbolorvar_from_string:n{ #1 }
4919 }
4920 }
4921
4922 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4923     \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4924         \bool_set_true:N \l_stex_symbol_or_var_bool
4925         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4926     }{
4927         \bool_set_false:N \l_stex_symbol_or_var_bool
4928         \stex_get_symbol:n{#1}
4929     }
4930 }
4931
4932 \keys_define:nn { stex / varinstantiate } {
4933     name .str_set_x:N = \l__stex_structures_name_str,
4934     bind .choices:nn =
4935         {forall,exists}
4936         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4937 }
4938 }
4939 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4940     \str_clear:N \l__stex_structures_name_str
4941     \str_clear:N \l__stex_structures_bind_str
4942     \keys_set:nn { stex / varinstantiate } { #1 }
4943 }
4944
4945 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4946     \begin{group}
4947         \stex_get_structure:n {#3}
4948         \__stex_structures_varinstantiate_args:n { #2 }
4949         \str_if_empty:NT \l__stex_structures_name_str {
4950             \str_set:Nn \l__stex_structures_name_str { #1 }
4951         }
4952         \stex_if_do_html:TF{
4953             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4954         }{\use:n}
4955         {
4956             \stex_if_do_html:T{
4957                 \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
4958             }
4959             \seq_clear:N \l__stex_structures_fields_seq
4960             \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4961             \seq_map_inline:Nn \l_stex_collect_imports_seq {
4962                 \seq_map_inline:cn {c_stex_module_##1_constants}{
4963                     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4964                 }

```

```

4965 }
4966 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4967 \prop_clear:N \l_tmpa_prop
4968 \tl_if_empty:nF {#5} {
4969   \seq_set_split:Nnn \l_tmpa_seq , {#5}
4970   \seq_map_inline:Nn \l_tmpa_seq {
4971     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4972     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4973       \msg_error:nnn{stex}{error/keyval}{##1}
4974     }
4975     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
4976     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4977     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq
4978     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4979     \stex_if_do_html:T{
4980       \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
4981         \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{}\l_stex_get_symbol_uri_str}{}}
4982     }
4983     \bool_if:NTF \l_stex_symbol_or_var_bool {
4984       \exp_args:Nxx \str_if_eq:nnF
4985         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4986         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{\
4987         \msg_error:nnxxx{stex}{error/incompatible}
4988         {\l__stex_structures_dom_str}
4989         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4990         {\l_stex_get_symbol_uri_str}
4991         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}}
4992     }
4993     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4994   }}{
4995     \exp_args:Nxx \str_if_eq:nnF
4996       {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4997       {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{\
4998       \msg_error:nnxxx{stex}{error/incompatible}
4999       {\l__stex_structures_dom_str}
5000       {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5001       {\l_stex_get_symbol_uri_str}
5002       {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}}
5003     }
5004     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
5005   }}
5006   }
5007 }
5008 \tl_gclear:N \g__stex_structures_aftergroup_tl
5009 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5010   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l__stex_structures_fields_seq ##1} \l__stex_structures_name_str}
5011   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5012   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5013     \stex_find_notation:nn{##1}{}
5014     \cs_gset_eq:cc{g__stex_structures_tmpa\l_tmpa_str _cs}
5015       {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5016     \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa\l_tmpa_str _cs}}
5017     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5018       \cs_gset_eq:cc {g__stex_structures_tmpa_op\l_tmpa_str _cs}

```

```

5019         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5020         \stex_debug:nn{varinstantiate}{Operator-Notation:~\cs_meaning:c{g__stex_struct
5021     }
5022 }
5023
5024 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5025     \prop_set_from_keyval:cn { \l_stex_variable_ \l_tmpa_str _prop}{
5026         name      = \l_tmpa_str ,
5027         args      = \prop_item:cn { \l_stex_symdecl_##1_prop}{args} ,
5028         arity     = \prop_item:cn { \l_stex_symdecl_##1_prop}{arity} ,
5029         assocs    = \prop_item:cn { \l_stex_symdecl_##1_prop}{assocs}
5030     }
5031     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
5032     {g__stex_structures_tmpa_\l_tmpa_str _cs}
5033     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
5034     {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5035 }
5036 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn { \l_stex_symdecl_##1_prop}{name}}{\stex_inv
5037 }
5038 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5039     \prop_set_from_keyval:cn { \l_stex_varinstance_\l__stex_structures_name_str _prop }{
5040         domain = \l_stex_get_structure_module_str ,
5041         \prop_to_keyval:N \l_tmpa_prop
5042     }
5043     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5044     \tl_set:cn { \l_stex_varinstance_\l__stex_structures_name_str _op_tl }{
5045         \exp_args:Nnx \exp_not:N \use:nn {
5046             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5047             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5048                 \exp_not:n{
5049                     \_varcomp{#4}
5050                 }
5051             }
5052         }{
5053             \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5054         }
5055     }
5056 }
5057 }
5058 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5059 \aftergroup\g__stex_structures_aftergroup_tl
5060 \endgroup
5061 \stex_smsmode_do:\ignorespacesandpars
5062 }
5063
5064 \cs_new_protected:Nn \stex_invoke_instance:n {
5065     \peek_charcode_remove:NTF ! {
5066         \stex_invoke_symbol:n{#1}
5067     }{
5068         \_stex_invoke_instance:nn {#1}
5069     }
5070 }
5071
5072

```

```

5073 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5074   \peek_charcode_remove:NTF ! {
5075     \exp_args:Nnx \use:nn {
5076       \def\comp{\_varcomp}
5077       \use:c{l_stex_varinstance_#1_op_tl}
5078     }{
5079       \_stex_reset:N \comp
5080     }
5081   }{
5082     \_stex_invoke_varinstance:nn {#1}
5083   }
5084 }
5085
5086 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5087   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5088     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5089   }{
5090     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5091     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5092       \prop_to_keyval:N \l_tmpa_prop
5093     }
5094   }
5095 }
5096
5097 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5098   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5099     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5100     \l_tmpa_tl
5101   }{
5102     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5103   }
5104 }

```

(End definition for \instantiate. This function is documented on page 33.)

\stex_invoke_structure:nnn

```

5105 % #1: URI of the instance
5106 % #2: URI of the instantiated module
5107 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5108   \tl_if_empty:nTF{ #3 }{
5109     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5110       c_stex_feature_ #2 _prop
5111     }
5112     \tl_clear:N \l_tmpa_tl
5113     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5114     \seq_map_inline:Nn \l_tmpa_seq {
5115       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5116       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5117       \cs_if_exist:cT {
5118         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5119       }{
5120         \tl_if_empty:NF \l_tmpa_tl {
5121           \tl_put_right:Nn \l_tmpa_tl {,}
5122         }

```

```

5123         \tl_put_right:Nx \l_tmpa_tl {
5124             \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5125         }
5126     }
5127 }
5128 \exp_args:No \mathstruct \l_tmpa_tl
5129 }{
5130     \stex_invoke_symbol:n{#1/#3}
5131 }
5132 }

```

(End definition for \stex_invoke_structure:nnn. This function is documented on page ??.)

```

5133 </package>

```

Chapter 31

STEX -Statements Implementation

```
5134 <*package>
5135
5136 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5137
5138 <@@=stex_statements>
    Warnings and error messages
5139

\titleemph
5140 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

31.1 Definitions

definiendum

```
5141 \keys_define:nn {stex / definiendum }{
5142   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5143   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5144   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5145   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5146 }
5147 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5148   \str_clear:N \l__stex_statements_definiendum_root_str
5149   \tl_clear:N \l__stex_statements_definiendum_post_tl
5150   \str_clear:N \l__stex_statements_definiendum_gfa_str
5151   \keys_set:nn { stex / definiendum }{ #1 }
5152 }
5153 \NewDocumentCommand \definiendum { O{} m m } {
5154   \__stex_statements_definiendum_args:n { #1 }
5155   \stex_get_symbol:n { #2 }
5156   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5157   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5158     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```



```

5159     \tl_set:Nn \l_tmpa_tl { #3 }
5160   } {
5161     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5162     \tl_set:Nn \l_tmpa_tl {
5163       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5164     }
5165   }
5166 } {
5167   \tl_set:Nn \l_tmpa_tl { #3 }
5168 }
5169
5170 % TODO root
5171 \stex_html_backend:TF {
5172   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5173 } {
5174   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5175 }
5176 }
5177 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 42.)

definame

```

5178
5179 \NewDocumentCommand \definame { 0{ } m } {
5180   \__stex_statements_definiendum_args:n { #1 }
5181   % TODO: root
5182   \stex_get_symbol:n { #2 }
5183   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5184   \str_set:Nx \l_tmpa_str {
5185     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5186   }
5187   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5188   \stex_html_backend:TF {
5189     \stex_if_do_html:T {
5190       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5191         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5192       }
5193     }
5194   } {
5195     \exp_args:Nnx \defemph@uri {
5196       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5197     } { \l_stex_get_symbol_uri_str }
5198   }
5199 }
5200 \stex_deactivate_macro:Nn \definame {definition~environments}
5201
5202 \NewDocumentCommand \Definame { 0{ } m } {
5203   \__stex_statements_definiendum_args:n { #1 }
5204   \stex_get_symbol:n { #2 }
5205   \str_set:Nx \l_tmpa_str {
5206     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5207   }
5208   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5209 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5210 \stex_html_backend:TF {
5211   \stex_if_do_html:T {
5212     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5213       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5214     }
5215   }
5216 } {
5217   \exp_args:Nnx \defemph@uri {
5218     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5219   } { \l_stex_get_symbol_uri_str }
5220 }
5221 }
5222 \stex_deactivate_macro:Nn \Definame {definition-environments}
5223
5224 \NewDocumentCommand \premise { m }{
5225   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5226 }
5227 \NewDocumentCommand \conclusion { m }{
5228   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5229 }
5230 \NewDocumentCommand \definiens { 0{} m }{
5231   \str_clear:N \l_stex_get_symbol_uri_str
5232   \tl_if_empty:nF {#1} {
5233     \stex_get_symbol:n { #1 }
5234   }
5235   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5236     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5237       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5238     }{
5239       % TODO throw error
5240     }
5241   }
5242   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5243   {\l_stex_current_module_str}{
5244     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5245   }{true}{
5246     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5247     \exp_args:Nx \stex_add_to_current_module:n {
5248       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5249     }
5250   }
5251 }
5252 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5253 }
5254
5255 \NewDocumentCommand \varbindforall {m}{
5256   \stex_symbol_or_var:n {#1}
5257   \bool_if:NTF\l_stex_symbol_or_var_bool{
5258     \stex_if_do_html:T {
5259       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5260     }
5261   }{
5262     % todo throw error

```

```

5263 }
5264 }
5265
5266 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5267 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5268 \stex_deactivate_macro:Nn \definiens {definition~environments}
5269 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5270

```

(End definition for definame. This function is documented on page 42.)

sdefinition

```

5271
5272 \keys_define:nn {stex / sdefinition }{
5273   type      .str_set_x:N = \sdefinitiontype,
5274   id        .str_set_x:N = \sdefinitionid,
5275   name      .str_set_x:N = \sdefinitionname,
5276   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5277   title     .tl_set:N    = \sdefinitiontitle
5278 }
5279 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5280   \str_clear:N \sdefinitiontype
5281   \str_clear:N \sdefinitionid
5282   \str_clear:N \sdefinitionname
5283   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5284   \tl_clear:N \sdefinitiontitle
5285   \keys_set:nn { stex / sdefinition }{ #1 }
5286 }
5287
5288 \NewDocumentEnvironment{sdefinition}{0{}}{
5289   \__stex_statements_sdefinition_args:n{ #1 }
5290   \stex_reactivate_macro:N \definiendum
5291   \stex_reactivate_macro:N \definame
5292   \stex_reactivate_macro:N \Definame
5293   \stex_reactivate_macro:N \premise
5294   \stex_reactivate_macro:N \definiens
5295   \stex_reactivate_macro:N \varbindforall
5296   \stex_if_smsmode:F{
5297     \seq_clear:N \l_tmpb_seq
5298     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5299       \tl_if_empty:nF{ ##1 }{
5300         \stex_get_symbol:n { ##1 }
5301         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5302           \l_stex_get_symbol_uri_str
5303         }
5304       }
5305     }
5306     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5307     \exp_args:Nnnx
5308     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5309     \str_if_empty:NF \sdefinitiontype {
5310       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5311     }
5312     \str_if_empty:NF \sdefinitionname {

```

```

5313     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5314 }
5315 \clist_set:No \l_tmpa_clist \sdefinitiontype
5316 \tl_clear:N \l_tmpa_tl
5317 \clist_map_inline:Nn \l_tmpa_clist {
5318     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5319         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5320     }
5321 }
5322 \tl_if_empty:NTF \l_tmpa_tl {
5323     \__stex_statements_sdefinition_start:
5324 }{
5325     \l_tmpa_tl
5326 }
5327 }
5328 \stex_ref_new_doc_target:n \sdefinitionid
5329 \stex_smsmode_do:
5330 }{
5331     \stex_suppress_html:n {
5332         \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5333     }
5334     \stex_if_smsmode:F {
5335         \clist_set:No \l_tmpa_clist \sdefinitiontype
5336         \tl_clear:N \l_tmpa_tl
5337         \clist_map_inline:Nn \l_tmpa_clist {
5338             \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5339                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5340             }
5341         }
5342         \tl_if_empty:NTF \l_tmpa_tl {
5343             \__stex_statements_sdefinition_end:
5344         }{
5345             \l_tmpa_tl
5346         }
5347         \end{stex_annotate_env}
5348     }
5349 }

```

\stexpatchdefinition

```

5350 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5351     \stex_par:\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5352         ~(\sdefinitiontitle)
5353     }~}
5354 }
5355 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5356
5357 \newcommand\stexpatchdefinition[3]{} {
5358     \str_set:Nx \l_tmpa_str{ #1 }
5359     \str_if_empty:NTF \l_tmpa_str {
5360         \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5361         \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5362     }{
5363         \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5364         \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5365     }
5366 }

```

(End definition for `\stexpatchdefinition`. This function is documented on page 49.)

`\inlinedef` inline:

```

5367 \keys_define:nn {stex / inlinedef }{
5368   type      .str_set_x:N = \sdefinitiontype,
5369   id        .str_set_x:N = \sdefinitionid,
5370   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5371   name      .str_set_x:N = \sdefinitionname
5372 }
5373 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5374   \str_clear:N \sdefinitiontype
5375   \str_clear:N \sdefinitionid
5376   \str_clear:N \sdefinitionname
5377   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5378   \keys_set:nn { stex / inlinedef }{ #1 }
5379 }
5380 \NewDocumentCommand \inlinedef { 0{} m } {
5381   \begingroup
5382   \__stex_statements_inlinedef_args:n{ #1 }
5383   \stex_reactivate_macro:N \definiendum
5384   \stex_reactivate_macro:N \definame
5385   \stex_reactivate_macro:N \Definame
5386   \stex_reactivate_macro:N \premise
5387   \stex_reactivate_macro:N \definiens
5388   \stex_reactivate_macro:N \varbindforall
5389   \stex_ref_new_doc_target:n \sdefinitionid
5390   \stex_if_smsmode:TF{\stex_suppress_html:n {
5391     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5392   }}{
5393     \seq_clear:N \l_tmpb_seq
5394     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5395       \tl_if_empty:nF{ ##1 }{
5396         \stex_get_symbol:n { ##1 }
5397         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5398           \l_stex_get_symbol_uri_str
5399         }
5400       }
5401     }
5402     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5403     \exp_args:Nnx
5404     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5405       \str_if_empty:NF \sdefinitiontype {
5406         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5407       }
5408       #2
5409       \str_if_empty:NF \sdefinitionname {
5410         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5411         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5412       }
5413     }
5414   }

```



```

5462 \clist_map_inline:Nn \l_tmpa_clist {
5463   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5464     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5465   }
5466 }
5467 \tl_if_empty:NTF \l_tmpa_tl {
5468   \__stex_statements_sassertion_start:
5469 }{
5470   \l_tmpa_tl
5471 }
5472 }
5473 \str_if_empty:NTF \sassertionid {
5474   \str_if_empty:NF \sassertionname {
5475     \stex_ref_new_doc_target:n {}
5476   }
5477 } {
5478   \stex_ref_new_doc_target:n \sassertionid
5479 }
5480 \stex_smsmode_do:
5481 ){
5482   \str_if_empty:NF \sassertionname {
5483     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5484     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5485   }
5486   \stex_if_smsmode:F {
5487     \clist_set:No \l_tmpa_clist \sassertiontype
5488     \tl_clear:N \l_tmpa_tl
5489     \clist_map_inline:Nn \l_tmpa_clist {
5490       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5491         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5492       }
5493     }
5494     \tl_if_empty:NTF \l_tmpa_tl {
5495       \__stex_statements_sassertion_end:
5496     }{
5497       \l_tmpa_tl
5498     }
5499     \end{stex_annotate_env}
5500   }
5501 }

```

\stexpatchassertion

```

5502
5503 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5504   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5505     (\sassertiontitle)
5506   }~}
5507 }
5508 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5509
5510 \newcommand\stexpatchassertion[3] [] {
5511   \str_set:Nx \l_tmpa_str{ #1 }
5512   \str_if_empty:NTF \l_tmpa_str {
5513     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

5514     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5515   }{
5516     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5517     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5518   }
5519 }

```

(End definition for `\stexpatchassertion`. This function is documented on page 49.)

`\inlineass` inline:

```

5520 \keys_define:nn {stex / inlineass }{
5521   type      .str_set_x:N = \sassertiontype,
5522   id        .str_set_x:N = \sassertionid,
5523   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5524   name      .str_set_x:N = \sassertionname
5525 }
5526 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5527   \str_clear:N \sassertiontype
5528   \str_clear:N \sassertionid
5529   \str_clear:N \sassertionname
5530   \clist_clear:N \l__stex_statements_sassertion_for_clist
5531   \keys_set:nn { stex / inlineass }{ #1 }
5532 }
5533 \NewDocumentCommand \inlineass { 0{} m } {
5534   \beginngroup
5535   \stex_reactivate_macro:N \premise
5536   \stex_reactivate_macro:N \conclusion
5537   \stex_reactivate_macro:N \varbindforall
5538   \__stex_statements_inlineass_args:n{ #1 }
5539   \str_if_empty:NTF \sassertionid {
5540     \str_if_empty:NF \sassertionname {
5541       \stex_ref_new_doc_target:n {}
5542     }
5543   } {
5544     \stex_ref_new_doc_target:n \sassertionid
5545   }
5546
5547   \stex_if_smsmode:TF{
5548     \str_if_empty:NF \sassertionname {
5549       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5550       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5551     }
5552   }{
5553     \seq_clear:N \l_tmpb_seq
5554     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5555       \tl_if_empty:nF{ ##1 }{
5556         \stex_get_symbol:n { ##1 }
5557         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5558           \l_stex_get_symbol_uri_str
5559         }
5560       }
5561     }
5562     \exp_args:Nnx
5563     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {},,}{

```



```

5564 \str_if_empty:NF \sassassertiontype {
5565   \stex_annotate_invisible:nnn{typestrings}{\sassassertiontype}{ }
5566 }
5567 #2
5568 \str_if_empty:NF \sassassertionname {
5569   \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassassertionname}}
5570   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassassertionname}
5571   \stex_annotate_invisible:nnn{statementname}{\sassassertionname}{ }
5572 }
5573 }
5574 }
5575 \endgroup
5576 \stex_smsmode_do:
5577 }

```

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

31.3 Examples

`sexample`

```

5578
5579 \keys_define:nn {stex / sexample }{
5580   type      .str_set_x:N = \exampletype,
5581   id        .str_set_x:N = \sexampleid,
5582   title     .tl_set:N   = \sexamplename,
5583   name      .str_set_x:N = \sexamplename ,
5584   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5585 }
5586 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5587   \str_clear:N \sexampletype
5588   \str_clear:N \sexampleid
5589   \str_clear:N \sexamplename
5590   \tl_clear:N \sexamplename
5591   \clist_clear:N \l__stex_statements_sexample_for_clist
5592   \keys_set:nn { stex / sexample }{ #1 }
5593 }
5594
5595 \NewDocumentEnvironment{sexample}{0{}}{
5596   \__stex_statements_sexample_args:n{ #1 }
5597   \stex_reactivate_macro:N \premise
5598   \stex_reactivate_macro:N \conclusion
5599   \stex_if_smsmode:F {
5600     \seq_clear:N \l_tmpb_seq
5601     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5602       \tl_if_empty:nF{ ##1 }{
5603         \stex_get_symbol:n { ##1 }
5604         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5605           \l_stex_get_symbol_uri_str
5606         }
5607       }
5608     }
5609     \exp_args:Nnnx
5610     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```

```

5611 \str_if_empty:NF \sexamplotype {
5612   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5613 }
5614 \str_if_empty:NF \sexamplename {
5615   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5616 }
5617 \clist_set:No \l_tmpa_clist \sexamplotype
5618 \tl_clear:N \l_tmpa_tl
5619 \clist_map_inline:Nn \l_tmpa_clist {
5620   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5621     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5622   }
5623 }
5624 \tl_if_empty:NTF \l_tmpa_tl {
5625   \__stex_statements_sexample_start:
5626 }{
5627   \l_tmpa_tl
5628 }
5629 }
5630 \str_if_empty:NF \sexampleid {
5631   \stex_ref_new_doc_target:n \sexampleid
5632 }
5633 \stex_smsmode_do:
5634 }{
5635   \str_if_empty:NF \sexamplename {
5636     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5637   }
5638   \stex_if_smsmode:F {
5639     \clist_set:No \l_tmpa_clist \sexamplotype
5640     \tl_clear:N \l_tmpa_tl
5641     \clist_map_inline:Nn \l_tmpa_clist {
5642       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5643         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5644       }
5645     }
5646     \tl_if_empty:NTF \l_tmpa_tl {
5647       \__stex_statements_sexample_end:
5648     }{
5649       \l_tmpa_tl
5650     }
5651     \end{stex_annotate_env}
5652   }
5653 }

```

\stexpatchexample

```

5654
5655 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5656   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
5657     (\sexampltitle)
5658   }~}
5659 }
5660 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5661
5662 \newcommand\stexpatchexample[3]{} {

```

```

5663 \str_set:Nx \l_tmpa_str{ #1 }
5664 \str_if_empty:NTF \l_tmpa_str {
5665   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5666   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5667 }{
5668   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5669   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5670 }
5671 }

```

(End definition for `\stexpatchexample`. This function is documented on page 49.)

`\inlineex` inline:

```

5672 \keys_define:nn {stex / inlineex }{
5673   type      .str_set_x:N = \sexamplotype,
5674   id        .str_set_x:N = \sexampleid,
5675   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5676   name      .str_set_x:N = \sexamplename
5677 }
5678 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5679   \str_clear:N \sexamplotype
5680   \str_clear:N \sexampleid
5681   \str_clear:N \sexamplename
5682   \clist_clear:N \l__stex_statements_sexample_for_clist
5683   \keys_set:nn { stex / inlineex }{ #1 }
5684 }
5685 \NewDocumentCommand \inlineex { 0{} m } {
5686   \begingroup
5687   \stex_reactivate_macro:N \premise
5688   \stex_reactivate_macro:N \conclusion
5689   \__stex_statements_inlineex_args:n{ #1 }
5690   \str_if_empty:NF \sexampleid {
5691     \stex_ref_new_doc_target:n \sexampleid
5692   }
5693   \stex_if_smsmode:TF{
5694     \str_if_empty:NF \sexamplename {
5695       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
5696   }
5697 }{
5698   \seq_clear:N \l_tmpb_seq
5699   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5700     \tl_if_empty:nF{ ##1 }{
5701       \stex_get_symbol:n { ##1 }
5702       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5703         \l_stex_get_symbol_uri_str
5704       }
5705     }
5706   }
5707   \exp_args:Nnx
5708   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
5709     \str_if_empty:NF \sexamplotype {
5710       \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5711     }
5712   } #2

```

```

5713     \str_if_empty:NF \sexamplename {
5714       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5715       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5716     }
5717   }
5718 }
5719 \endgroup
5720 \stex_smsmode_do:
5721 }

```

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

31.4 Logical Paragraphs

`sparagraph`

```

5722 \keys_define:nn { stex / sparagraph } {
5723   id      .str_set:x:N = \sparagraphid ,
5724   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
5725   type    .str_set:x:N = \sparagraphtype ,
5726   for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
5727   from    .tl_set:N     = \sparagraphfrom ,
5728   to      .tl_set:N     = \sparagraphto ,
5729   start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
5730   name    .str_set:N    = \sparagraphname ,
5731   imports .tl_set:N     = \l__stex_statements_sparagraph_imports_tl
5732 }
5733
5734 \cs_new_protected:Nn \stex_sparagraph_args:n {
5735   \tl_clear:N \l_stex_sparagraph_title_tl
5736   \tl_clear:N \sparagraphfrom
5737   \tl_clear:N \sparagraphto
5738   \tl_clear:N \l_stex_sparagraph_start_tl
5739   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5740   \str_clear:N \sparagraphid
5741   \str_clear:N \sparagraphtype
5742   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5743   \str_clear:N \sparagraphname
5744   \keys_set:nn { stex / sparagraph } { #1 }
5745 }
5746 \newif\if@in@omtext\@in@omtextfalse
5747
5748 \NewDocumentEnvironment {sparagraph} { 0{ } } {
5749   \stex_sparagraph_args:n { #1 }
5750   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5751     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5752   }{
5753     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5754   }
5755   \@in@omtexttrue
5756   \stex_if_smsmode:F {
5757     \seq_clear:N \l_tmpb_seq
5758     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5759       \tl_if_empty:nF{ ##1 }{

```

```

5760     \stex_get_symbol:n { ##1 }
5761     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5762         \l_stex_get_symbol_uri_str
5763     }
5764 }
5765 }
5766 \exp_args:Nnnx
5767 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
5768 \str_if_empty:NF \sparagraphtype {
5769     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5770 }
5771 \str_if_empty:NF \sparagraphfrom {
5772     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5773 }
5774 \str_if_empty:NF \sparagraphto {
5775     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5776 }
5777 \str_if_empty:NF \sparagraphname {
5778     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5779 }
5780 \clist_set:No \l_tmpa_clist \sparagraphtype
5781 \tl_clear:N \l_tmpa_tl
5782 \clist_map_inline:Nn \sparagraphtype {
5783     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5784         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5785     }
5786 }
5787 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5788 \tl_if_empty:NTF \l_tmpa_tl {
5789     \__stex_statements_sparagraph_start:
5790 }{
5791     \l_tmpa_tl
5792 }
5793 }
5794 \clist_set:No \l_tmpa_clist \sparagraphtype
5795 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5796     {
5797         \stex_reactivate_macro:N \definiendum
5798         \stex_reactivate_macro:N \definame
5799         \stex_reactivate_macro:N \Definame
5800         \stex_reactivate_macro:N \premise
5801         \stex_reactivate_macro:N \definiens
5802     }
5803 \str_if_empty:NTF \sparagraphid {
5804     \str_if_empty:NTF \sparagraphname {
5805         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5806             \stex_ref_new_doc_target:n {}
5807         }
5808     } {
5809         \stex_ref_new_doc_target:n {}
5810     }
5811 } {
5812     \stex_ref_new_doc_target:n \sparagraphid
5813 }

```

```

5814 \exp_args:NNx
5815 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5816   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5817     \tl_if_empty:nF{ ##1 }{
5818       \stex_get_symbol:n { ##1 }
5819       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5820     }
5821   }
5822 }
5823 \stex_smsmode_do:
5824 \ignorespacesandpars
5825 }{
5826   \str_if_empty:NF \sparagraphname {
5827     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5828     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5829   }
5830   \stex_if_smsmode:F {
5831     \clist_set:No \l_tmpa_clist \sparagraphtype
5832     \tl_clear:N \l_tmpa_tl
5833     \clist_map_inline:Nn \l_tmpa_clist {
5834       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5835         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5836       }
5837     }
5838     \tl_if_empty:NTF \l_tmpa_tl {
5839       \__stex_statements_sparagraph_end:
5840     }{
5841       \l_tmpa_tl
5842     }
5843     \end{stex_annotate_env}
5844   }
5845 }

```

\stexpatchparagraph

```

5846
5847 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5848   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5849     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5850       \titleemph{\l_stex_sparagraph_title_tl}:~
5851     }
5852   }{
5853     \titleemph{\l_stex_sparagraph_start_tl}~
5854   }
5855 }
5856 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5857
5858 \newcommand\stexpatchparagraph[3] [] {
5859   \str_set:Nx \l_tmpa_str{ #1 }
5860   \str_if_empty:NTF \l_tmpa_str {
5861     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5862     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5863   }{
5864     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5865     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

5866     }
5867 }
5868
5869 \keys_define:nn { stex / inlinepara } {
5870   id      .str_set:N = \sparagraphid ,
5871   type    .str_set:N = \sparagraphtype ,
5872   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5873   from    .tl_set:N   = \sparagraphfrom ,
5874   to      .tl_set:N   = \sparagraphto ,
5875   name    .str_set:N   = \sparagraphname
5876 }
5877 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5878   \tl_clear:N \sparagraphfrom
5879   \tl_clear:N \sparagraphto
5880   \str_clear:N \sparagraphid
5881   \str_clear:N \sparagraphtype
5882   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5883   \str_clear:N \sparagraphname
5884   \keys_set:nn { stex / inlinepara }{ #1 }
5885 }
5886 \NewDocumentCommand \inlinepara { O{} m } {
5887   \begingroup
5888   \__stex_statements_inlinepara_args:n{ #1 }
5889   \clist_set:Nn \l_tmpa_clist \sparagraphtype
5890   \str_if_empty:NTF \sparagraphid {
5891     \str_if_empty:NTF \sparagraphname {
5892       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5893         \stex_ref_new_doc_target:n {}
5894       }
5895     } {
5896       \stex_ref_new_doc_target:n {}
5897     }
5898   } {
5899     \stex_ref_new_doc_target:n \sparagraphid
5900   }
5901   \stex_if_smsmode:TF{
5902     \str_if_empty:NF \sparagraphname {
5903       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5904     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5905   }
5906 }{
5907   \seq_clear:N \l_tmpb_seq
5908   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5909     \tl_if_empty:nF{ ##1 }{
5910       \stex_get_symbol:n { ##1 }
5911       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5912         \l_stex_get_symbol_uri_str
5913       }
5914     }
5915   }
5916   \exp_args:Nnx
5917   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
5918     \str_if_empty:NF \sparagraphtype {
5919       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

5920     }
5921     \str_if_empty:NF \sparagraphfrom {
5922       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5923     }
5924     \str_if_empty:NF \sparagraphto {
5925       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5926     }
5927     \str_if_empty:NF \sparagraphname {
5928       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5929       \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5930       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5931     }
5932     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5933       \clist_map_inline:Nn \l_tmpb_seq {
5934         \stex_ref_new_sym_target:n {##1}
5935       }
5936     }
5937     #2
5938   }
5939 }
5940 \endgroup
5941 \stex_smsmode_do:
5942 }
5943

```

(End definition for `\stexpatchparagraph`. This function is documented on page 49.)

```

5944 </package>

```


Chapter 32

The Implementation

```
5945 <*package>
5946 <@@=stex_sproof>
5947
5948 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
5949
```

32.1 Proofs

We first define some keys for the proof environment.

```
5950 \keys_define:nn { stex / spf } {
5951   id          .str_set_x:N = \spfid,
5952   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5953   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5954   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5955   type        .str_set_x:N = \spftype,
5956   title       .tl_set:N    = \spftitle,
5957   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5958   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5959   term        .tl_set:N    = \l__stex_sproof_spf_term_tl,
5960   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
5961   hide        .bool_set:N  = \l__stex_sproof_spf_hide_bool
5962 }
5963 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
5964   \str_clear:N \spfid
5965   \tl_clear:N \l__stex_sproof_spf_for_tl
5966   \tl_clear:N \l__stex_sproof_spf_from_tl
5967   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5968   \str_clear:N \spftype
5969   \tl_clear:N \spftitle
5970   \tl_clear:N \l__stex_sproof_spf_continues_tl
5971   \tl_clear:N \l__stex_sproof_spf_term_tl
5972   \tl_clear:N \l__stex_sproof_spf_functions_tl
5973   \tl_clear:N \l__stex_sproof_spf_method_tl
5974   \bool_set_false:N \l__stex_sproof_spf_hide_bool
5975   \keys_set:nn { stex / spf }{ #1 }
5976 }
5977 \bool_set_true:N \l__stex_sproof_inc_counter_bool
```

`\c__stex_sproof_flow_str` We define this macro, so that we can test whether the `display` key has the value `flow`

```
5978 \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.

```
5979 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5980 \cs_new_protected:Npn \sproofnumber {
5981   \int_set:Nn \l_tmpa_int {1}
5982   \bool_while_do:nn {
5983     \int_compare_p:nNn {
5984       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5985     } > 0
5986   }{
5987     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5988     \int_incr:N \l_tmpa_int
5989   }
5990 }
5991 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5992   \int_set:Nn \l_tmpa_int {1}
5993   \bool_while_do:nn {
5994     \int_compare_p:nNn {
5995       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5996     } > 0
5997   }{
5998     \int_incr:N \l_tmpa_int
5999   }
6000   \int_compare:nNnF \l_tmpa_int = 1 {
6001     \int_decr:N \l_tmpa_int
6002   }
6003   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6004     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6005   }
6006 }
6007
6008 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6009   \int_set:Nn \l_tmpa_int {1}
6010   \bool_while_do:nn {
6011     \int_compare_p:nNn {
6012       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6013     } > 0
6014   }{
6015     \int_incr:N \l_tmpa_int
6016   }
6017   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6018 }
6019
```

```

6020 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6021   \int_set:Nn \l_tmpa_int {1}
6022   \bool_while_do:nn {
6023     \int_compare_p:nNn {
6024       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6025     } > 0
6026   }{
6027     \int_incr:N \l_tmpa_int
6028   }
6029   \int_decr:N \l_tmpa_int
6030   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6031 }

```

\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

```

6032 \def\sproof@box{
6033   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6034 }
6035 \def\sproofend{
6036   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6037     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6038   }
6039 }

```

(End definition for \sproofend. This function is documented on page 49.)

spf@*kw

```

6040 \def\spf@proofsketch@kw{Proof~Sketch}
6041 \def\spf@proof@kw{Proof}
6042 \def\spf@step@kw{Step}

```

(End definition for spf@*kw. This function is documented on page ??.)

For the other languages, we set up triggers

```

6043 \AddToHook{begindocument}{
6044   \ltx@ifpackageloaded{babel}{
6045     \makeatletter
6046     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6047     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6048       \input{sproof-ngerman.ldf}
6049     }
6050     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6051       \input{sproof-finnish.ldf}
6052     }
6053     \clist_if_in:NnT \l_tmpa_clist {french}{
6054       \input{sproof-french.ldf}
6055     }
6056     \clist_if_in:NnT \l_tmpa_clist {russian}{
6057       \input{sproof-russian.ldf}
6058     }
6059     \makeatother
6060   }{}
6061 }

```

spfsketch

```

6062 \newcommand\spfsketch[2] []{
6063   \begin{group}
6064   \let \premise \stex_proof_premise:
6065   \stex_sproof_spf_args:n{#1}
6066   \stex_if_smsmode:TF {
6067     \str_if_empty:NF \spfid {
6068       \stex_ref_new_doc_target:n \spfid
6069     }
6070   }{
6071     \seq_clear:N \l_tmpa_seq
6072     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6073       \tl_if_empty:nF{ ##1 }{
6074         \stex_get_symbol:n { ##1 }
6075         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6076           \l_stex_get_symbol_uri_str
6077         }
6078       }
6079     }
6080     \exp_args:Nnx
6081     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {},,}{
6082       \str_if_empty:NF \spftype {
6083         \stex_annotate_invisible:nnn{type}{\spftype}{
6084         }
6085       }
6086       \clist_set:Nn \l_tmpa_clist \spftype
6087       \tl_set:Nn \l_tmpa_tl {
6088         \titleemph{
6089           \tl_if_empty:NTF \spftitle {
6090             \spf@proofsketch@kw
6091           }{
6092             \spftitle
6093           }
6094         }
6095       }
6096       \clist_map_inline:Nn \l_tmpa_clist {
6097         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6098           \tl_clear:N \l_tmpa_tl
6099         }
6100       }
6101       \str_if_empty:NF \spfid {
6102         \stex_ref_new_doc_target:n \spfid
6103       }
6104       \l_tmpa_tl #2 \sproofend
6105     }
6106   }
6107   \endgroup
6108 }
6109

```

(End definition for *spfsketch*. This function is documented on page 48.)

```

\stex_sproof_maybe_comment:
\stex_sproof_maybe_comment_end:
\stex_sproof_start_comment:
6110 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6111
6112 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6113   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6114     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6115   }
6116 }
6117 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6118   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6119 }
6120 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6121   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6122 }
6123

```

(End definition for __stex_sproof_maybe_comment:, __stex_sproof_maybe_comment_end:, and __stex_sproof_start_comment:.)

\stexcommentfont

```

6124 \cs_new_protected:Npn \stexcommentfont {
6125   \small\itshape
6126 }

```

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

sproof 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.

```

6127 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6128   \seq_clear:N \l_tmpa_seq
6129   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6130     \tl_if_empty:NF{ ##1 }{
6131       \stex_get_symbol:n { ##1 }
6132       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6133         \l_stex_get_symbol_uri_str
6134       }
6135     }
6136   }
6137   \exp_args:Nnnx
6138   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6139   \str_if_empty:NF \spftype {
6140     \stex_annotate_invisible:nnn{type}{\spftype}{}
6141   }
6142   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6143   \str_if_empty:NF \spfid {
6144     \stex_ref_new_doc_target:n \spfid
6145   }
6146   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6147   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6148     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6149   }
6150   \begin{list}{}{
6151     \setlength\topsep{0pt}
6152     \setlength\parsep{0pt}
6153     \setlength\rightmargin{0pt}

```

```

6154 } \__stex_sproof_maybe_comment:
6155 }
6156 \cs_new_protected:Nn \__stex_sproof_end_env:n {
6157   \stex_if_smsmode:F{
6158     \__stex_sproof_maybe_comment_end:
6159     \end{list}
6160     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6161       \stex_html_backend:F{\egroup}
6162     }
6163     \clist_set:No \l_tmpa_clist \spftype
6164     #1
6165     \end{stex_annotate_env}
6166     \end{stex_annotate_env}
6167   }
6168 }
6169 }
6170 \NewDocumentEnvironment{sproof}{s O{} m}{
6171   \intarray_gzero:N \l__stex_sproof_counter_intarray
6172   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6173   \stex_reactivate_macro:N \yield
6174   \stex_reactivate_macro:N \eqstep
6175   \stex_reactivate_macro:N \assumption
6176   \stex_reactivate_macro:N \conclude
6177   \stex_reactivate_macro:N \spfstep
6178   \__stex_sproof_spf_args:n{#2}
6179   \stex_if_smsmode:TF {
6180     \str_if_empty:NF \spfid {
6181       \stex_ref_new_doc_target:n \spfid
6182     }
6183   }{
6184     \__stex_sproof_start_env:nnn{sproof}{#3}{
6185       \clist_set:No \l_tmpa_clist \spftype
6186       \tl_clear:N \l_tmpa_tl
6187       \clist_map_inline:Nn \l_tmpa_clist {
6188         \tl_if_exist:cT {\__stex_sproof_sproof_##1_start:}{
6189           \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_start:}}
6190         }
6191         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6192           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6193         }
6194       }
6195       \tl_if_empty:NTF \l_tmpa_tl {
6196         \__stex_sproof_sproof_start:
6197       }{
6198         \l_tmpa_tl
6199       }
6200     }
6201   }
6202   \stex_smsmode_do:
6203 }{\__stex_sproof_end_env:n{
6204   \tl_clear:N \l_tmpa_tl
6205   \clist_map_inline:Nn \l_tmpa_clist {
6206     \tl_if_exist:cT {\__stex_sproof_sproof_##1_end:}{
6207       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_end:}}

```

```

6208     }
6209   }
6210   \tl_if_empty:NTF \l_tmpa_tl {
6211     \__stex_sproof_sproof_end:
6212   }{
6213     \l_tmpa_tl
6214   }
6215 }}
6216 \NewDocumentEnvironment{subproof}{s O{} m}{
6217   \__stex_sproof_spf_args:n{#2}
6218   \stex_if_smsmode:TF {
6219     \str_if_empty:NF \spfid {
6220       \stex_ref_new_doc_target:n \spfid
6221     }
6222   }{
6223     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6224   }
6225   \__stex_sproof_add_counter:
6226   \stex_smsmode_do:
6227 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6228   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6229     \__stex_sproof_inc_counter:
6230   }
6231   \aftergroup\__stex_sproof_maybe_comment:
6232 }
6233 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6234
6235 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6236   \par\noindent\titleemph{
6237     \tl_if_empty:NTF \spftype {
6238       \spf@proof@kw
6239     }{
6240       \spftype
6241     }
6242   }:
6243 }
6244 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6245
6246 \newcommand\stexpatchproof[3] [] {
6247   \str_set:Nx \l_tmpa_str{ #1 }
6248   \str_if_empty:NTF \l_tmpa_str {
6249     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6250     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6251   }{
6252     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6253     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6254   }
6255 }

\pstep
\conclude
\assumption
\have
\eqstep

```

```

6256
6257 \keys_define:nn { stex / spfsteps } {
6258   id          .str_set_x:N = \spfstepid,
6259   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,

```

```

6260 type .str_set_x:N = \spftype,
6261 title .tl_set:N = \spftitle,
6262 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6263 term .tl_set:N = \l__stex_sproof_spf_term_tl
6264 }
6265 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6266 \str_clear:N \spfstepid
6267 \clist_clear:N \l__stex_sproof_spf_for_clist
6268 \str_clear:N \spftype
6269 \tl_clear:N \l__stex_sproof_spf_method_tl
6270 \tl_clear:N \l__stex_sproof_spf_term_tl
6271 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6272 \keys_set:nn { stex / spfsteps }{ #1 }
6273 }
6274
6275 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6276 \NewDocumentCommand #1 {s O{} +m} {
6277 \__stex_sproof_maybe_comment_end:
6278
6279 \__stex_sproof_spfstep_args:n{##2}
6280 \stex_annotate:nnn{spfstep}{#2}{
6281 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6282 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6283 }
6284 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6285 #4
6286 }{
6287 \item[\IfBooleanTF ##1 {}{#3}]
6288 }
6289 \ignorespacesandpars ##3
6290 }
6291 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6292 \__stex_sproof_maybe_comment:
6293 }
6294 \stex_deactivate_macro:Nn #1 {sproof~environments}
6295 }
6296
6297 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6298 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6299 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6300
6301 \NewDocumentCommand \eqstep {s m}{
6302 \__stex_sproof_maybe_comment_end:
6303 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6304 $=$
6305 }{
6306 \item[$=$]
6307 }
6308 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6309 \__stex_sproof_maybe_comment:
6310 }
6311 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6312
6313 \NewDocumentCommand \yield {+m}{

```



```

6314 \stex_annotate:nnn{spfyield}{\}{ #1 }
6315 }
6316 \stex_deactivate_macro:Nn \yield {sproof~environments}
6317
6318 \NewDocumentEnvironment{spfblock}{\}{\{
6319 \item[]
6320 \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6321 \}{
6322 \aftergroup\__stex_sproof_maybe_comment:
6323 }
6324 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6325

```

(End definition for `\pstep` and others. These functions are documented on page ??.)

`\spfidea`

```

6326 \NewDocumentCommand\spfidea{0{\} +m}{
6327 \__stex_sproof_spf_args:n{#1}
6328 \titleemph{
6329 \tl_if_empty:NTF \spftype {Proof~Idea}{
6330 \spftype
6331 }~#2
6332 }~#2
6333 \sproofend
6334 }

```

(End definition for `\spfidea`. This function is documented on page 48.)

```

6335 \newcommand\spfjust[1]{
6336 #1
6337 }
6338 \</package>

```

Some auxiliary code, and clean up to be executed at the end of the package.

Chapter 33

STEX -Others Implementation

```
6339 <*package>
6340
6341 %%%%%%%%%% others.dtx %%%%%%%%%%
6342
6343 <@@=stex_others>
        Warnings and error messages
6344 % None

\MSC Math subject classifier

6345 \NewDocumentCommand \MSC {m} {
6346 % TODO
6347 }

(End definition for \MSC. This function is documented on page ??.)
        Patching tikzinput, if loaded

6348 \@ifpackageloaded{tikzinput}{
6349 \RequirePackage{stex-tikzinput}
6350 }{}
6351
6352 \bool_if:NT \c_stex_persist_mode_bool {
6353 \let__stex_notation_restore_notation_old:nnnnn
6354 \__stex_notation_restore_notation:nnnnn
6355 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6356 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6357 \ExplSyntaxOn
6358 }
6359 \def__stex_notation_restore_notation:nnnnn{
6360 \ExplSyntaxOff
6361 \catcode'\sim10
6362 \__stex_notation_restore_notation_new:nnnnn
6363 }
6364 \input{\jobname.sms}
6365 \let__stex_notation_restore_notation:nnnnn
6366 \__stex_notation_restore_notation_old:nnnnn
6367 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```

6368     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6369     \l_tmpa_str
6370     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6371     \c_stex_mathhub_main_manifest_prop
6372     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6373   }
6374 }
6375 </package>

```

Chapter 34

STEX -Metatheory Implementation

```
6376 <*package>
6377 <@@=stex_modules>
6378
6379 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6380
6381 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6382 \begingroup
6383 \stex_module_setup:nn{
6384   ns=\c_stex_metatheory_ns_str,
6385   meta=NONE
6386 }{Metatheory}
6387 \stex_reactivate_macro:N \symdecl
6388 \stex_reactivate_macro:N \notation
6389 \stex_reactivate_macro:N \symdef
6390 \ExplSyntaxOff
6391 \csname stex_suppress_html:n\endcsname{
6392   % is-a (a:A, a \in A, a is an A, etc.)
6393   \symdecl{isa}[args=ai]
6394   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6395   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6396   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6397
6398   % bind (\forall, \Pi, \lambda etc.)
6399   \symdecl{bind}[args=Bi,assoc=pre]
6400   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6401   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6402   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6403
6404   % implicit bind
6405   \symdecl{implicitbind}[args=Bi,assoc=pre]
6406   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6407   \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6408   \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6409
6410   % dummy variable
```

```

6411 \symdecl{dummyvar}
6412 \notation{dummyvar}[underscore]{\comp\_}
6413 \notation{dummyvar}[dot]{\comp\cdot}
6414 \notation{dummyvar}[dash]{\comp{\rm --}}
6415
6416 %fromto (function space, Hom-set, implication etc.)
6417 \symdecl{fromto}[args=ai]
6418 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6419 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6420
6421 % mapto (lambda etc.)
6422 \symdecl{mapto}[args=Bi]
6423 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6424 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6425 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6426
6427 % function/operator application
6428 \symdecl{apply}[args=ia]
6429 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6430 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6431
6432 % collection of propositions/booleans/truth values
6433 \symdecl{prop}[name=proposition]
6434 \notation{prop}[prop]{\comp{\rm prop}}
6435 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6436
6437 \symdecl{judgmentholds}[args=1]
6438 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6439
6440 % sequences
6441 \symdecl{seqtype}[args=1]
6442 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6443
6444 \symdecl{seqexpr}[args=a]
6445 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6446
6447 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
6448 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6449 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6450 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6451 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp
6452 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6453 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6454 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6455 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6456
6457 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6458 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
6459
6460 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6461 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6462 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
6463
6464 % nat literals

```

```

6465 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6466
6467 % letin (''let'', local definitions, variable substitution)
6468 \symdecl{letin}[args=bii]
6469 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
6470 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6471 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6472
6473 % structures
6474 \symdecl*{module-type}[args=1]
6475 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6476 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6477 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6478
6479 % objects
6480 \symdecl{object}
6481 \notation{object}{\comp{\mathtt{OBJECT}}}
6482
6483 }
6484
6485 % The following are abbreviations in the sTeX corpus that are left over from earlier
6486 % developments. They will eventually be phased out.
6487
6488 \ExplSyntaxOn
6489 \stex_add_to_current_module:n{
6490   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6491   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6492   \def\livar{\csname sequence-index\endcsname[li]}
6493   \def\uivar{\csname sequence-index\endcsname[ui]}
6494   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6495   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6496 }
6497 \__stex_modules_end_module:
6498 \endgroup
6499 \</package>

```

Chapter 35

Tikzinput Implementation

```
6500 <@@=tikzinput>
6501 <*package>
6502
6503 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6504
6505 \ProvidesExplPackage{tikzinput}{2022/05/24}{3.1.0}{tikzinput package}
6506 \RequirePackage{l3keys2e}
6507
6508 \keys_define:nn { tikzinput } {
6509   image .bool_set:N = \c_tikzinput_image_bool,
6510   image .default:n = false ,
6511   unknown .code:n = {}
6512 }
6513
6514 \ProcessKeysOptions { tikzinput }
6515
6516 \bool_if:NTF \c_tikzinput_image_bool {
6517   \RequirePackage{graphicx}
6518
6519   \providecommand\usetikzlibrary[]{}
6520   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6521 }{
6522   \RequirePackage{tikz}
6523   \RequirePackage{standalone}
6524
6525   \newcommand \tikzinput [2] [] {
6526     \setkeys{Gin}{#1}
6527     \ifx \Gin@ewidth \Gin@exclamation
6528       \ifx \Gin@eheight \Gin@exclamation
6529         \input { #2 }
6530       \else
6531         \resizebox{!}{ \Gin@eheight }{
6532           \input { #2 }
6533         }
6534       \fi
6535     \else
6536       \ifx \Gin@eheight \Gin@exclamation
6537         \resizebox{ \Gin@ewidth }{!}{
```

```

6538         \input { #2 }
6539     }
6540     \else
6541         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6542             \input { #2 }
6543         }
6544     \fi
6545 \fi
6546 }
6547 }
6548
6549 \newcommand \ctikzinput [2] [] {
6550     \begin{center}
6551         \tikzinput [#1] {#2}
6552     \end{center}
6553 }
6554
6555 \@ifpackageloaded{stex}{
6556     \RequirePackage{stex-tikzinput}
6557 }{}
6558
6559 </package>
6560 <*stex>
6561 \ProvidesExplPackage{stex-tikzinput}{2022/05/24}{3.1.0}{stex-tikzinput}
6562 \RequirePackage{stex}
6563 \RequirePackage{tikzinput}
6564
6565 \newcommand\mhtikzinput[2] []{%
6566     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
6567     \stex_in_repository:nn\Gin@mhrepos{
6568         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
6569     }
6570 }
6571 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6572
6573 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6574     \pgfkeys@spdef\pgf@temp{#1}
6575     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6576     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6577     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6578     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6579     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6580     \catcode'\@=11
6581     \catcode'\|=12
6582     \catcode'\$=3
6583     \pgfutil@InputIfFileExists{#2}{-}{-}
6584     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6585     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6586     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6587 }
6588
6589
6590 \newcommand\libusetikzlibrary[1]{

```



```

6591 \prop_if_exist:NF \l_stex_current_repository_prop {
6592   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6593 }
6594 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6595   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6596 }
6597 \seq_clear:N \l__tikzinput_libinput_files_seq
6598 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6599 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6600
6601 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6602   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6603   \IfFileExists{ \l_tmpa_str }{
6604     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6605   }{}
6606   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6607   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6608 }
6609
6610 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6611 \IfFileExists{ \l_tmpa_str }{
6612   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6613 }{}
6614
6615 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6616   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6617 }{
6618   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6619     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6620       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6621     }
6622   }{
6623     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6624   }
6625 }
6626 }
6627 </stex>

```

Chapter 36

document-structure.sty Implementation

```
6628 <*package>
6629 <@@=document_structure>
6630 \ProvidesExplPackage{document-structure}{2022/05/24}{3.1.0}{Modular Document Structure}
6631 \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).

```
6632
6633 \keys_define:nn{ document-structure }{
6634   class      .str_set_x:N = \c_document_structure_class_str,
6635   topsect    .str_set_x:N = \c_document_structure_topsect_str,
6636   unknown    .code:n      = {
6637     \PassOptionsToClass{\CurrentOption}{stex}
6638     \PassOptionsToClass{\CurrentOption}{tikzinput}
6639   }
6640   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6641 }
6642 \ProcessKeysOptions{ document-structure }
6643 \str_if_empty:NT \c_document_structure_class_str {
6644   \str_set:Nn \c_document_structure_class_str {article}
6645 }
6646 \str_if_empty:NT \c_document_structure_topsect_str {
6647   \str_set:Nn \c_document_structure_topsect_str {section}
6648 }
```

Then we need to set up the packages by requiring the `sref` package to be loaded, and set up triggers for other languages

```
6649 \RequirePackage{xspace}
6650 \RequirePackage{comment}
6651 \RequirePackage{stex}
6652 \AddToHook{begindocument}{}
```

```

6653 \ltx@ifpackageloaded{babel}{
6654   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6655   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
6656     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6657   }
6658 }{}
6659 }

```

`\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.

```

6660 \int_new:N \l_document_structure_section_level_int
6661 \str_case:NnF \c_document_structure_topsect_str {
6662   {part}}{
6663     \int_set:Nn \l_document_structure_section_level_int {0}
6664   }
6665   {chapter}{
6666     \int_set:Nn \l_document_structure_section_level_int {1}
6667   }
6668 }{
6669   \str_case:NnF \c_document_structure_class_str {
6670     {book}{
6671       \int_set:Nn \l_document_structure_section_level_int {0}
6672     }
6673     {report}{
6674       \int_set:Nn \l_document_structure_section_level_int {0}
6675     }
6676   }{
6677     \int_set:Nn \l_document_structure_section_level_int {2}
6678   }
6679 }

```

36.2 Document Structure

The structure of the document is given by the `sfragment` environment. The hierarchy is adjusted automatically according to the \LaTeX 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.⁹

EdN:9

```

6680 \def\current@section@level{document}%
6681 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6682 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

(End definition for `\currentsectionlevel`. This function is documented on page 54.)

`\skipfragment`

```

6683 \cs_new_protected:Npn \skipfragment {

```

⁹EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6684 \ifcase\l_document_structure_section_level_int
6685 \or\stepcounter{part}
6686 \or\stepcounter{chapter}
6687 \or\stepcounter{section}
6688 \or\stepcounter{subsection}
6689 \or\stepcounter{subsubsection}
6690 \or\stepcounter{paragraph}
6691 \or\stepcounter{subparagraph}
6692 \fi
6693 }

```

(End definition for `\skipfragment`. This function is documented on page 53.)

blindfragment

```

6694 \newcommand\at@begin@blindsfragment[1]{
6695 \newenvironment{blindfragment}
6696 {
6697 \int_incr:N\l_document_structure_section_level_int
6698 \at@begin@blindsfragment\l_document_structure_section_level_int
6699 }{}

```

\sfragment@nonum convenience macro: `\sfragment@nonum{<level>}{<title>}` makes an unnumbered sectioning with title `<title>` at level `<level>`.

```

6700 \newcommand\sfragment@nonum[2]{
6701 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6702 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6703 }

```

(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.

```

6704 \newcommand\sfragment@num[2]{
6705 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6706 \@nameuse{#1}{#2}
6707 }{
6708 \cs_if_exist:NTF\rdfmata@sectioning{
6709 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6710 }{
6711 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6712 }
6713 }
6714 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6715 }

```

(End definition for `\sfragment@num`. This function is documented on page ??.)

sfragment

```

6716 \keys_define:nn { document-structure / sfragment }{
6717 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6718 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6719 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6720 contributors   .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6721 srccite        .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6722 type           .tl_set:N     = \l__document_structure_sfragment_type_tl,
6723 short          .tl_set:N     = \l__document_structure_sfragment_short_tl,
6724 intro          .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6725 imports        .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6726 loadmodules    .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6727 }
6728 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6729   \str_clear:N \l__document_structure_sfragment_id_str
6730   \str_clear:N \l__document_structure_sfragment_date_str
6731   \clist_clear:N \l__document_structure_sfragment_creators_clist
6732   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6733   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6734   \tl_clear:N \l__document_structure_sfragment_type_tl
6735   \tl_clear:N \l__document_structure_sfragment_short_tl
6736   \tl_clear:N \l__document_structure_sfragment_imports_tl
6737   \tl_clear:N \l__document_structure_sfragment_intro_tl
6738   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6739   \keys_set:nn { document-structure / sfragment } { #1 }
6740 }

```

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.

```

6741 \newif\if@mainmatter\@mainmattertrue
6742 \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.

```

6743 \keys_define:nn { document-structure / sectioning }{
6744   name      .str_set_x:N = \l__document_structure_sect_name_str   ,
6745   ref       .str_set_x:N = \l__document_structure_sect_ref_str    ,
6746   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6747   clear     .default:n   = {true}                                ,
6748   num       .bool_set:N  = \l__document_structure_sect_num_bool   ,
6749   num       .default:n   = {true}
6750 }
6751 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6752   \str_clear:N \l__document_structure_sect_name_str
6753   \str_clear:N \l__document_structure_sect_ref_str
6754   \bool_set_false:N \l__document_structure_sect_clear_bool
6755   \bool_set_false:N \l__document_structure_sect_num_bool
6756   \keys_set:nn { document-structure / sectioning } { #1 }
6757 }
6758 \newcommand\omdoc@sectioning[3][]{
6759   \l__document_structure_sect_args:n {#1 }
6760   \let\omdoc@sect@name\l__document_structure_sect_name_str
6761   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6762   \if@mainmatter% numbering not overridden by frontmatter, etc.
6763     \bool_if:NTF \l__document_structure_sect_num_bool {
6764       \sfragment@num{#2}{#3}
6765     }{

```

```

6766     \sfragment@nonum{#2}{#3}
6767   }
6768   \def\current@section@level{\omdoc@sect@name}
6769   \else
6770     \sfragment@nonum{#2}{#3}
6771   \fi
6772 }% 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.

```

6773 \newcommand\sfragment@redefine@addtocontents[1]{%
6774 %\edef\__document_structureimport{#1}%
6775 %\@for\@I:=\__document_structureimport\do{%
6776 %\edef\@path{\csname module@\@I @path\endcsname}%
6777 %\@ifundefined{tf@toc}\relax%
6778 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6779 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6780 %\def\addcontentsline##1##2##3{%
6781 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
6782 %\else% hyperref.sty not loaded
6783 %\def\addcontentsline##1##2##3{%
6784 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
6785 %\fi
6786 }% 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.

```

6787 \newenvironment{sfragment}[2][ ]% keys, title
6788 {
6789   \__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.

```

6790   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6791
6792   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6793     \sfragment@redefine@addtocontents{
6794       %\@ifundefined{module@id}\used@modules%
6795       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6796     }
6797   }

```

now we only need to construct the right sectioning depending on the value of `\section@level`.

```

6798
6799   \stex_document_title:n { #2 }
6800
6801   \int_incr:N\l__document_structure_section_level_int
6802   \ifcase\l__document_structure_section_level_int
6803     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6804     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6805     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6806     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

6807 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6808 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
6809 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#1}
6810 \fi
6811 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
6812 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6813   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6814 }
6815 }% for customization
6816 {}

```

and finally, we localize the sections

```

6817 \newcommand\omdoc@part@kw{Part}
6818 \newcommand\omdoc@chapter@kw{Chapter}
6819 \newcommand\omdoc@section@kw{Section}
6820 \newcommand\omdoc@subsection@kw{Subsection}
6821 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6822 \newcommand\omdoc@paragraph@kw{paragraph}
6823 \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`

```

6824 \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).

```

6825 \cs_if_exist:NTF\frontmatter{
6826   \let\__document_structure_orig_frontmatter\frontmatter
6827   \let\frontmatter\relax
6828 }{
6829   \tl_set:Nn\__document_structure_orig_frontmatter{
6830     \clearpage
6831     \@mainmatterfalse
6832     \pagenumbering{roman}
6833   }
6834 }
6835 \cs_if_exist:NTF\backmatter{
6836   \let\__document_structure_orig_backmatter\backmatter
6837   \let\backmatter\relax
6838 }{
6839   \tl_set:Nn\__document_structure_orig_backmatter{
6840     \clearpage
6841     \@mainmatterfalse
6842     \pagenumbering{roman}
6843   }

```

6844 }

Using these, we can now define the `frontmatter` and `backmatter` environments

frontmatter we use the `\orig@frontmatter` macro defined above and `\mainmatter` if it exists, otherwise we define it.

```
6845 \newenvironment{frontmatter}{
6846   \_document_structure_orig_frontmatter
6847 }{
6848   \cs_if_exist:NTF\mainmatter{
6849     \mainmatter
6850   }{
6851     \clearpage
6852     \@mainmattertrue
6853     \pagenumbering{arabic}
6854   }
6855 }
```

backmatter As `backmatter` is at the end of the document, we do nothing for `\endbackmatter`.

```
6856 \newenvironment{backmatter}{
6857   \_document_structure_orig_backmatter
6858 }{
6859   \cs_if_exist:NTF\mainmatter{
6860     \mainmatter
6861   }{
6862     \clearpage
6863     \@mainmattertrue
6864     \pagenumbering{arabic}
6865   }
6866 }
```

finally, we make sure that page numbering is arabic and we have main matter as the default

```
6867 \@mainmattertrue\pagenumbering{arabic}
```

\prematurestop We initialize `\afterprematurestop`, and provide `\prematurestop@endsfragment` which looks up `\sfragment@level` and recursively ends enough `{sfragment}s`.

```
6868 \def \c__document_structure_document_str{document}
6869 \newcommand\afterprematurestop{}
6870 \def\prematurestop@endsfragment{
6871   \unless\ifx\@currenvir\c__document_structure_document_str
6872     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6873       \expandafter\prematurestop@endsfragment
6874     }
6875   }
6876 \providecommand\prematurestop{
6877   \message{Stopping~sTeX~processing~prematurely}
6878   \prematurestop@endsfragment
6879   \afterprematurestop
6880   \end{document}
6881 }
```

(End definition for `\prematurestop`. This function is documented on page 54.)

36.4 Global Variables

\setSGvar set a global variable

```
6882 \RequirePackage{etoolbox}
6883 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

(End definition for \setSGvar. This function is documented on page 54.)

\useSGvar use a global variable

```
6884 \newrobustcmd\useSGvar[1]{%
6885   \@ifundefined{sTeX@Gvar@#1}
6886   {\PackageError{document-structure}
6887     {The sTeX Global variable #1 is undefined}
6888     {set it with \protect\setSGvar}}
6889   \@nameuse{sTeX@Gvar@#1}}
```

(End definition for \useSGvar. This function is documented on page 54.)

\ifSGvar execute something conditionally based on the state of the global variable.

```
6890 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6891   \@ifundefined{sTeX@Gvar@#1}
6892   {\PackageError{document-structure}
6893     {The sTeX Global variable #1 is undefined}
6894     {set it with \protect\setSGvar}}
6895   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

(End definition for \ifSGvar. This function is documented on page 54.)

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.

```
6896 \*cls)
6897 \@@=notesslides)
6898 \ProvidesExplClass{notesslides}{2022/05/24}{3.1.0}{notesslides Class}
6899 \RequirePackage{13keys2e}
6900
6901 \keys_define:nn{notesslides / cls}{
6902   class .str_set_x:N = \c__notesslides_class_str,
6903   notes .bool_set:N = \c__notesslides_notes_bool ,
6904   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6905   docopt .str_set_x:N = \c__notesslides_docopt_str,
6906   unknown .code:n = {
6907     \PassOptionsToPackage{\CurrentOption}{document-structure}
6908     \PassOptionsToClass{\CurrentOption}{beamer}
6909     \PassOptionsToPackage{\CurrentOption}{notesslides}
6910     \PassOptionsToPackage{\CurrentOption}{stex}
6911   }
6912 }
6913 \ProcessKeysOptions{ notesslides / cls }
6914
6915 \str_if_empty:NF \c__notesslides_class_str {
6916   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6917 }
6918
6919 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6920   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6921 }
6922 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6923   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6924 }
6925
6926 \RequirePackage{stex}
```

```

6927 \stex_html_backend:T {
6928   \bool_set_true:N\c__notesslides_notes_bool
6929 }
6930
6931 \bool_if:NTF \c__notesslides_notes_bool {
6932   \PassOptionsToPackage{notes=true}{notesslides}
6933   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
6934 }{
6935   \PassOptionsToPackage{notes=false}{notesslides}
6936   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
6937 }
6938 </cls>

```

now we do the same for the notesslides package.

```

6939 <*package>
6940 \ProvidesExplPackage{notesslides}{2022/05/24}{3.1.0}{notesslides Package}
6941 \RequirePackage{l3keys2e}
6942
6943 \keys_define:nn{notesslides / pkg}{
6944   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6945   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6946   notes            .bool_set:N = \c__notesslides_notes_bool ,
6947   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6948   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
6949   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
6950   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
6951   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
6952   unknown          .code:n      = {
6953     \PassOptionsToClass{\CurrentOption}{stex}
6954     \PassOptionsToClass{\CurrentOption}{tikzinput}
6955   }
6956 }
6957 \ProcessKeysOptions{ notesslides / pkg }
6958
6959 \RequirePackage{stex}
6960 \stex_html_backend:T {
6961   \bool_set_true:N\c__notesslides_notes_bool
6962 }
6963
6964 \newif\ifnotes
6965 \bool_if:NTF \c__notesslides_notes_bool {
6966   \notesttrue
6967 }{
6968   \notesfalse
6969 }
6970

```

we give ourselves a macro \@@topsect that needs only be evaluated once, so that the \ifdefstring conditionals work below.

```

6971 \str_if_empty:NTF \c__notesslides_topsect_str {
6972   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6973 }{
6974   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6975 }
6976 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
6977 </package>
```

Depending on the options, we either load the `article`-based document-structure or the `beamer` class (and set some counters).

```
6978 <*cls>
6979 \bool_if:NTF \c__notesslides_notes_bool {
6980   \str_if_empty:NT \c__notesslides_class_str {
6981     \str_set:Nn \c__notesslides_class_str {article}
6982   }
6983   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
6984     {\c__notesslides_class_str}
6985 }{
6986   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6987   \newcounter{Item}
6988   \newcounter{paragraph}
6989   \newcounter{subparagraph}
6990   \newcounter{Hfootnote}
6991 }
6992 \RequirePackage{document-structure}
```

now it only remains to load the `notesslides` package that does all the rest.

```
6993 \RequirePackage{notesslides}
6994 </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.

```
6995 <*package>
6996 \bool_if:NT \c__notesslides_notes_bool {
6997   \RequirePackage{a4wide}
6998   \RequirePackage{marginnote}
6999   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7000   \RequirePackage{mdframed}
7001   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7002   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7003 }
7004 \RequirePackage{stex-tikzinput}
7005 \RequirePackage{comment}
7006 \RequirePackage{url}
7007 \RequirePackage{graphicx}
7008 \RequirePackage{pgf}
7009 \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.

```
7010 \bool_if:NT \c__notesslides_notes_bool {
7011   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7012 }
```

```

7013 \NewDocumentCommand \libusetheme {0{} m} {
7014   \libusepackage[#1]{beamertheme#2}
7015 }
7016

```

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```

7017 \newcounter{slide}
7018 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7019 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

note 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.

```

7020 \bool_if:NTF \c__notesslides_notes_bool {
7021   \renewenvironment{note}{\ignorespaces}{}
7022 }{
7023   \excludecomment{note}
7024 }

```

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.

```

7025 \bool_if:NT \c__notesslides_notes_bool {
7026   \newlength{\slideframewidth}
7027   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

7028 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7029   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7030     \bool_set_true:N #1
7031   }{
7032     \bool_set_false:N #1
7033   }
7034 }
7035 \keys_define:nn{notesslides / frame}{
7036   label .str_set_x:N = \l__notesslides_frame_label_str,
7037   allowframebreaks .code:n = {
7038     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7039   },
7040   allowdisplaybreaks .code:n = {
7041     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7042   },
7043   fragile .code:n = {
7044     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7045   },
7046   shrink .code:n = {
7047     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7048   },
7049   squeeze .code:n = {
7050     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7051   },
7052   t .code:n = {

```

```

7053     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7054   },
7055   unknown    .code:n      = {}
7056 }
7057 \cs_new_protected:Nn \__notesslides_frame_args:n {
7058   \str_clear:N \l__notesslides_frame_label_str
7059   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7060   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7061   \bool_set_true:N \l__notesslides_frame_fragile_bool
7062   \bool_set_true:N \l__notesslides_frame_shrink_bool
7063   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7064   \bool_set_true:N \l__notesslides_frame_t_bool
7065   \keys_set:nn { notesslides / frame }{ #1 }
7066 }

```

We define the environment, read them, and construct the slide number and label.

```

7067 \renewenvironment{frame}[1][]{
7068   \__notesslides_frame_args:n{#1}
7069   \sffamily
7070   \stepcounter{slide}
7071   \def\@currentlabel{\theslide}
7072   \str_if_empty:NF \l__notesslides_frame_label_str {
7073     \label{\l__notesslides_frame_label_str}
7074   }

```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```

7075   \def\itemize@level{outer}
7076   \def\itemize@outer{outer}
7077   \def\itemize@inner{inner}
7078   \renewcommand\newpage{\addtocounter{framenum}{1}}
7079   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7080   \renewenvironment{itemize}{
7081     \ifx\itemize@level\itemize@outer
7082       \def\itemize@label{$\rhd$}
7083     \fi
7084     \ifx\itemize@level\itemize@inner
7085       \def\itemize@label{$\scriptstyle\rhd$}
7086     \fi
7087     \begin{list}
7088       {\itemize@label}
7089       {\setlength{\labelsep}{.3em}
7090        \setlength{\labelwidth}{.5em}
7091        \setlength{\leftmargin}{1.5em}
7092       }
7093     \edef\itemize@level{\itemize@inner}
7094   }{
7095     \end{list}
7096   }

```

We create the box with the `mdframed` environment from the `equinymous` package.

```

7097   \stex_html_backend:TF {
7098     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7099     \mdf@patchamsthm
7100   }{
7101     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7102     }
7103   }{
7104     \stex_html_backend:TF {
7105       \miko@slidelabel\egroup\end{stex_annotate_env}
7106     }\medskip\miko@slidelabel\end{mdframed}}
7107   }

```

Now, we need to redefine the frametitle (we are still in course notes mode).

\frametitle

```

7108   \renewcommand{\frametitle}[1]{
7109     \stex_document_title:n { #1 }
7110     {\Large\bf\sf\color{blue}{#1}}\medskip
7111   }
7112 }

```

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

EdN:10

\pause 10

```

7113 \bool_if:NT \c__notesslides_notes_bool {
7114   \newcommand\pause{}
7115 }

```

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

nparagraph

```

7116 \bool_if:NTF \c__notesslides_notes_bool {
7117   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7118 }{
7119   \excludecomment{nparagraph}
7120 }

```

nfragment

```

7121 \bool_if:NTF \c__notesslides_notes_bool {
7122   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
7123 }{
7124   \excludecomment{nfragment}
7125 }

```

ndefinition

```

7126 \bool_if:NTF \c__notesslides_notes_bool {
7127   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7128 }{
7129   \excludecomment{ndefinition}
7130 }

```

nassertion

```

7131 \bool_if:NTF \c__notesslides_notes_bool {
7132   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7133 }{
7134   \excludecomment{nassertion}
7135 }

```

¹⁰EDNOTE: MK: fake it in notes mode for now

nsproof

```
7136 \bool_if:NTF \c__notesslides_notes_bool {
7137   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7138 }{
7139   \excludecomment{nproof}
7140 }
```

nexample

```
7141 \bool_if:NTF \c__notesslides_notes_bool {
7142   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7143 }{
7144   \excludecomment{nexample}
7145 }
```

\inputref@*skip We customize the hooks for in \inputref.

```
7146 \def\inputref@preskip{\smallskip}
7147 \def\inputref@postskip{\medskip}
```

(End definition for \inputref@*skip. This function is documented on page ??.)

\inputref*

```
7148 \let\orig@inputref\inputref
7149 \def\inputref{\@ifstar\ninputref\orig@inputref}
7150 \newcommand\ninputref[2] []{
7151   \bool_if:NT \c__notesslides_notes_bool {
7152     \orig@inputref[#1]{#2}
7153   }
7154 }
```

(End definition for \inputref*. This function is documented on page 56.)

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 \TeX logo. Customization can be done by \setslidelogo{<logo name>}.

```
7155 \newlength{\slidelogoheight}
7156
7157 \RequirePackage{graphicx}
7158
7159 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7160 \providecommand\mhgraphics[2] []{
7161   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7162   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7163 }
7164
7165 \bool_if:NTF \c__notesslides_notes_bool {
7166   \setlength{\slidelogoheight}{.4cm}
7167 }{
7168   \setlength{\slidelogoheight}{.25cm}
7169 }
```



```

7170 \ifcsname slidelogo\endcsname\else
7171 \newsavebox{\slidelogo}
7172 \sbox{\slidelogo}{\sTeX}
7173 \fi
7174 \newrobustcmd{\setslidelogo}[2][]{
7175 \tl_if_empty:nTF{#1}{
7176 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7177 }{
7178 \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7179 }
7180 }

```

(End definition for `\setslidelogo`. This function is documented on page 57.)

\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.

```

7181 \bool_if:NT \c__notesslides_notes_bool {
7182 \def\author{\@dblarg\@ns@author}
7183 \long\def\@ns@author[#1]#2{%
7184 \def\c__notesslides_shortauthor{#1}%
7185 \def\@author{#2}
7186 }
7187 }

```

(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.

```

7188 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

(End definition for `\setsource`. This function is documented on page 57.)

\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.

```

7189 \def\copyrightnotice{%
7190 \footnotesize\copyright : \hspace{.3ex}%
7191 \ifcsname source\endcsname\source\else%
7192 \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7193 \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7194 ?source/author?\fi%
7195 \fi}
7196 \newsavebox{\cclogo}
7197 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7198 \newif\ifcchref\cchreffalse
7199 \AtBeginDocument{
7200 \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7201 }
7202 \def\licensing{
7203 \ifcchref
7204 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7205 \else
7206 {\usebox{\cclogo}}

```

```

7207 \fi
7208 }
7209 \newrobustcmd{\setlicensing}[2][]{
7210   \def\@url{#1}
7211   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7212   \ifx\@url\@empty
7213     \def\licensing{\usebox{\cclogo}}
7214   \else
7215     \def\licensing{
7216       \ifcchref
7217         \href{#1}{\usebox{\cclogo}}
7218       \else
7219         {\usebox{\cclogo}}
7220     \fi
7221   }
7222 \fi
7223 }

```

(End definition for \setlicensing. This function is documented on page 57.)

EdN:11

```

\slidelabel Now, we set up the slide label for the article mode.11
7224 \newrobustcmd\miko@slidelabel{
7225   \vbox to \slidelogoheight{
7226     \vss\hbox to \slidewidth
7227       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7228   }
7229 }

```

(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.

```

7230 \def\Gin@mhrepos{}
7231 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7232 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7233 \newrobustcmd\frameimage[2][]{
7234   \stepcounter{slide}
7235   \bool_if:NT \c__notesslides_frameimages_bool {
7236     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7237     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7238     \begin{center}
7239       \bool_if:NTF \c__notesslides_fiboxed_bool {
7240         \fbox{
7241           \ifx\Gin@ewidth\@empty
7242             \ifx\Gin@mhrepos\@empty
7243               \mhgraphics[width=\slidewidth,#1]{#2}
7244             \else
7245               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7246             \fi
7247           \else% Gin@ewidth empty

```

¹¹EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

7248         \ifx\Gin@mhrepos\@empty
7249             \mhgraphics[#1]{#2}
7250         \else
7251             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7252         \fi
7253     \fi% Gin@ewidth empty
7254 }
7255 }{
7256     \ifx\Gin@ewidth\@empty
7257     \ifx\Gin@mhrepos\@empty
7258         \mhgraphics[width=\slidewidth,#1]{#2}
7259     \else
7260         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7261     \fi
7262     \ifx\Gin@mhrepos\@empty
7263         \mhgraphics[#1]{#2}
7264     \else
7265         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7266     \fi
7267     \fi% Gin@ewidth empty
7268 }
7269 \end{center}
7270 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7271 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7272 }
7273 } % ifmks@sty@frameimages

```

(End definition for `\frameimage`. This function is documented on page 57.)

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`.

```

7274 \stex_html_backend:F {
7275     \bool_if:NT \c__notesslides_sectocframes_bool {
7276         \str_if_eq:VnTF \__notesslidesstopsect{part}{
7277             \newcounter{chapter}\counterwithin*{section}{chapter}
7278         }{
7279             \str_if_eq:VnT\__notesslidesstopsect{chapter}{
7280                 \newcounter{chapter}\counterwithin*{section}{chapter}
7281             }
7282         }
7283     }
7284 }

```

`\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
7285 \def\part@prefix{}
7286 \@ifpackageloaded{document-structure}{
7287     \str_case:VnF \__notesslidesstopsect {

```

```

7288 {part}{
7289   \int_set:Nn \l_document_structure_section_level_int {0}
7290   \def\thesection{\arabic{chapter}.\arabic{section}}
7291   \def\part@prefix{\arabic{chapter}.}
7292 }
7293 {chapter}{
7294   \int_set:Nn \l_document_structure_section_level_int {1}
7295   \def\thesection{\arabic{chapter}.\arabic{section}}
7296   \def\part@prefix{\arabic{chapter}.}
7297 }
7298 }{
7299   \int_set:Nn \l_document_structure_section_level_int {2}
7300   \def\part@prefix{}
7301 }
7302 }
7303
7304 \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`

```

7305 \renewenvironment{sfragment}[2][]{
7306   \__document_structure_sfragment_args:n { #1 }
7307   \int_incr:N \l_document_structure_section_level_int
7308   \bool_if:NT \c__notesslides_sectocframes_bool {
7309     \stepcounter{slide}
7310     \begin{frame}[noframenumbering]
7311     \vfill\Large\centering
7312     \red{
7313       \ifcase\l_document_structure_section_level_int\or
7314         \stepcounter{part}
7315         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7316         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7317         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7318         \def\currentsectionlevel{\omdoc@part@kw}
7319       \or
7320         \stepcounter{chapter}
7321         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7322         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7323         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7324         \def\currentsectionlevel{\omdoc@chapter@kw}
7325       \or
7326         \stepcounter{section}
7327         \def\__notesslideslabel{\part@prefix\arabic{section}}
7328         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7329         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7330         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7331         \def\currentsectionlevel{\omdoc@section@kw}
7332       \or
7333         \stepcounter{subsection}
7334         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7335         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

7336         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7337         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7338         \def\currentsectionlevel{\omdoc@subsection@kw}
7339     \or
7340         \stepcounter{subsubsection}
7341         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7342         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7343         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\thesubsubsection}
7344         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7345         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7346     \or
7347         \stepcounter{paragraph}
7348         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7349         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7350         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\theparagraph}
7351         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\theparagraph}
7352         \def\currentsectionlevel{\omdoc@paragraph@kw}
7353     \else
7354         \def\__notesslideslabel{}
7355         \def\currentsectionlevel{\omdoc@paragraph@kw}
7356     \fi% end ifcase
7357     \__notesslideslabel\quad #2%
7358 }%
7359 \vfill%
7360 \end{frame}%
7361 }
7362 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7363     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7364 }
7365 }{}
7366 }

```

We set up a beamer template for theorems like ams style, but without a block environment.

```

7367 \def\inserttheorembodyfont{\normalfont}
7368 %\bool_if:NF \c__notesslides_notes_bool {
7369 %   \defbeamertemplate{theorem begin}{miko}
7370 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7371 %    \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7372 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
7373 %   \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

7374 % \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.

```

7375 % \expandafter\def\csname Parent2\endcsname{}
7376 %}
7377
7378 \AddToHook{begindocument}{ % this does not work for some reason
7379     \setbeamertemplate{theorems}[ams style]
7380 }
7381 \bool_if:NT \c__notesslides_notes_bool {
7382     \renewenvironment{columns}[1][{}]{%

```

```

7383     \par\noindent%
7384     \begin{minipage}%
7385     \slidewidth\centering\leavevmode%
7386   }{%
7387     \end{minipage}\par\noindent%
7388   }%
7389   \newsavebox\columnbox%
7390   \renewenvironment<>{column}[2][]{%
7391     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7392   }{%
7393     \end{minipage}\end{lrbox}\usebox\columnbox%
7394   }%
7395 }

7396 \bool_if:NTF \c__notesslides_noproblems_bool {
7397   \newenvironment{problems}{}{}
7398 }{
7399   \excludacomment{problems}
7400 }

```

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.

```

7401 \gdef\printexcursions{}
7402 \newcommand\excursionref[2]{% label, text
7403   \bool_if:NT \c__notesslides_notes_bool {
7404     \begin{sparagraph}[title=Excursion]
7405       #2 \sref[fallback=the appendix]{#1}.
7406     \end{sparagraph}
7407   }
7408 }
7409 \newcommand\activate@excursion[2][]{
7410   \gappto\printexcursions{\inputref{#1}{#2}}
7411 }
7412 \newcommand\excursion[4][]{% repos, label, path, text
7413   \bool_if:NT \c__notesslides_notes_bool {
7414     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7415   }
7416 }

```

(End definition for `\excursion`. This function is documented on page 58.)

\excursiongroup

```

7417 \keys_define:nn{notesslides / excursiongroup }{
7418   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7419   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7420   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7421 }
7422 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7423   \tl_clear:N \l__notesslides_excursion_intro_tl
7424   \str_clear:N \l__notesslides_excursion_id_str

```

```

7425 \str_clear:N \l__notesslides_excursion_mhrepos_str
7426 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7427 }
7428 \newcommand\excursionsgroup[1][]{
7429   \__notesslides_excursion_args:n{ #1 }
7430   \ifdefempty\printexcursions{}% only if there are excursions
7431   {\begin{note}
7432     \begin{sfragment}[#1]{Excursions}%
7433     \ifdefempty\l__notesslides_excursion_intro_tl}{
7434       \inputref[\l__notesslides_excursion_mhrepos_str]{
7435         \l__notesslides_excursion_intro_tl
7436       }
7437     }
7438     \printexcursions%
7439     \end{sfragment}
7440   \end{note}}
7441 }
7442 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7443 \end{package}

```

(End definition for \excursionsgroup. This function is documented on page 58.)

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.

```
7444 <*package>
7445 <@@=problems>
7446 \ProvidesExplPackage{problem}{2022/05/24}{3.1.0}{Semantic Markup for Problems}
7447 \RequirePackage{13keys2e}
7448 \RequirePackage{amssymb}% for \Box
7449
7450 \keys_define:nn { problem / pkg }{
7451   notes      .default:n    = { true },
7452   notes      .bool_set:N   = \c__problems_notes_bool,
7453   gnotes     .default:n    = { true },
7454   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7455   hints      .default:n    = { true },
7456   hints      .bool_set:N   = \c__problems_hints_bool,
7457   solutions  .default:n    = { true },
7458   solutions  .bool_set:N   = \c__problems_solutions_bool,
7459   pts        .default:n    = { true },
7460   pts        .bool_set:N   = \c__problems_pts_bool,
7461   min        .default:n    = { true },
7462   min        .bool_set:N   = \c__problems_min_bool,
7463   boxed      .default:n    = { true },
7464   boxed      .bool_set:N   = \c__problems_boxed_bool,
7465   unknown    .code:n       = {
7466     \PassOptionsToPackage{\CurrentOption}{stex}
7467   }
7468 }
7469 \newif\ifsolutions
7470
7471 \ProcessKeysOptions{ problem / pkg }
7472 \bool_if:NTF \c__problems_solutions_bool {
7473   \solutionstrue
7474 }{
7475   \solutionsfalse
```



```

7476 }
7477 \RequirePackage{stex}

```

Then we make sure that the necessary packages are loaded (in the right versions).

```

7478 \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.

```

7479 \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.

```

7480 \def\prob@problem@kw{Problem}
7481 \def\prob@solution@kw{Solution}
7482 \def\prob@hint@kw{Hint}
7483 \def\prob@note@kw{Note}
7484 \def\prob@gnote@kw{Grading}
7485 \def\prob@pt@kw{pt}
7486 \def\prob@min@kw{min}
7487 \def\prob@correct@kw{Correct}
7488 \def\prob@wrong@kw{Wrong}

```

(End definition for `\prob@*@kw`. This function is documented on page ??.)

For the other languages, we set up triggers

```

7489 \AddToHook{begindocument}{
7490   \ltx@ifpackageloaded{babel}{
7491     \makeatletter
7492     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7493     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7494       \input{problem-ngerman.ldf}
7495     }
7496     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
7497       \input{problem-finnish.ldf}
7498     }
7499     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
7500       \input{problem-french.ldf}
7501     }
7502     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
7503       \input{problem-russian.ldf}
7504     }
7505     \makeatother
7506   }{}
7507 }

```

38.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```

7508 \keys_define:nn{ problem / problem }{
7509   id      .str_set_x:N = \l__problems_prob_id_str,
7510   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7511   min     .tl_set:N    = \l__problems_prob_min_tl,

```

```

7512 title .tl_set:N = \l__problems_prob_title_tl,
7513 type .tl_set:N = \l__problems_prob_type_tl,
7514 imports .tl_set:N = \l__problems_prob_imports_tl,
7515 name .str_set_x:N = \l__problems_prob_name_str,
7516 refnum .int_set:N = \l__problems_prob_refnum_int
7517 }
7518 \cs_new_protected:Nn \l__problems_prob_args:n {
7519 \str_clear:N \l__problems_prob_id_str
7520 \str_clear:N \l__problems_prob_name_str
7521 \tl_clear:N \l__problems_prob_pts_tl
7522 \tl_clear:N \l__problems_prob_min_tl
7523 \tl_clear:N \l__problems_prob_title_tl
7524 \tl_clear:N \l__problems_prob_type_tl
7525 \tl_clear:N \l__problems_prob_imports_tl
7526 \int_zero_new:N \l__problems_prob_refnum_int
7527 \keys_set:nn { problem / problem }{ #1 }
7528 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7529 \let\l__problems_prob_refnum_int\undefined
7530 }
7531 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7532 \newcounter{problem}[section]
7533 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
7534 \def\theplainsproblem{\arabic{problem}}
7535 \def\thesproblem{\thesection.\theplainsproblem}

```

(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.

```

7536 \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

```

7537 \newcommand\prob@number{
7538 \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7539 \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7540 }{
7541 \int_if_exist:NTF \l__problems_prob_refnum_int {
7542 \prob@label{\int_use:N \l__problems_prob_refnum_int }
7543 }{
7544 \prob@label\theplainsproblem
7545 }
7546 }
7547 }
7548 \def\sproblemautorefname{\prob@problem@kw}

```

(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.

```

7549 \newcommand\prob@title[3]{%
7550   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7551     #2 \l__problems_inclprob_title_tl #3
7552   }{
7553     \tl_if_empty:NTF \l__problems_prob_title_tl {
7554       #1
7555     }{
7556       #2 \l__problems_prob_title_tl #3
7557     }
7558   }
7559 }

```

(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.

```

7560 \def\prob@heading{
7561   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7562   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
7563 }

```

(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

```

7564 \newenvironment{sproblem}[1][]{
7565   \__problems_prob_args:n{#1}%\sref@target%
7566   \@in@omtexttrue% we are in a statement (for inline definitions)
7567   \refstepcounter{sproblem}\record@problem
7568   \def\current@section@level{\prob@problem@kw}
7569
7570   \str_if_empty:NT \l__problems_prob_name_str {
7571     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7572     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7573     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7574   }
7575
7576   \stex_if_do_html:T{
7577     \tl_if_empty:NF \l__problems_prob_title_tl {
7578       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7579     }
7580   }
7581
7582   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7583
7584   \stex_reactivate_macro:N \STEXexport
7585   \stex_reactivate_macro:N \importmodule
7586   \stex_reactivate_macro:N \symdecl
7587   \stex_reactivate_macro:N \notation
7588   \stex_reactivate_macro:N \symdef

```

```

7589 \stex_if_do_html:T{
7590   \begin{stex_annotate_env} {problem} {
7591     \l_stex_module_ns_str ? \l_stex_module_name_str
7592   }
7593
7594
7595   \stex_annotate_invisible:nnn{header}{} {
7596     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
7597     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7598     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7599       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7600     }
7601   }
7602 }
7603
7604 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7605
7606
7607 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7608   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7609 }{
7610   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7611 }
7612 \str_if_exist:NTF \l__problems_inclprob_id_str {
7613   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7614 }{
7615   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7616 }
7617
7618
7619 \stex_if_smsmode:F {
7620   \clist_set:No \l_tmpa_clist \sproblemtype
7621   \tl_clear:N \l_tmpa_tl
7622   \clist_map_inline:Nn \l_tmpa_clist {
7623     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7624       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7625     }
7626   }
7627   \tl_if_empty:NTF \l_tmpa_tl {
7628     \__problems_sproblem_start:
7629   }{
7630     \l_tmpa_tl
7631   }
7632 }
7633 \stex_ref_new_doc_target:n \sproblemid
7634 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
7635 }{
7636   \__stex_modules_end_module:
7637   \stex_if_smsmode:F{
7638     \clist_set:No \l_tmpa_clist \sproblemtype
7639     \tl_clear:N \l_tmpa_tl
7640     \clist_map_inline:Nn \l_tmpa_clist {
7641       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7642         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}

```

```

7643     }
7644   }
7645   \tl_if_empty:NTF \l_tmpa_tl {
7646     \__problems_sproblem_end:
7647   }{
7648     \l_tmpa_tl
7649   }
7650 }
7651 \stex_if_do_html:T{
7652   \end{stex_annotate_env}
7653 }
7654
7655 \smallskip
7656 }
7657
7658 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7659
7660
7661
7662 \cs_new_protected:Nn \__problems_sproblem_start: {
7663   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7664 }
7665 \cs_new_protected:Nn \__problems_sproblem_end: { \par\smallskip}
7666
7667 \newcommand\stexpatchproblem[3][] {
7668   \str_set:Nx \l_tmpa_str{ #1 }
7669   \str_if_empty:NTF \l_tmpa_str {
7670     \tl_set:Nn \__problems_sproblem_start: { #2 }
7671     \tl_set:Nn \__problems_sproblem_end: { #3 }
7672   }{
7673     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7674     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7675   }
7676 }
7677
7678
7679 \bool_if:NT \c__problems_boxed_bool {
7680   \surroundwithhmdframed{problem}
7681 }

```

\record@problem This macro records information about the problems in the *.aux file.

```

7682 \def\record@problem{
7683   \protected@write\@auxout{}
7684   {
7685     \string\@problem{\prob@number}
7686     {
7687       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7688         \l__problems_inclprob_pts_tl
7689       }{
7690         \l__problems_prob_pts_tl
7691       }
7692     }%
7693     {
7694       \tl_if_exist:NTF \l__problems_inclprob_min_tl {

```

```

7695         \l__problems_inclprob_min_tl
7696     }{
7697         \l__problems_prob_min_tl
7698     }
7699 }
7700 }
7701 }

```

(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).

```

7702 \def\@problem#1#2#3{

```

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

solution 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.

```

7703 \keys_define:nn { problem / solution }{
7704   id          .str_set_x:N = \l__problems_solution_id_str ,
7705   for         .str_set_x:N = \l__problems_solution_for_str ,
7706   type        .str_set_x:N = \l__problems_solution_type_str ,
7707   title       .tl_set:N    = \l__problems_solution_title_tl
7708 }
7709 \cs_new_protected:Nn \__problems_solution_args:n {
7710   \str_clear:N \l__problems_solution_id_str
7711   \str_clear:N \l__problems_solution_type_str
7712   \str_clear:N \l__problems_solution_for_str
7713   \tl_clear:N \l__problems_solution_title_tl
7714   \keys_set:nn { problem / solution }{ #1 }
7715 }

```

\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.

```

7716 \box_new:N \l__problems_solution_box
7717 \newenvironment{solution}[1][{}{
7718   \__problems_solution_args:n{#1}
7719   \stex_html_backend:TF{
7720     \stex_if_do_html:T{
7721       \begin{stex_annotate_env}{solution}{}
7722       \str_if_empty:NF \l__problems_solution_type_str {
7723         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
7724       }
7725       \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
7726     }
7727   }{
7728     \setbox\l__problems_solution_box\vbox\bgroup
7729     \par\smallskip\hrule\smallskip
7730     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
7731   }
7732 }{
7733   \stex_html_backend:TF{
7734     \stex_if_do_html:T{
7735       \end{stex_annotate_env}

```

```

7736     }
7737   }{
7738     \smallskip\hrule
7739     \egroup
7740     \bool_if:NT \c__problems_solutions_bool {
7741       \box\l__problems_solution_box
7742     }
7743   }
7744 }
7745
7746 \newcommand\startsolutions{
7747   \bool_set_true:N \c__problems_solutions_bool
7748   \solutionstrue
7749   % \specialcomment{solution}{\@startsolution}{
7750   %   \bool_if:NF \c__problems_boxed_bool {
7751   %     \hrule\medskip
7752   %   }
7753   %   \end{small}%
7754   % }
7755   % \bool_if:NT \c__problems_boxed_bool {
7756   %   \surroundwithmdframed{solution}
7757   % }
7758 }

```

(End definition for \startsolutions. This function is documented on page 60.)

\stopsolutions

```

7759 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

(End definition for \stopsolutions. This function is documented on page 60.)

exnote

```

7760 \bool_if:NTF \c__problems_notes_bool {
7761   \newenvironment{exnote}[1][ ]{
7762     \par\smallskip\hrule\smallskip
7763     \noindent\textbf{\prob@note@kw :~ }\small
7764   }{
7765     \smallskip\hrule
7766   }
7767 }{
7768   \excludacomment{exnote}
7769 }

```

hint

```

7770 \bool_if:NTF \c__problems_notes_bool {
7771   \newenvironment{hint}[1][ ]{
7772     \par\smallskip\hrule\smallskip
7773     \noindent\textbf{\prob@hint@kw :~ }\small
7774   }{
7775     \smallskip\hrule
7776   }
7777   \newenvironment{exhint}[1][ ]{
7778     \par\smallskip\hrule\smallskip
7779     \noindent\textbf{\prob@hint@kw :~ }\small

```

```

7780 }{
7781   \smallskip\hrule
7782 }
7783 }{
7784   \excludecomment{hint}
7785   \excludecomment{exhint}
7786 }

```

gnote

```

7787 \bool_if:NTF \c__problems_notes_bool {
7788   \newenvironment{gnote}[1][]{
7789     \par\smallskip\hrule\smallskip
7790     \noindent\textbf{\prob@gnote@kw :~ }\small
7791   }{
7792     \smallskip\hrule
7793   }
7794   }{
7795     \excludecomment{gnote}
7796   }

```

38.3 Marup for Added Value Services

38.4 Multiple Choice Blocks

EdN:12

mcb

12

```

7797 \newenvironment{mcb}{
7798   \begin{enumerate}
7799 }{
7800   \end{enumerate}
7801 }

```

we define the keys for the mcb macro

```

7802 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7803   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7804     \bool_set_true:N #1
7805   }{
7806     \bool_set_false:N #1
7807   }
7808 }
7809 \keys_define:nn { problem / mcb }{
7810   id          .str_set_x:N = \l__problems_mcc_id_str ,
7811   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7812   T           .default:n    = { false } ,
7813   T           .bool_set:N   = \l__problems_mcc_t_bool ,
7814   F           .default:n    = { false } ,
7815   F           .bool_set:N   = \l__problems_mcc_f_bool ,
7816   Ttext       .tl_set:N     = \l__problems_mcc_Ttext_tl ,
7817   Ftext       .tl_set:N     = \l__problems_mcc_Ftext_tl
7818 }
7819 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

¹²EdNOTE: MK: maybe import something better here from a dedicated MC package


```

7820 \str_clear:N \l__problems_mcc_id_str
7821 \tl_clear:N \l__problems_mcc_feedback_tl
7822 \bool_set_false:N \l__problems_mcc_t_bool
7823 \bool_set_false:N \l__problems_mcc_f_bool
7824 \tl_clear:N \l__problems_mcc_Ttext_tl
7825 \tl_clear:N \l__problems_mcc_Ftext_tl
7826 \str_clear:N \l__problems_mcc_id_str
7827 \keys_set:nn { problem / mcc }{ #1 }
7828 }

```

\mcc

```

7829 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
7830 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
7831 \newcommand\mcc[2][]{
7832   \l__problems_mcc_args:n{ #1 }
7833   \item[$\Box$] #2
7834   \bool_if:NT \c__problems_solutions_bool{
7835     \
7836     \bool_if:NT \l__problems_mcc_t_bool {
7837       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7838     }
7839     \bool_if:NT \l__problems_mcc_f_bool {
7840       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7841     }
7842     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7843       \emph{\l__problems_mcc_feedback_tl}
7844     }
7845   }
7846 } %solutions

```

(End definition for \mcc. This function is documented on page 61.)

38.5 Filling in Concrete Solutions

\includeproblem This is embarrassingly simple, but can grow over time.

```

7847 \newcommand\fillinsol[1]{\quad%
7848   \ifsolutions\textcolor{red}{\#!}\else%
7849   \fbox{\phantom{\huge{\#!}}}%
7850   \fi}

```

(End definition for \includeproblem. This function is documented on page 63.)

38.6 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.

```

7851
7852 \keys_define:nn{ problem / inclproblem }{
7853   id      .str_set_x:N = \l__problems_inclprob_id_str,
7854   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7855   min     .tl_set:N    = \l__problems_inclprob_min_tl,

```

```

7856 title .tl_set:N = \l__problems_inclprob_title_tl,
7857 refnum .int_set:N = \l__problems_inclprob_refnum_int,
7858 type .tl_set:N = \l__problems_inclprob_type_tl,
7859 mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
7860 }
7861 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7862 \str_clear:N \l__problems_prob_id_str
7863 \tl_clear:N \l__problems_inclprob_pts_tl
7864 \tl_clear:N \l__problems_inclprob_min_tl
7865 \tl_clear:N \l__problems_inclprob_title_tl
7866 \tl_clear:N \l__problems_inclprob_type_tl
7867 \int_zero_new:N \l__problems_inclprob_refnum_int
7868 \str_clear:N \l__problems_inclprob_mhrepos_str
7869 \keys_set:nn { problem / inclproblem }{ #1 }
7870 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7871 \let\l__problems_inclprob_pts_tl\undefined
7872 }
7873 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7874 \let\l__problems_inclprob_min_tl\undefined
7875 }
7876 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7877 \let\l__problems_inclprob_title_tl\undefined
7878 }
7879 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7880 \let\l__problems_inclprob_type_tl\undefined
7881 }
7882 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7883 \let\l__problems_inclprob_refnum_int\undefined
7884 }
7885 }
7886
7887 \cs_new_protected:Nn \l__problems_inclprob_clear: {
7888 \let\l__problems_inclprob_id_str\undefined
7889 \let\l__problems_inclprob_pts_tl\undefined
7890 \let\l__problems_inclprob_min_tl\undefined
7891 \let\l__problems_inclprob_title_tl\undefined
7892 \let\l__problems_inclprob_type_tl\undefined
7893 \let\l__problems_inclprob_refnum_int\undefined
7894 \let\l__problems_inclprob_mhrepos_str\undefined
7895 }
7896 \l__problems_inclprob_clear:
7897
7898 \newcommand\includeproblem[2][ ]{
7899 \l__problems_inclprob_args:n{ #1 }
7900 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7901 \stex_html_backend:TF {
7902 \str_clear:N \l_tmpa_str
7903 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7904 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7905 }
7906 \stex_annotate_invisible:nnn{includeproblem}{
7907 \l_tmpa_str / #2
7908 }{ }
7909 }{

```

```

7910     \begingroup
7911     \inputreftrue
7912     \tl_if_empty:nTF{ ##1 }{
7913       \input{#2}
7914     }{
7915       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7916     }
7917   \endgroup
7918 }
7919 }
7920 \__problems_inclprob_clear:
7921 }

```

(End definition for `\includeproblem`. This function is documented on page 63.)

38.7 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.

```

7922 \AddToHook{enddocument}{
7923   \bool_if:NT \c__problems_pts_bool {
7924     \message{Total:~\arabic{pts}~points}
7925   }
7926   \bool_if:NT \c__problems_min_bool {
7927     \message{Total:~\arabic{min}~minutes}
7928   }
7929 }

```

The margin pars are reader-visible, so we need to translate

```

7930 \def\pts#1{
7931   \bool_if:NT \c__problems_pts_bool {
7932     \marginpar{#1~\prob@pt@kw}
7933   }
7934 }
7935 \def\min#1{
7936   \bool_if:NT \c__problems_min_bool {
7937     \marginpar{#1~\prob@min@kw}
7938   }
7939 }

```

`\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.

```

7940 \newcounter{pts}
7941 \def\show@pts{
7942   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7943     \bool_if:NT \c__problems_pts_bool {
7944       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7945       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7946     }
7947   }{
7948     \tl_if_exist:NT \l__problems_prob_pts_tl {
7949       \bool_if:NT \c__problems_pts_bool {

```

```

7950         \tl_if_empty:NT\l__problems_prob_pts_tl{
7951             \tl_set:Nn \l__problems_prob_pts_tl {0}
7952         }
7953         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7954         \addtocounter{pts}{\l__problems_prob_pts_tl}
7955     }
7956 }
7957 }
7958 }

```

(End definition for \show@pts. This function is documented on page ??.)
and now the same for the minutes

\show@min

```

7959 \newcounter{min}
7960 \def\show@min{
7961     \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7962         \bool_if:NT \c__problems_min_bool {
7963             \marginpar{\l__problems_inclprob_pts_tl\ min}
7964             \addtocounter{min}{\l__problems_inclprob_min_tl}
7965         }
7966     }{
7967         \tl_if_exist:NT \l__problems_prob_min_tl {
7968             \bool_if:NT \c__problems_min_bool {
7969                 \tl_if_empty:NT\l__problems_prob_min_tl{
7970                     \tl_set:Nn \l__problems_prob_min_tl {0}
7971                 }
7972                 \marginpar{\l__problems_prob_min_tl\ min}
7973                 \addtocounter{min}{\l__problems_prob_min_tl}
7974             }
7975         }
7976     }
7977 }
7978 \</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.

```
7979 \< *package>
7980 \ProvidesExplPackage{hwexam}{2022/05/24}{3.1.0}{homework assignments and exams}
7981 \RequirePackage{13keys2e}
7982
7983 \newif\iftest\testfalse
7984 \DeclareOption{test}{\testtrue}
7985 \newif\ifmultiple\multiplefalse
7986 \DeclareOption{multiple}{\multipletrue}
7987 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
7988 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7989 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7990 \RequirePackage{keyval}[1997/11/10]
7991 \RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
7992 \newcommand\hwexam@assignment@kw{Assignment}
7993 \newcommand\hwexam@given@kw{Given}
7994 \newcommand\hwexam@due@kw{Due}
7995 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
7996 \newcommand\hwexam@minutes@kw{minutes}
7997 \newcommand\correction@probs@kw{prob.}
7998 \newcommand\correction@pts@kw{total}
7999 \newcommand\correction@reached@kw{reached}
8000 \newcommand\correction@sum@kw{Sum}
8001 \newcommand\correction@grade@kw{grade}
8002 \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

```

8003 \AddToHook{begindocument}{
8004 \ltx@ifpackageloaded{babel}{
8005 \makeatletter
8006 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8007 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8008 \input{hwexam-ngerman.ldf}
8009 }
8010 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8011 \input{hwexam-finnish.ldf}
8012 }
8013 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8014 \input{hwexam-french.ldf}
8015 }
8016 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8017 \input{hwexam-russian.ldf}
8018 }
8019 \makeatother
8020 }{}
8021 }
8022

```

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.

```

8023 \newcounter{assignment}
8024 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8025 \keys_define:nn { hwexam / assignment } {
8026 id .str_set:N = \l_@@_assign_id_str,
8027 number .int_set:N = \l_@@_assign_number_int,
8028 title .tl_set:N = \l_@@_assign_title_tl,
8029 type .tl_set:N = \l_@@_assign_type_tl,
8030 given .tl_set:N = \l_@@_assign_given_tl,
8031 due .tl_set:N = \l_@@_assign_due_tl,
8032 loadmodules .code:n = {
8033 \bool_set_true:N \l_@@_assign_loadmodules_bool
8034 }
8035 }
8036 \cs_new_protected:Nn \_@@_assignment_args:n {
8037 \str_clear:N \l_@@_assign_id_str
8038 \int_set:Nn \l_@@_assign_number_int {-1}
8039 \tl_clear:N \l_@@_assign_title_tl
8040 \tl_clear:N \l_@@_assign_type_tl
8041 \tl_clear:N \l_@@_assign_given_tl
8042 \tl_clear:N \l_@@_assign_due_tl
8043 \bool_set_false:N \l_@@_assign_loadmodules_bool
8044 \keys_set:nn { hwexam / assignment }{ #1 }
8045 }

```

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.

```

8046 \newcommand\given@due[2]{
8047 \bool_lazy_all:nF {
8048 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8049 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8050 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8051 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8052 }{ #1 }
8053
8054 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8055 \tl_if_empty:NF \l_@@_assign_given_tl {
8056 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8057 }
8058 }{
8059 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8060 }
8061
8062 \bool_lazy_or:nnF {
8063 \bool_lazy_and_p:nn {
8064 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8065 }{
8066 \tl_if_empty_p:V \l_@@_assign_due_tl
8067 }
8068 }{
8069 \bool_lazy_and_p:nn {
8070 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8071 }{
8072 \tl_if_empty_p:V \l_@@_assign_due_tl
8073 }
8074 }{ ,~ }
8075
8076 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8077 \tl_if_empty:NF \l_@@_assign_due_tl {
8078 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8079 }
8080 }{
8081 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8082 }
8083
8084 \bool_lazy_all:nF {
8085 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8086 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8087 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8088 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8089 }{ #2 }
8090 }

```

`\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.

```

8091 \newcommand\assignment@title[3]{
8092 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8093 \tl_if_empty:NTF \l_@@_assign_title_tl {
8094 #1
8095 }{
8096 #2\l_@@_assign_title_tl#3
8097 }
8098 }{
8099 #2\l_@@_inclasssign_title_tl#3
8100 }
8101 }

```

(End definition for \assignment@title. This function is documented on page ??.)

\assignment@number Like \assignment@title only for the number, and no around part.

```

8102 \newcommand\assignment@number{
8103 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8104 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8105 \arabic{assignment}
8106 } {
8107 \int_use:N \l_@@_assign_number_int
8108 }
8109 }{
8110 \int_use:N \l_@@_inclasssign_number_int
8111 }
8112 }

```

(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 For the assignment environment we delegate the work to the @assignment environment that depends on whether multiple option is given.

```

8113 \newenvironment{assignment}[1][]{
8114 \_@@_assignment_args:n { #1 }
8115 %\sref@target
8116 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8117 \global\stepcounter{assignment}
8118 }{
8119 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8120 }
8121 \setcounter{sproblem}{0}
8122 \renewcommand\prob@label[1]{\assignment@number.##1}
8123 \def\current@section@level{\document@hwexamtype}
8124 %\sref@label{id}{\document@hwexamtype \thesection}
8125 \begin{@assignment}
8126 }{
8127 \end{@assignment}
8128 }

```


In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

8129 \def\ass@title{
8130 {\protect\document@hwexamtype}\arabic{assignment}
8131 \assignment@title{\;(\;)\;}\; -- \given@due{\;}
8132 }
8133 \ifmultiple
8134 \newenvironment{@assignment}{
8135 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8136 \begin{sfragment}[loadmodules]{\ass@title}
8137 }{
8138 \begin{sfragment}{\ass@title}
8139 }
8140 }{
8141 \end{sfragment}
8142 }

```

for the single-page case we make a title block from the same components.

```

8143 \else
8144 \newenvironment{@assignment}{
8145 \begin{center}\bf
8146 \Large@title\strut\
8147 \document@hwexamtype\arabic{assignment}\assignment@title{\;(\;)\;}\;
8148 \large\given@due{--\;}\;{\;--}
8149 \end{center}
8150 }{}
8151 \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.

```

8152 \keys_define:nn { hwexam / inclassignment } {
8153 %id .str_set_x:N = \l_@@_assign_id_str,
8154 number .int_set:N = \l_@@_inclassign_number_int,
8155 title .tl_set:N = \l_@@_inclassign_title_tl,
8156 type .tl_set:N = \l_@@_inclassign_type_tl,
8157 given .tl_set:N = \l_@@_inclassign_given_tl,
8158 due .tl_set:N = \l_@@_inclassign_due_tl,
8159 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8160 }
8161 \cs_new_protected:Nn \_@@_inclassignment_args:n {
8162 \int_set:Nn \l_@@_inclassign_number_int {-1}
8163 \tl_clear:N \l_@@_inclassign_title_tl
8164 \tl_clear:N \l_@@_inclassign_type_tl
8165 \tl_clear:N \l_@@_inclassign_given_tl
8166 \tl_clear:N \l_@@_inclassign_due_tl
8167 \str_clear:N \l_@@_inclassign_mhrepos_str
8168 \keys_set:nn { hwexam / inclassignment }{ #1 }
8169 }
8170 \_@@_inclassignment_args:n {}
8171
8172 \newcommand\inputassignment[2][{}]{

```

```

8173 \_@@_inclassassignment_args:n { #1 }
8174 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8175   \input{#2}
8176 }{
8177   \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8178     \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8179   }
8180 }
8181 \_@@_inclassassignment_args:n {}
8182 }
8183 \newcommand\includeassignment[2][]{
8184   \newpage
8185   \inputassignment[#1]{#2}
8186 }

```

(End definition for \in*assignment. This function is documented on page ??.)

39.4 Typesetting Exams

\quizheading

```

8187 \ExplSyntaxOff
8188 \newcommand\quizheading[1]{%
8189   \def\@tas{#1}%
8190   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8191   \ifx\@tas\@empty\else%
8192     \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8193   \fi%
8194 }
8195 \ExplSyntaxOn

```

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

\testheading

```

8196
8197 \def\hwexamheader{\input{hwexam-default.header}}
8198
8199 \def\hwexamminutes{
8200   \tl_if_empty:NTF \testheading@duration {
8201     {\testheading@min}~\hwexam@minutes@kw
8202   }{
8203     \testheading@duration
8204   }
8205 }
8206
8207 \keys_define:nn { hwexam / testheading } {
8208   min .tl_set:N = \testheading@min,
8209   duration .tl_set:N = \testheading@duration,
8210   reqpts .tl_set:N = \testheading@reqpts,
8211   tools .tl_set:N = \testheading@tools
8212 }
8213 \cs_new_protected:Nn \_@@_testheading_args:n {
8214   \tl_clear:N \testheading@min
8215   \tl_clear:N \testheading@duration

```

```

8216 \tl_clear:N \testheading@reqpts
8217 \tl_clear:N \testheading@tools
8218 \keys_set:nn { hwexam / testheading }{ #1 }
8219 }
8220 \newenvironment{testheading}[1][]{
8221 \_@@_testheading_args:n{ #1 }
8222 \newcount\check@time\check@time=\testheading@min
8223 \advance\check@time by -\theassignment@totalmin
8224 \newif\if@bonuspoints
8225 \tl_if_empty:NTF \testheading@reqpts {
8226 \@bonuspointsfalse
8227 }{
8228 \newcount\bonus@pts
8229 \bonus@pts=\theassignment@totalpts
8230 \advance\bonus@pts by -\testheading@reqpts
8231 \edef\bonus@pts{\the\bonus@pts}
8232 \@bonuspointstrue
8233 }
8234 \edef\check@time{\the\check@time}
8235
8236 \makeatletter\hwexamheader\makeatother
8237 }{
8238 \newpage
8239 }

```

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

\testspace

```

8240 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8241 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8242 \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.

```

8243 <@=problems>
8244 \renewcommand\@problem[3]{
8245 \stepcounter{assignment@probs}
8246 \def\__problemspts{#2}
8247 \ifx\__problemspts\@empty\else
8248 \addtocounter{assignment@totalpts}{#2}
8249 \fi
8250 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8251 \xdef\correction@probs{\correction@probs & #1}%
8252 \xdef\correction@pts{\correction@pts & #2}
8253 \xdef\correction@reached{\correction@reached &}

```

```

8254 }
8255 <@@=hwexam>

```

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

`\correction@table` This macro generates the correction table

```

8256 \newcounter{assignment@probs}
8257 \newcounter{assignment@totalpts}
8258 \newcounter{assignment@totalmin}
8259 \def\correction@probs{\correction@probs@kw}
8260 \def\correction@pts{\correction@pts@kw}
8261 \def\correction@reached{\correction@reached@kw}
8262 \stepcounter{assignment@probs}
8263 \newcommand\correction@table{
8264 \resizebox{\textwidth}{!}{%
8265 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8266 &\multicolumn{\theassignment@probs}{c|}||%|
8267 {\footnotesize\correction@forgrading@kw} &\\ \hline
8268 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8269 \correction@pts & \theassignment@totalpts & \\ \hline
8270 \correction@reached & & \[.7cm]\hline
8271 \end{tabular}}
8272 </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

- [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/graphics.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).

¹³EdNOTE: 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).