

The sTeX3 Package Collection *

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

2022-08-11

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 3.2 (last revised 2022-08-11)

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	20
3.3.3	Argument Modes	20
3.3.3	Mode-b Arguments	20
3.3.3	Mode-a Arguments	21
3.3.3	Mode-B Arguments	22
3.3.4	Type and Definiens Components	23
3.3.5	Precedences and Automated Bracketing	24
3.3.6	Variables	26
3.3.7	Variable Sequences	27
3.4	Module Inheritance and Structures	29
3.4.1	Multilinguality and Translations	29
3.4.2	Simple Inheritance and Namespaces	30
3.4.3	The mathstructure Environment	32
3.4.4	The copymodule Environment	35
3.4.5	The interpretmodule Environment	36
3.5	Primitive Symbols (The sTeX Metatheory)	37
4	Using sTeX Symbols	38
4.1	\symref and its variants	38
4.2	Marking Up Text and On-the-Fly Notations	39

5	sTeX Statements	43
5.1	Definitions, Theorems, Examples, Paragraphs	43
5.2	Proofs	46
5.3	Highlighting and Presentation Customizations	51
6	Cross References	53
7	Additional Packages	55
7.1	Tikzinput: Treating TIKZ code as images	55
7.2	Modular Document Structuring	56
7.2.1	Introduction	56
7.2.2	Package Options	56
7.2.3	Document Fragments	56
7.2.4	Ending Documents Prematurely	58
7.2.5	Global Document Variables	58
7.3	Slides and Course Notes	58
7.3.1	Introduction	58
7.3.2	Package Options	59
7.3.3	Notes and Slides	59
7.3.4	Customizing Header and Footer Lines	60
7.3.5	Frame Images	61
7.3.6	Excursions	62
7.4	Representing Problems and Solutions	63
7.4.1	Introduction	63
7.4.2	Problems and Solutions	63
7.4.3	Markup for Added-Value Services	65
	Multiple Choice Blocks	65
	Filling-In Concrete Solutions	66
7.4.4	Including Problems	67
7.5	Homeworks, Quizzes and Exams	68
7.5.1	Introduction	68
7.5.2	Package Options	68
7.5.3	Assignments	68
7.5.4	Including Assignments	68
7.5.5	Typesetting Exams	69
II	Documentation	70
8	sTeX-Basics	71
8.1	Macros and Environments	71
8.1.1	HTML Annotations	71
8.1.2	Babel Languages	72
8.1.3	Auxiliary Methods	72
9	sTeX-MathHub	73
9.1	Macros and Environments	73
9.1.1	Files, Paths, URIs	73
9.1.2	MathHub Archives	74
9.1.3	Using Content in Archives	75

10	sTeX-References	76
10.1	Macros and Environments	76
10.1.1	Setting Reference Targets	76
10.1.2	Using References	77
11	sTeX-Modules	78
11.1	Macros and Environments	78
11.1.1	The <code>smodule</code> environment	80
12	sTeX-Module Inheritance	82
12.1	Macros and Environments	82
12.1.1	SMS Mode	82
12.1.2	Imports and Inheritance	83
13	sTeX-Symbols	85
13.1	Macros and Environments	85
14	sTeX-Terms	87
14.1	Macros and Environments	87
15	sTeX-Structural Features	89
15.1	Macros and Environments	89
15.1.1	Structures	89
16	sTeX-Statements	90
16.1	Macros and Environments	90
17	sTeX-Proofs: Structural Markup for Proofs	91
18	sTeX-Metatheory	92
18.1	Symbols	92
III	Extensions	93
19	Tikzinput: Treating TIKZ code as images	94
19.1	Macros and Environments	94
20	document-structure: Semantic Markup for Open Mathematical Documents in L^AT_EX	95
21	NotesSlides – Slides and Course Notes	96
22	problem.sty: An Infrastructure for formatting Problems	97
23	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	98
IV	Implementation	99

24	STeX-Basics Implementation	100
24.1	The STeXDocument Class	100
24.2	Preliminaries	101
24.3	Messages and logging	101
24.4	HTML Annotations	102
24.5	Babel Languages	104
24.6	Persistence	106
24.7	Auxiliary Methods	106
25	STeX-MathHub Implementation	110
25.1	Generic Path Handling	110
25.2	PWD and kpsewhich	112
25.3	File Hooks and Tracking	113
25.4	MathHub Repositories	114
25.5	Using Content in Archives	119
26	STeX-References Implementation	124
26.1	Document URIs and URLs	125
26.2	Setting Reference Targets	126
26.3	Using References	129
27	STeX-Modules Implementation	136
27.1	The smodule environment	140
27.2	Invoking modules	146
28	STeX-Module Inheritance Implementation	148
28.1	SMS Mode	148
28.2	Inheritance	152
29	STeX-Symbols Implementation	158
29.1	Symbol Declarations	158
29.2	Notations	166
29.3	Variables	175
30	STeX-Terms Implementation	183
30.1	Symbol Invocations	183
30.2	Terms	191
30.3	Notation Components	196
30.4	Variables	198
30.5	Sequences	201
31	STeX-Structural Features Implementation	202
31.1	Imports with modification	203
31.2	The feature environment	211
31.3	Structure	211
32	STeX-Statements Implementation	222
32.1	Definitions	222
32.2	Assertions	228
32.3	Examples	231
32.4	Logical Paragraphs	234

33 The Implementation	239
33.1 Proofs	239
34 \TeX-Others Implementation	248
35 \TeX-Metatheory Implementation	250
36 Tikzinput Implementation	253
37 document-structure.sty Implementation	256
37.1 Package Options	256
37.2 Document Structure	257
37.3 Front and Backmatter	261
37.4 Global Variables	263
38 NotesSlides – Implementation	264
38.1 Class and Package Options	264
38.2 Notes and Slides	266
38.3 Header and Footer Lines	270
38.4 Frame Images	272
38.5 Sectioning	273
38.6 Excursions	276
39 The Implementation	278
39.1 Package Options	278
39.2 Problems and Solutions	279
39.3 Markup for Added Value Services	286
39.4 Multiple Choice Blocks	286
39.5 Filling in Concrete Solutions	287
39.6 Including Problems	287
39.7 Reporting Metadata	289
40 Implementation: The hwexam Package	291
40.1 Package Options	291
40.2 Assignments	292
40.3 Including Assignments	295
40.4 Typesetting Exams	296
40.5 Leftovers	298
41 References	299

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 `TeX` 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 $\text{\texttt{L}^A\text{\texttt{T}}_E\text{\texttt{X}}$ 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 $\text{\texttt{T}}_E\text{\texttt{X}}\text{\texttt{Live}}$ on your system as a $\text{\texttt{L}^A\text{\texttt{T}}_E\text{\texttt{X}}$ enthusiast. If not now is the time to install it; see [\[TL\]](#). You can usually update $\text{\texttt{T}}_E\text{\texttt{X}}\text{\texttt{Live}}$ via a package manager or the $\text{\texttt{T}}_E\text{\texttt{X}}\text{\texttt{Live}}$ manager **$\text{\texttt{t}}\text{\texttt{l}}\text{\texttt{m}}\text{\texttt{g}}\text{\texttt{r}}$** .

Alternatively, you can install $\text{\texttt{sTeX}}$ from CTAN, the Comprehensive $\text{\texttt{T}}_E\text{\texttt{X}}$ 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 \LaTeX Document

Having set everything up, we can write a first \LaTeX 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     \[\defeq{\geometricSeries}{\definiens{
14       \infinitiesum{\svar{n}}{1}{
15         \realdivide[frac]{1}{
16           \realpower{2}{\svar{n}}
17         }
18       }}
19     \].\]
20   \end{sdefinition}
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 ??.

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

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

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

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

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

Next, we *import* two modules – `series` from the \TeX archive `snglom/calculus`, and `realarith` from the \TeX archive `snglom/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. `\infinitiesum`, `\realdive`, `\realpower`) in the imported module available to the current module `GeometricSeries`. The module `GeometricSeries` “exports” all of these symbols to all modules imports it via an `\importmodule{GeometricSeries}` instruction. Additionally it exports the local symbol `\geometricSeries`.

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

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

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

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

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

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

```

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

```

The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) `series` and `realarithmetics`, such as `\defeq`, `\infinitiesum`, etc. In math mode, using a semantic macro inserts its (default) definition. A semantic macro can have several notations – in that case, we can explicitly choose a specific notation by providing its identifier as an optional argument; e.g. `\realdivide[frac]{a}{b}` will use the explicit notation named `frac` of the semantic macro `\realdivide`, which yields $\frac{a}{b}$ instead of a/b .

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

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

2.2.1 OMDoc/xhtml Conversion

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

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

...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 ??) 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 `\symdecl{symbolname}`, `\symdef{symbolname}` and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro* `\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*. `\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].

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

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

3.2.1 The Local MathHub-Directory

`\usemodule`, `\importmodule`, `\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/arithmetic,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
```

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

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

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

source-base or

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

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

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

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

3.2.4 Using Files in \TeX Archives Directly

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

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

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

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

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

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

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

`\libinput` `\libinput{some/file}` searches for a file `some/file` in

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

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

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

`\libusepackage` `\libusepackage[package-options]{some/file}` searches for a file `some/file.sty` in the same way that `\libinput` does, but will call `\usepackage[package-options]{path/to/some/file}` instead of `\input`.

`\libusepackage` throws an error if not *exactly one* candidate for `some/file` is found.

Remark 3.2.1:

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

```
1 \documentclass{stex}
2 \libinput{preamble}
3 \begin{document}
4   ...
5   \ifinputref \else \libinput{postamble} \fi
6 \end{document}
```

Then the `preamble.tex` files can take care of loading the generally required packages, setting presentation customizations etc. (per archive or archive group or both), and `postamble.tex` can e.g. print the bibliography, index etc.

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

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

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

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

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

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

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

Example 1

Input:

```

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

```

Output:

Hello World

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

For example:

Example 2

Input:

```

1 \stexpatchmodule[display]
2   {\textbf{Module (\smodulename)}}\par}
3   {\par\noindent\textbf{End of Module (\smodulename)}}}
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 $\binarysymbol[\text{highlight}]$:

Example 6

Input:

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

Output:

First: a ; Second: b



Ideally, `\comp` would not be necessary: Everything in a notation that is *not* an argument should be a notation component. Unfortunately, it is computationally expensive to determine where an argument begins and ends, and the argument markers `#n` may themselves be nested in other macro applications or \TeX groups, making it ultimately almost impossible to determine them automatically while also remaining compatible with arbitrary highlighting customizations (such as tooltips, hyperlinks, colors) that users might employ, and that are ultimately invoked by `\comp`.



Note that it is required that

1. the argument markers `#n` never occur inside a `\comp`, and
2. no semantic arguments may ever occur inside a notation.

Both criteria are not just required for technical reasons, but conceptionally meaningful:

The underlying principle is that the arguments to a semantic macro represent *arguments to the mathematical operation* represented by a symbol. For example, a semantic macro `\addition{a}{b}` taking two arguments would represent *the actual addition of (mathematical objects) a and b*. It should therefore be impossible for a or b to be part of a notation component of `\addition`.

Similarly, a semantic macro can not conceptually be part of the notation of `\addition`, since a semantic macro represents a *distinct mathematical concept* with *its own semantics*, whereas notations are syntactic representations of the very symbol to which the notation belongs.

If you want an argument to a semantic macro to be a purely syntactic parameter, then you are likely somewhat confused with respect to the distinction between the precise *syntax* and *semantics* of the symbol you are trying to declare (which happens quite often even to experienced \TeX users), and might want to give those another thought - quite likely, the macro you aim to implement does not actually represent a semantically meaningful mathematical concept, and you will want to use `\def` and similar native \LaTeX macro definitions rather than semantic macros.

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

Example 7

Input:

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

Output:

```
1.: a; 2.: b
```

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

But one man’s meat is another man’s poison: it is very subjective what the “default notation” of an operator should be. Different communities have different practices. For instance, the complex unit is written as i in Mathematics and as j in electrical engineering. So to allow modular specification and facilitate re-use of document fragments $\text{\texttt{STEX}}$ 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 $\text{\texttt{TeX}}$ ’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"/>` directly.

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 $\text{T}_{\text{E}}\text{X}$, but applications of binding operators, i.e. symbols with *mode-b* arguments, are translated to *OMBIND*-terms in OMDoc/MMT, rather than *OMA*.

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

Example 9

Input:

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

Output:

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

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

Mode-a Arguments

Mode-a arguments represent a *flexary argument sequence*, i.e. a sequence of arguments of arbitrary length. Formally, operators that take arbitrarily many arguments don't "exist", but in informal mathematics, they are ubiquitous. Mode-a arguments allow us to write e.g. `\addition{a,b,c,d,e}` rather than having to write something like `\addition{a}\addition{b}\addition{c}\addition{d}\addition{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 $\text{\texttt{STeX}}$ (or, rather, $\text{\texttt{MMT/OMDoc}}$) 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 $\forall x. \forall y. \forall z. P$, 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 $\text{\texttt{OMDoc/MMT}}$ 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.
- \hookrightarrow Correspondingly, the name “type” should be taken with a grain of salt, since
- \hookrightarrow OMDoc/MMT – being foundation-independent – does not a priori implement a fixed typing system.

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

Example 13

Input:

```

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

```

Output:

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

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

Example 14

Input:

```

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

```

Output:

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

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

```

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

```

Output:

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

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

Example 16

Input:

```

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

```

Output:

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

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

Example 17

Input:

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

Output:

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

but we can also do better by supplying *precedences* and have \TeX 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).

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.

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

When $\text{\S}\text{\TeX}$ steps into an argument of a semantic macro, it sets p_d to the respective argument precedence of the notation used.

In the example above:



1. $\text{\S}\text{\TeX}$ starts out with $p_d = \text{\code{\infprec}}$.
2. $\text{\S}\text{\TeX}$ encounters `\addition` with $p_{op} = 100$. Since $100 \not> \text{\code{\infprec}}$, it inserts no parentheses.
3. Next, $\text{\S}\text{\TeX}$ encounters the two arguments for `\addition`. Both have no specifically provided argument precedence, so $\text{\S}\text{\TeX}$ uses $p_d = p_{op} = 100$ for both and recurses.
4. Next, $\text{\S}\text{\TeX}$ encounters `\multiplication{b,...}`, whose notation has $p_{op} = 50$.
5. We compare to the current downward precedence p_d set by `\addition`, arriving at $p_{op} = 50 \not> 100 = p_d$, so $\text{\S}\text{\TeX}$ again inserts no parentheses.
6. Since the notation of `\multiplication` has no explicitly set argument precedences, $\text{\S}\text{\TeX}$ uses the operator precedence for all arguments of `\multiplication`, hence sets $p_d = p_{op} = 50$ and recurses.
7. Next, $\text{\S}\text{\TeX}$ encounters the inner `\addition{c,...}` whose notation has $p_{op} = 100$.
8. We compare to the current downward precedence p_d set by `\multiplication`, arriving at $p_{op} = 100 > 50 = p_d$ – which finally prompts $\text{\S}\text{\TeX}$ 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 `\symdecl`, `\notation`, `\symdef` etc. are disabled outside of `smodule`-environments.

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

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

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

Example 19

Input:

```

1 \vardef{varf}[
2   name=f,
3   type=\funtype{\Nat}{\Nat},
4   op=f,
5   args=1,
6   prec=0;\neginfpref
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 $\varf!:\funtype{\Nat}{\Nat}$,
12 by $\addition{\varf!,\varn}$ we mean the function\rustexBREAK
13 $\fun{\varx}{\varf{\addition{\varx,\varn}}}$

```

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 T_EX 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 $\seqa!$ is $\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 $\addition{\seqa}$
```

Output:

$a_1 + \dots + a_n$

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

Example 22

Input:

```

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

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

3.4.1 Multilinguality and Translations

If we load the \TeX document class or package with the option `lang=<lang>`, \TeX 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:
 \hookrightarrow `\begin{smodule}[lang=<lang>]{Foo}` generates a theory `some/namespace?Foo` that only contains the “formal” part of the module – i.e. exactly the content that is exported when using `\importmodule`.
 \rightsquigarrow Additionally, MMT generates a *language theory* `some/namespace/Foo?<lang>` that includes `some/namespace?Foo` and contains all the other document content – variable declarations, includes for each `\usemodule`, etc.

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

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

Additionally to translating the informal content of a module file to different languages, it also allows for customizing notations between languages. For example, the *least common multiple* of two numbers is often denoted as `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    $\text{lcm}\{a,b\}$  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` `\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, \LaTeX 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 \LaTeX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

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

In other words: outside of archives, the namespace corresponds to the file URI



with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix.

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



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

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

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

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.
- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document. Since this is less compatible with a modular development, using full URIs directly is strongly discouraged, unless the module is declared in the current file directly.

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



For persistency reasons, everything in an `\STEXexport` is digested by T_EX in the L^AT_EX3-category code scheme. This means that the characters `_` and `:` are considered *letters* and valid parts of control sequence names, and space characters are



ignored entirely. For spaces, use the character `~` instead, and keep in mind, that if you want to use subscripts, you should use `\c_math_subscript_token` instead of `_`!

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

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

3.4.3 The `mathstructure` Environment

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

- A *monoid* is a structure $\langle M, \circ, e \rangle$ with $\circ : M \times M \rightarrow M$ and $e \in M$ such that...
- A *topological space* is a structure $\langle X, \mathcal{T} \rangle$ where X is a set and \mathcal{T} is a topology on X
- A *partial order* is a structure $\langle S, \leq \rangle$ where \leq is a binary relation on S such that...

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

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

How this works is again best demonstrated by example:

Example 24

Input:

```

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

Output:

A *monoid* is...

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

Example 25

Input:

```

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

Output:

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

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

Example 26

Input:

```

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

Output:

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

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

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

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

- `mathstructure{<name>}` simply creates a nested theory with name $\hookrightarrow M \rightarrow$ `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)`
- $\hookrightarrow M \rightarrow$ – a *dependent record type with manifest fields*, the fields of which are generated
- $\rightsquigarrow T \rightsquigarrow$ 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  $\varMb := \mathsf{mathstruct}\{\varMb{universe}, \varMb{op}!, \varMb{unit}\}$ 
4 be a  $\mathsf{symname}\{\mathsf{monoid}\}$  on  $\mathbb{Z}$  ...

```

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

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

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

Example 32

Input:

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

Output:

A natural number is...

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

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

Example 33

Input:

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

Output:

Natural numbers are...



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

If `\symbolname` is a semantic macro, then \TeX has no trouble resolving `symbolname` to the full URI of the symbol that is being invoked.

However, especially in `\symname` (or if a symbol was introduced using `\symdecl*` without generating a semantic macro), we might prefer to use the *name* of a symbol directly for readability – e.g. we would want to write A `\symname{natural-number}` is... rather than A `\symname{Nat}` is.... \TeX attempts to handle this case thusly:

If `string` does *not* correspond to a semantic macro `\string` and does *not* contain a `?`, then \TeX checks all symbols currently in scope until it finds one, whose name is `string`. If `string` is of the form `pre?name`, \TeX first looks through all modules currently in scope, whose full URI ends with `pre`, and then looks for a symbol with name `name` in those. This allows for disambiguating more precisely, e.g. by saying `\symname{Integers?addition}` or `\symname{RealNumbers?addition}` in the case where several `additions` are in scope.

4.2 Marking Up Text and On-the-Fly Notations

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

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

Example 34

Input:

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

Output:

The sum of n and m is...

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

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



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

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

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

Example 35

Input:

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

Output:

```
Addition is...
```

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

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

Example 36

Input:

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

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

Output:

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...

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see [subsection 7.2.1](#)) seriously, and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputrefs` a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also inputted in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex` it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

What we would want in that instance is an equivalent of `\autoref`, that takes the document context into account to yield something like *Definition 1*, *Definition 3.1* or *Definition 1 in the section on Foo* respectively.

The `\sref` command attempts to do precisely that. Unlike plain `\ref`, `\autoref` etc., `\sref` refers to not just a *label*, but instead a pair consisting of a *label* and the *document* in whose context we want to refer to it. Conversely, every *document* (i.e. standalone compilable `.tex`-file) keeps track of the “names” (*Definition 3.1* etc.) for every label as determined in the context of the document, and stores them in a dedicated file `\jobname.sref`. Additionally, every document has a “reference name” (e.g. “*the section on Foo*”). This allows us to refer to “label x in document D ” to yield “*Definition 1 in the section on Foo*”. And of course, \TeX can decide based on the current document

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

to either refer to the label by its “full name” or directly as e.g. *Definition 3.1* depending on whether the label occurs in the current document anyway (and link to it accordingly).

For that to work, we need to supply (up to) three pieces of information:

- The *label* of the reference target (e.g. `some_definition`),
- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary, but allows for additional disambiguation between possibly duplicate labels across files, and
- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

Additionally, the document in which we want to reference a label needs to provide a title for external references.

`\STEXreftitle` For the latter, we can use `\STEXreftitle`. For example, the full \TeX documentation (`stex-doc.tex`) declares `\STEXreftitle{the full \stex{3} documentation}`, whereas the manual (`stex-manual.tex`) declares `\STEXreftitle{the \stex{3} manual}`.

`\sref` `\sref[archive=<archive1>,file=<file>]`
`{<label>}[archive=<archive2>,in=<document-context>,cite=<citation>]`

This command references *<label>* (declared in *<file>* in *<archive1>*). If the object (section, figure, etc.) with that label occurs ultimately in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the `hyperref` package is not included.

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

For example, the reference to the `sfragment`-environment above will appear as “subsection 7.2.1 (Introduction) in the \TeX 3 manual” if you are reading this in the package documentation for `stex-references` directly, but as a linked “subsection 7.2.1” in the full documentation or manual. This is achieved using

`\sref[file=stex-document-structure]{sec:ds:intro}[in=./stex-manual]`.

If the optional `cite=<citation>`-argument is provided, `\sref` will ignore the document title as provided by the document context, and will instead use “in `\cite{<citation>}`”.

For a further example, the following:

Part III

will say “Part III” (and link accordingly) in the full documentation, and “Part III (Extensions) in the full \TeX 3 documentation” everywhere else. This is achieved using

`\sref[file=./stex-doc]{part:extends}[in=./stex-doc]`.

`\extref` `\sref[archive=<archive1>,file=<file>]`
`{<label>}[archive=<archive2>,in=<document-context>,cite=<citation>]`

The `\extref`-command behaves exactly like `\sref`, but takes *required* the document context argument and will always use it for generating the document text, regardless of whether the label occurs in the current document.

Chapter 5

ST_EX 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 ?? 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 ??), `type=` for customization (see ??) and additional information (e.g. definition principles, “difficulty” etc), as well as `title=` (for giving the paragraph a title), and finally `for=`.

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

Example 38

Input:

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

Output:

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

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

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

EdN:4

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

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

```

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

```

⁴EdNOTE: MK: reference

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

5.2 Proofs

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

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{spproof}

```

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. □

sproof (*env.*) The **sproof** 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.

\spfilea The **\spfilea** 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 **sproof** 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 (*env.*) 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}}

```

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

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

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

Chapter 6

Cross References

If we take features like `\inputref` and `\mhinput` (and the `sfragment`-environment, see [subsection 7.2.1](#)) seriously, and build large documents modularly from individually compiling documents for sections, chapters and so on, cross-referencing becomes an interesting problem.

Say, we have a document `main.tex`, which `\inputrefs` a section `section1.tex`, which references a definition with label `some_definition` in `section2.tex` (subsequently also inputted in `main.tex`). Then the numbering of the definition will depend on the *document context* in which the document fragment `section2.tex` occurs - in `section2.tex` itself (as a standalone document), it might be *Definition 1*, in `main.tex` it might be *Definition 3.1*, and in `section1.tex`, the definition *does not even occur*, so it needs to be referenced by some other text.

What we would want in that instance is an equivalent of `\autoref`, that takes the document context into account to yield something like *Definition 1*, *Definition 3.1* or *Definition 1 in the section on Foo* respectively.

The `\sref` command attempts to do precisely that. Unlike plain `\ref`, `\autoref` etc., `\sref` refers to not just a *label*, but instead a pair consisting of a *label* and the *document* in whose context we want to refer to it. Conversely, every *document* (i.e. standalone compilable `.tex`-file) keeps track of the “names” (*Definition 3.1* etc.) for every label as determined in the context of the document, and stores them in a dedicated file `\jobname.sref`. Additionally, every document has a “reference name” (e.g. “*the section on Foo*”). This allows us to refer to “label *x* in document *D*” to yield “*Definition 1 in the section on Foo*”. And of course, `TEX` can decide based on the current document to either refer to the label by its “full name” or directly as e.g. *Definition 3.1* depending on whether the label occurs in the current document anyway (and link to it accordingly).

For that to work, we need to supply (up to) three pieces of information:

- The *label* of the reference target (e.g. `some_definition`),
- (optionally) the *file*/document containing the reference target (e.g. `section2`). This is not strictly necessary, but allows for additional disambiguation between possibly duplicate labels across files, and
- (optionally) the document context, in which we want to refer to the reference target (e.g. `main`).

Additionally, the document in which we want to reference a label needs to provide a title for external references.

\STEXreftitle For the latter, we can use `\STEXreftitle`. For example, the full \TeX documentation (`stex-doc.tex`) declares `\STEXreftitle{the full \stex{3} documentation}`, whereas the manual (`stex-manual.tex`) declares `\STEXreftitle{the \stex{3} manual}`.

\sref `\sref[archive=<archive1>,file=<file>]{<label>}[archive=<archive2>,in=<document-context>,cite=<citation>]`

This command references `<label>` (declared in `<file>` in `<archive1>`). If the object (section, figure, etc.) with that label occurs ultimately in the same document, `\sref` will ignore the second set of optional arguments and simply defer to `\autoref` if that command exists, or `\ref` if the `hyperref` package is not included.

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

For example, the reference to the `sfragment`-environment above will appear as “subsection 7.2.1 (Introduction) in the \TeX 3 manual” if you are reading this in the package documentation for `stex-references` directly, but as a linked “subsection 7.2.1” in the full documentation or manual. This is achieved using

```
\sref[file=stex-document-structure]{sec:ds:intro}[in=./stex-manual].
```

If the optional `cite=<citation>`-argument is provided, `\sref` will ignore the document title as provided by the document context, and will instead use “in `\cite{<citation>}`”.

For a further example, the following:

Part III

will say “Part III” (and link accordingly) in the full documentation, and “Part III (Extensions) in the full \TeX 3 documentation” everywhere else. This is achieved using

```
\sref[file=./stex-doc]{part:extends}[in=./stex-doc].
```

\extref `\sref[archive=<archive1>,file=<file>]{<label>}[archive=<archive2>,in=<document-context>,cite=<citation>]`

The `\extref`-command behaves exactly like `\sref`, but takes *required* the document context argument and will always use it for generating the document text, regardless of whether the label occurs in the current document.

Chapter 7

Additional Packages

7.1 Tikzinput: Treating TIKZ code as images

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

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

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

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

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

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

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

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

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

7.2 Modular Document Structuring

7.2.1 Introduction

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

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

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

7.2.2 Package Options

The document-structure package accepts the following options:

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

7.2.3 Document Fragments

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

```

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

```

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

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

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

```

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

```

Here we use two levels of `blindfragment`:

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

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

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

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

7.2.4 Ending Documents Prematurely

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

`\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.

7.2.5 Global Document Variables

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

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

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

7.3 Slides and Course Notes

7.3.1 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.

7.3.2 Package Options

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

<code>slides</code> <code>notes</code>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see subsection 7.3.3).
<code>sectocframes</code>	If the option <code>sectocframes</code> is given, then for the <code>sfragments</code> , special frames with the <code>sfragment</code> title (and number) are generated.
<code>frameimages</code> <code>fiboxed</code>	If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code> -generated frames (see ??). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.

7.3.3 Notes and Slides

`frame` (*env.*) Slides are represented with the `frame` environment just like in the `beamer` class, see [\[Tanb\]](#) for details.

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



Note that it is essential to start and end the `notes` environment at the start of the line – in particular, there may not be leading blanks – else L^AT_EX becomes confused and throws error messages that are difficult to decipher.

By interleaving the `frame` and `note` environments, we can build course notes as shown here:

```

1 \ifnotes\maketitle\else
2 \frame[noframenumbering]\maketitle\fi
3
4 \begin{note}
5   We start this course with ...
6 \end{note}
7
8 \begin{frame}
9   \frametitle{The first slide}
10  ...

```

```

11 \end{frame}
12 \begin{note}
13   ... and more explanatory text
14 \end{note}
15
16 \begin{frame}
17   \frametitle{The second slide}
18   ...
19 \end{frame}
20 ...

```

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



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



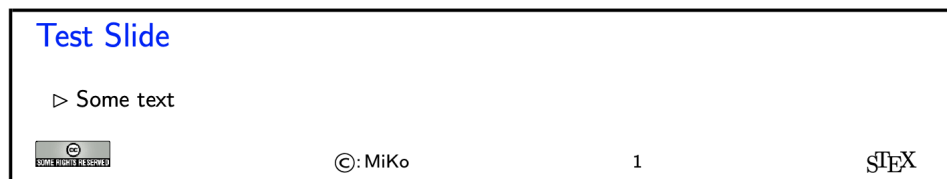
The `beamer` class recommends not to use the `allowframebreaks` option on frames (even though it is very convenient). This holds even more in the `notesslides` case: At least in conjunction with `\newpage`, frame numbering behaves funnily (we have tried to fix this, but who knows).

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

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

7.3.4 Customizing Header and Footer Lines

The `notesslides` package and class comes with a simple default theme named `sTeX` that provided by the `beamterthemesTeX`. It is assumed as the default theme for `sTeX`-based notes and slides. The result in `notes` mode (which is like the `slides` version except that the slide height is variable) is



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

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

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

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

7.3.5 Frame Images

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

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

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

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


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

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

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

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

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

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

7.3.6 Excursions

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

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

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

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

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

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

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

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

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



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

7.4 Representing Problems and Solutions

7.4.1 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.

7.4.2 Problems and Solutions

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

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

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

Example 40

Input:

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

```

1 \documentclass{article}
2 \usepackage[solutions,hints,pts,min]{problem}
3 \begin{document}
4   \begin{sproblem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
5     How many Elephants can you fit into a Volkswagen beetle?
6     \begin{hint}
7       Think positively, this is simple!
8     \end{hint}
9     \begin{exnote}
10      Justify your answer
11    \end{exnote}
12  \begin{solution}[for=elephants]
13    Four, two in the front seats, and two in the back.
14    \begin{gnote}
15      if they do not give the justification deduct 5 pts
16    \end{gnote}
17  \end{solution}
18 \end{sproblem}
19 \end{document}

```

Output:

Problem 7.4.1 (Fitting Elephants)
 How many Elephants can you fit into a Volkswagen beetle?

Hint: Think positively, this is simple!

Note: Justify your answer

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

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

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

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

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

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

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

`\mcc` `\mcc[⟨keyvals⟩]{⟨text⟩}` takes an optional key/value argument `⟨keyvals⟩` for choice 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 7.4.2 (Functions)

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

☐ def

Correct!

☐ function

Wrong! *that is for C and C++*

☐ fun

Wrong! *that is for Standard ML*

☐ public static void

Wrong! *that is for Java*

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

Example 42

Input:

```

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

```

Output:

Problem 7.4.3 (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 Elephants can you fit into a Volkswagen beetle? \fillinsol{4}
4 \end{problem}
```

Output:

Problem 7.4.4 (Fitting Elephants)

How many Elephants 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 Elephants can you fit into a Volkswagen beetle? \fillinsol{4}
3 \end{problem}
```

Output:

Problem 7.4.5 (Fitting Elephants)

How many Elephants 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

7.4.4 Including Problems

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

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

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.

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.

7.5 Homeworks, Quizzes and Exams

7.5.1 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.

7.5.2 Package Options

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

7.5.3 Assignments

<code>assignment</code> (<i>env.</i>)	This package supplies the <code>assignment</code> environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys <code>number</code> (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — 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>	

7.5.4 Including Assignments

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

7.5.5 Typesetting Exams

`\testspace` `\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 (env.)` 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.

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

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-08-11

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.	7.4.1	7.4.2	7.4.3	7.4.4	7.4.5	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

⁸EdNOTE: MK: The first three “problems” come from the stex examples above, how do we get rid of this?

Part II

Documentation

Chapter 8

STEX-Basics

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

8.1 Macros and Environments

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

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

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

8.1.1 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 `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 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`.

8.1.3 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 `⟨cs⟩` throw an error, indicating that it is only allowed in the context of `⟨environments⟩`.

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

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

Chapter 9

STEX-MathHub

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

9.1 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.
--------------------------------	---

9.1.1 Files, Paths, URIs

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

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

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

<code>\stex_path_if_absolute_p:N</code>	\star
<code>\stex_path_if_absolute:N\underline{T}</code>	\star

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

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

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

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

9.1.2 MathHub Archives

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

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

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

<code>\l_stex_current_repository_prop</code>
--

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

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

<code>\stex_set_current_repository:n</code>

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

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

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

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

9.1.3 Using Content in Archives

<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/> <code>\inputref</code> <hr/> <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/> <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/> <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/> <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/> <code>\mhgraphics</code> <hr/> <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/> <code>\lstinputmhlisting</code> <hr/> <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 10

STEX-References

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

10.1 Macros and Environments

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

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

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

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

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

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

10.1.1 Setting Reference Targets

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

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

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

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

10.1.2 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 11

sTeX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

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

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

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

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

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

`\l_stex_current_module_str` `\l_stex_current_module_str` always contains the URI of the current module (if existent).

`\l_stex_all_modules_seq` Stores full URIs for all modules currently in scope.

`\stex_if_in_module_p: *` Conditional for whether we are currently in a module
`\stex_if_in_module:TF *`

`\stex_if_module_exists_p:n *`
`\stex_if_module_exists:nTF *`

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

`\stex_collect_imports:n` 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 `\l_stex_collect_imports_seq`

`\stex_do_up_to_module:n` Code that is *exported* from module (such as symbol declarations) should be local *to the current module*. 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 `sparapraphs`. `\stex_do_up_to_module` therefore executes the provided code repeatedly in an `\aftergroup` 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`.

11.1.1 The smodule environment

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

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

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

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

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

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

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

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

sig (`\langle language \rangle`) if the current file is a translation of a file with the same base name but a different language suffix, setting `sig=<lang>` will preload the module from that language file. This helps ensuring that the (formal) content of both modules is (almost) identical across languages and avoids duplication.

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

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

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

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

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

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

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

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

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

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

`\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 12

STEX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

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

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

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

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

12.1.2 Imports and Inheritance

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

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

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

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

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

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

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

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

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

Chapter 13

STEX-Symbols

Code related to symbol declarations and notations

13.1 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 ST_EX, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by ST_EX, 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 ST_EX like an *i*-argument, but an application is turned into an **OMBind** in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g. $\backslash\text{symdecl}\{\text{forall}\}[\text{args}=\text{bi}]$ allows for $\backslash\text{forall}\{x\in\text{Nat}\}\{x\geq 0\}$.

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

Chapter 14

STEX-Terms

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

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

Chapter 15

TeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` (*env.*) TODO

Chapter 16

TeX-Statements

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

16.1 Macros and Environments

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

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

Chapter 17

sTeX-Proofs: Structural Markup for Proofs

Chapter 18

sTeX-Metatheory

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput: Treating TIKZ code as images

19.1 Macros and Environments

Chapter 20

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

Chapter 21

NotesSlides – Slides and Course Notes

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

Chapter 23

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

Part IV

Implementation

Chapter 24

\TeX -Basics Implementation

24.1 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}{2022/08/08}{3.2.0}{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>

```

24.2 Preliminaries

```

44 <*package>
45
46 %%%%%%%%% basics.dtx %%%%%%%%%
47
48 \RequirePackage{expl3,l3keys2e,ltxcmds}
49 \ProvidesExplPackage{stex}{2022/08/08}{3.2.0}{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~3.2.0~*^^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 71.)

24.3 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 71.)

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}

```

24.4 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 71.)

`\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 71.)

`\stex_annotate_html:nn`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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.

```

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 72.)

24.5 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 72.)

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

```

24.6 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 }

```

24.7 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 72.)

`\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 72.)

`\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 72.)

\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 25

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 }
```

25.1 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   \stex_debug:nn{files}{#2}
413   \str_set:Nx \l_tmpa_str { #2 }
414   \str_if_empty:NTF \l_tmpa_str {
415     \seq_clear:N #1
416   }{
417     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
418     \sys_if_platform_windows:T{
```

```

419 \seq_clear:N \l_tmpa_tl
420 \seq_map_inline:Nn #1 {
421   \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
422   \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
423 }
424 \seq_set_eq:NN #1 \l_tmpa_tl
425 }
426 \stex_path_canonicalize:N #1
427 }
428 \stex_debug:nn{files}{Yields: \stex_path_to_string:N#1}
429 }
430

```

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

```

\stex_path_to_string:NN
\stex_path_to_string:N
431 \cs_new_protected:Nn \stex_path_to_string:NN {
432   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
433 }
434
435 \cs_new:Nn \stex_path_to_string:N {
436   \seq_use:Nn #1 /
437 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 73.)

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
438 \str_const:Nn \c__stex_path_dot_str {.}
439 \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.

```

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

```

```

460     }
461   }{
462     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
463   }
464 }
465 }{
466   \str_if_empty:NF \l_tmpa_tl {
467     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
468   }
469 }
470 }
471 }
472 \seq_gset_eq:NN #1 \l_tmpa_seq
473 }
474 }

```

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

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:NTF`

```

475 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
476   \seq_if_empty:NTF #1 {
477     \prg_return_false:
478   }{
479     \seq_get_left:NN #1 \l_tmpa_tl
480     \sys_if_platform_windows:TF{
481       \str_if_in:NnTF \l_tmpa_tl {:}{
482         \prg_return_true:
483       }{
484         \prg_return_false:
485       }
486     }{
487       \str_if_empty:NTF \l_tmpa_tl {
488         \prg_return_true:
489       }{
490         \prg_return_false:
491       }
492     }
493   }
494 }

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

495 \str_new:N\l_stex_kpsewhich_return_str
496 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
497   \catcode'\ =12
498   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
499   \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
500   \endgroup
501   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
502   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
503 }

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

504 \sys_if_platform_windows:TF{
505   \begingroup\escapechar=-1\catcode'\=12
506   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
507   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
508   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
509   }}{
510   \stex_kpsewhich:n{-var-value~PWD}
511 }
512
513 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
514 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
515 \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 73.)

25.3 File Hooks and Tracking

516 `<@@=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

```

517 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

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

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 73.)

`\g_stex_currentfile_seq`

```

521 \seq_gclear_new:N\g_stex_currentfile_seq

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 74.)

`\stex_filestack_push:n`

```

522 \cs_new_protected:Nn \stex_filestack_push:n {
523   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
524   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
525     \stex_path_from_string:Nn\g_stex_currentfile_seq{
526       \c_stex_pwd_str/#1
527     }

```

```

528 }
529 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
530 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
531 \stex_get_document_uri:
532 }

```

(End definition for `\stex_filestack_push:n`. This function is documented on page 74.)

`\stex_filestack_pop:`

```

533 \cs_new_protected:Nn \stex_filestack_pop: {
534   \seq_if_empty:NF\g__stex_files_stack{
535     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
536   }
537   \seq_if_empty:NTF\g__stex_files_stack{
538     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
539   }{
540     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
541     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
542   }
543   \stex_get_document_uri:
544 }

```

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

Hooks for the current file:

```

545 \AddToHook{file/before}{
546   \tl_if_empty:NTF\CurrentFilePath{
547     \stex_filestack_push:n{\CurrentFile}
548   }{
549     \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
550   }
551 }
552 \AddToHook{file/after}{
553   \stex_filestack_pop:
554 }

```

25.4 MathHub Repositories

```

555 <@@=stex_mathhub>

```

`\mathhub`
`\c_stex_mathhub_seq`
`\c_stex_mathhub_str`

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

```

556 \str_if_empty:NTF\mathhub{
557   \sys_if_platform_windows:TF{
558     \begingroup\escapechar=-1\catcode'\=12
559     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
560     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
561     \exp_args:NNx\str_if_eq:onT\l_stex_kpsewhich_return_str{\c_percent_str MATHHUB\c_percent_str}
562     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
563   }{
564     \stex_kpsewhich:n{-var-value-MATHHUB}
565   }
566   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
567 }

```



```

568 \str_if_empty:NT \c_stex_mathhub_str {
569   \sys_if_platform_windows:TF{
570     \begingroup\escapechar=-1\catcode'\=12
571     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
572     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
573     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_s
574   }{
575     \stex_kpsewhich:n{-var-value-HOME}
576   }
577   \ior_open:NnT \g_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
578     \begingroup\escapechar=-1\catcode'\=12
579     \ior_str_get:NN \g_tmpa_ior \l_tmpa_str
580     \sys_if_platform_windows:T{
581       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
582     }
583     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
584     \endgroup
585     \ior_close:N \g_tmpa_ior
586   }
587 }
588 \str_if_empty:NTF\c_stex_mathhub_str{
589   \msg_warning:nn{stex}{warning/nomathhub}
590 }{
591   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
592   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
593 }
594 }{
595   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
596   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
597     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
598       \c_stex_pwd_str/\mathhub
599     }
600   }
601   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
602   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
603 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 74.)

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

```

604 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
605   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
606     \str_set:Nx \l_tmpa_str { #1 }
607     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
608     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
609     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
610     \__stex_mathhub_find_manifest:N \l_tmpa_seq
611     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
612       \msg_error:nnxx{stex}{error/norepository}{#1}{
613         \stex_path_to_string:N \c_stex_mathhub_str
614       }
615       \input{Fatal-Error!}

```

```

616     } {
617     \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
618     }
619   }
620 }

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

621 \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:

```

622 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
623   \seq_set_eq:NN\l_tmpa_seq #1
624   \bool_set_true:N\l_tmpa_bool
625   \bool_while_do:Nn \l_tmpa_bool {
626     \seq_if_empty:NTF \l_tmpa_seq {
627       \bool_set_false:N\l_tmpa_bool
628     }{
629       \file_if_exist:nTF{
630         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
631       }{
632         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
633         \bool_set_false:N\l_tmpa_bool
634       }{
635         \file_if_exist:nTF{
636           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
637         }{
638           \seq_put_right:Nn\l_tmpa_seq{META-INF}
639           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
640           \bool_set_false:N\l_tmpa_bool
641         }{
642           \file_if_exist:nTF{
643             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
644           }{
645             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
646             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
647             \bool_set_false:N\l_tmpa_bool
648           }{
649             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
650           }
651         }
652       }
653     }
654   }
655   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
656 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior

File variable used for MANIFEST-files

```

657 \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:

```

658 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
659   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
660   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
661   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
662     \str_set:Nn \l_tmpa_str {##1}
663     \exp_args:NNo \seq_set_split:Nnn
664       \l_tmpb_seq \c_colon_str \l_tmpa_str
665     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
666       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
667         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
668       }
669       \exp_args:No \str_case:nnTF \l_tmpa_tl {
670         {id} {
671           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
672             { id } \l_tmpb_tl
673         }
674         {narration-base} {
675           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
676             { narr } \l_tmpb_tl
677         }
678         {url-base} {
679           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
680             { docurl } \l_tmpb_tl
681         }
682         {source-base} {
683           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
684             { ns } \l_tmpb_tl
685         }
686         {ns} {
687           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
688             { ns } \l_tmpb_tl
689         }
690         {dependencies} {
691           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
692             { deps } \l_tmpb_tl
693         }
694       }{}{}
695     }{}
696   }
697   \ior_close:N \c__stex_mathhub_manifest_ior
698   \stex_persist:x {
699     \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
700       \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
701     }
702   }
703 }
```

(End definition for `_stex_mathhub_parse_manifest:n.`)

`\stex_set_current_repository:n`

```

704 \cs_new_protected:Nn \stex_set_current_repository:n {
```

```

705 \stex_require_repository:n { #1 }
706 \prop_set_eq:Nc \l_stex_current_repository_prop {
707   c_stex_mathhub_#1_manifest_prop
708 }
709 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 74.)

`\stex_require_repository:n`

```

710 \cs_new_protected:Nn \stex_require_repository:n {
711   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
712     \stex_debug:nn{mathhub}{Opening~archive:~#1}
713     \__stex_mathhub_do_manifest:n { #1 }
714   }
715 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 74.)

`\l_stex_current_repository_prop` Current MathHub repository

```

716 %\prop_new:N \l_stex_current_repository_prop
717 \bool_if:NF \c_stex_persist_mode_bool {
718   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
719   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
720     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
721   } {
722     \__stex_mathhub_parse_manifest:n { main }
723     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
724     \l_tmpa_str
725     \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
726     \c_stex_mathhub_main_manifest_prop
727     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
728     \stex_debug:nn{mathhub}{Current~repository:~
729     \prop_item:Nn \l_stex_current_repository_prop {id}
730   }
731 }
732 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 74.)

`\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.

```

733 \cs_new_protected:Nn \stex_in_repository:nn {
734   \str_set:Nx \l_tmpa_str { #1 }
735   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
736   \str_if_empty:NTF \l_tmpa_str {
737     \prop_if_exist:NTF \l_stex_current_repository_prop {
738       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
739       \exp_args:Ne \l_tmpa_cs{
740         \prop_item:Nn \l_stex_current_repository_prop { id }
741       }
742     }{
743       \l_tmpa_cs{}
744     }
745   }{

```

```

746 \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
747 \stex_require_repository:n \l_tmpa_str
748 \str_set:Nx \l_tmpa_str { #1 }
749 \exp_args:Nne \use:nn {
750   \stex_set_current_repository:n \l_tmpa_str
751   \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
752 }{
753   \stex_debug:nn{mathhub}{switching~back~to:~
754     \prop_if_exist:NTF \l_stex_current_repository_prop {
755       \prop_item:Nn \l_stex_current_repository_prop { id }::~
756       \meaning\l_stex_current_repository_prop
757     }{
758       no~repository
759     }
760   }
761   \prop_if_exist:NTF \l_stex_current_repository_prop {
762     \stex_set_current_repository:n {
763       \prop_item:Nn \l_stex_current_repository_prop { id }
764     }
765   }{
766     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
767   }
768 }
769 }
770 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 74.)

25.5 Using Content in Archives

`\mhpath`

```

771 \def \mhpath #1 #2 {
772   \exp_args:Ne \tl_if_empty:nTF{#1}{
773     \c_stex_mathhub_str /
774     \prop_item:Nn \l_stex_current_repository_prop { id }
775     / source / #2
776   }{
777     \c_stex_mathhub_str / #1 / source / #2
778   }
779 }

```

(End definition for `\mhpath`. This function is documented on page 75.)

`\inputref`

`\mhinput`

```

780 \newif \ifinputref \inputreffalse
781
782 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
783   \stex_in_repository:nn {#1} {
784     \ifinputref
785       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
786     \else
787       \inputreftrue
788       \input{ \c_stex_mathhub_str / ##1 / source / #2 }

```

```

789     \inputreffalse
790   \fi
791 }
792 }
793 \NewDocumentCommand \mhinput { 0{} m}{
794   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
795 }
796
797 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
798   \stex_in_repository:nn {#1} {
799     \stex_html_backend:TF {
800       \str_clear:N \l_tmpa_str
801       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
802         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
803       }
804
805       \tl_if_empty:nTF{ ##1 }{
806         \IfFileExists{#2}{
807           \stex_annotate_invisible:nnn{inputref}{
808             \l_tmpa_str / #2
809           }{}
810         }{
811           \input{#2}
812         }
813       }{
814         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
815           \stex_annotate_invisible:nnn{inputref}{
816             \l_tmpa_str / #2
817           }{}
818         }{
819           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
820         }
821       }
822
823     }{
824       \begingroup
825       \inputreftrue
826       \tl_if_empty:nTF{ ##1 }{
827         \input{#2}
828       }{
829         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
830       }
831     } \endgroup
832   }
833 }
834 }
835 \NewDocumentCommand \inputref { 0{} m}{
836   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
837 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 75.)

`\addmhbibresource`

```

838 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {

```

```

839 \stex_in_repository:nn {#1} {
840   \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
841 }
842 }
843 \newcommand\addmhbibresource[2] []{
844   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
845 }

```

(End definition for \addmhbibresource. This function is documented on page 75.)

\libinput

```

846 \cs_new_protected:Npn \libinput #1 {
847   \prop_if_exist:NF \l_stex_current_repository_prop {
848     \msg_error:nnn{stex}{error/notinarchive}\libinput
849   }
850   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
851     \msg_error:nnn{stex}{error/notinarchive}\libinput
852   }
853   \seq_clear:N \l__stex_mathhub_libinput_files_seq
854   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
855   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
856
857   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
858     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
859     \IfFileExists{ \l_tmpa_str }{
860       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
861     }{}
862     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
863     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
864   }
865
866   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
867   \IfFileExists{ \l_tmpa_str }{
868     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
869   }{}
870
871   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
872     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
873   }{
874     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
875       \input{ ##1 }
876     }
877   }
878 }

```

(End definition for \libinput. This function is documented on page 75.)

\libusepackage

```

879 \NewDocumentCommand \libusepackage {0{ } m} {
880   \prop_if_exist:NF \l_stex_current_repository_prop {
881     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
882   }
883   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
884     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
885   }

```

```

886 \seq_clear:N \l__stex_mathhub_libinput_files_seq
887 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
888 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
889
890 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
891   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
892   \IfFileExists{ \l_tmpa_str.sty }{
893     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
894   }{
895     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
896     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
897   }
898
899 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
900 \IfFileExists{ \l_tmpa_str.sty }{
901   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
902 }{
903
904 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
905   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
906 }{
907   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
908     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
909       \usepackage[#1]{ #1 }
910     }
911   }{
912     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
913   }
914 }
915 }

```

(End definition for `\libusepackage`. This function is documented on page 75.)

`\mhgraphics`
`\cmhgraphics`

```

916
917 \AddToHook{begindocument}{
918 \ltx@ifpackageloaded{graphicx}{
919   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
920   \providecommand\mhgraphics[2] [] {%
921     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
922     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
923   \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
924 }{

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 75.)

`\lstinputmhlisting`
`\clstinputmhlisting`

```

925 \ltx@ifpackageloaded{listings}{
926   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
927   \newcommand\lstinputmhlisting[2] [] {%
928     \def\lst@mhrepos{}\setkeys{lst}{#1}%
929     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
930   \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
931 }{
932 }

```


933

934 `\end{package}`

(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 75.)

Chapter 26

STEX -References Implementation

```
935 <*package>
936
937 %%%%%%%%% stex-references.dtx %%%%%%%%%
938
939 <@@=stex_refs>
940
941   Warnings and error messages
942   \msg_new:nnn{stex}{error/extrefmissing}{
943     Missing~in~or~cite~value~for~\detokenize{\extref}!
944   }
945   \msg_new:nnn{stex}{warning/smsmissing}{
946     .sref~file~#1~doesn't~exist!
947   }
948   \msg_new:nnn{stex}{warning/smslabelmissing}{
949     No~label~#2~in~.sref~file~#1!
950   }
```

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

```
949 \iow_new:N \c__stex_refs_refs_iow
950 \AtBeginDocument{
951   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
952 }
953 \AtEndDocument{
954   \iow_close:N \c__stex_refs_refs_iow
955 }
```

`\STEXreftitle`

```
956 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
957
958 \NewDocumentCommand \STEXreftitle { m } {
959   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
960   \iow_now:Nx \c__stex_refs_refs_iow {
961     \STEXInternalSrefDocumentName { #1 }
962   }
963 }
```

(End definition for `\STEXrefitle`. This function is documented on page 76.)

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
964 \str_new:N \l_stex_current_docns_str
```

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

`\stex_get_document_uri:`

```
965 \cs_new_protected:Nn \stex_get_document_uri: {
966   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
967   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
968   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
969   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
970   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
971
972   \str_clear:N \l_tmpa_str
973   \prop_if_exist:NT \l_stex_current_repository_prop {
974     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
975       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
976     }
977   }
978
979   \str_if_empty:NTF \l_tmpa_str {
980     \str_set:Nx \l_stex_current_docns_str {
981       file:/\stex_path_to_string:N \l_tmpa_seq
982     }
983   }{
984     \bool_set_true:N \l_tmpa_bool
985     \bool_while_do:Nn \l_tmpa_bool {
986       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
987       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
988         {source} { \bool_set_false:N \l_tmpa_bool }
989       }{}{
990         \seq_if_empty:NT \l_tmpa_seq {
991           \bool_set_false:N \l_tmpa_bool
992         }
993       }
994     }
995
996     \seq_if_empty:NTF \l_tmpa_seq {
997       \str_gset_eq:NN \l_stex_current_docns_str \l_tmpa_str
998     }{
999       \str_gset:Nx \l_stex_current_docns_str {
1000         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1001       }
1002     }
1003   }
1004   %\stex_get_document_uri:
1005 }
```

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

`\l_stex_current_docurl_str`

```
1006 \str_new:N \l_stex_current_docurl_str
```

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

`\stex_get_document_url:`

```
1007 \cs_new_protected:Nn \stex_get_document_url: {
1008   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1009   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1010   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1011   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1012   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1013
1014   \str_clear:N \l_tmpa_str
1015   \prop_if_exist:NT \l_stex_current_repository_prop {
1016     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
1017       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
1018         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
1019       }
1020     }
1021   }
1022
1023   \str_if_empty:NTF \l_tmpa_str {
1024     \str_set:Nx \l_stex_current_docurl_str {
1025       file:/\stex_path_to_string:N \l_tmpa_seq
1026     }
1027   }{
1028     \bool_set_true:N \l_tmpa_bool
1029     \bool_while_do:Nn \l_tmpa_bool {
1030       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1031       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1032         {source} { \bool_set_false:N \l_tmpa_bool }
1033       }{}{
1034         \seq_if_empty:NT \l_tmpa_seq {
1035           \bool_set_false:N \l_tmpa_bool
1036         }
1037       }
1038     }
1039
1040     \seq_if_empty:NTF \l_tmpa_seq {
1041       \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1042     }{
1043       \str_set:Nx \l_stex_current_docurl_str {
1044         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1045       }
1046     }
1047   }
1048 }
```

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

26.2 Setting Reference Targets

```

1049 \str_const:Nn \c__stex_refs_url_str{URL}
1050 \str_const:Nn \c__stex_refs_ref_str{REF}
1051 \str_new:N \l__stex_refs_curr_label_str
1052 % @currentlabel -> number
1053 % @currentlabelname -> title
1054 % @currentHref -> name.number <- id of some kind
1055 % @currentcounter <- name/id
1056 % \#autorefname <- "Section"
1057 % \theH# -> \arabic{section}
1058 % \the# -> number
1059 % \hyper@makecurrent{#}
1060 \int_new:N \l__stex_refs_unnamed_counter_int

```

Restoring references from .sref-files

\STEXInternalSrefRestoreTarget

```

1061 \cs_new_protected:Npn \STEXInternalSrefDocumentName #1 {}
1062 \cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}

```

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

\stex_ref_new_doc_target:n

```

1063 \seq_new:N \g_stex_ref_files_seq
1064
1065 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1066   %\stex_get_document_uri:
1067   \str_clear:N \l__stex_refs_curr_label_str
1068   \str_set:Nx \l_tmpa_str { #1 }
1069   \str_if_empty:NT \l_tmpa_str {
1070     \int_gincr:N \l__stex_refs_unnamed_counter_int
1071     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1072   }
1073   \str_set:Nx \l__stex_refs_curr_label_str {
1074     \l_stex_current_docns_str?\l_tmpa_str
1075   }
1076
1077   \exp_args:Noo \STEXInternalAuxAddDocRef\l_stex_current_docns_str\l_tmpa_str
1078
1079   %\seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1080   %   \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1081   %}
1082   %\seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1083   %   \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1084   %}
1085
1086
1087   \stex_if_smsmode:TF {
1088     %\stex_get_document_url:
1089     %\str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1090     %\str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1091   }{
1092     \iow_now:Nx \c__stex_refs_refs_iow {
1093       \STEXInternalSrefRestoreTarget
1094       {\l_stex_current_docns_str}
1095       {\l_tmpa_str}

```

```

1096         {\@currentcounter}
1097         {\@currentlabel}
1098         {\tl_if_exist:NT\@currentlabelname{\exp_args:No\unexpanded\@currentlabelname}}
1099     }
1100     %\iow_now:Nx \c__stex_refs_refs_iow {
1101     %   {\l_stex_current_docns_str?\l_tmpa_str}~=={\use:c{\@currentcounter autorefname}~\@currentcounter}
1102     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1103     \immediate\write\@auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_str}{\l_stex_refs_curr_label_str}}
1104     %\str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1105 }
1106 }
1107 \NewDocumentCommand \slabel {m} {\stex_ref_new_doc_target:n {#1}}

```

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

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

```

1108 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1109   \exp_args:NNx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1110     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1111       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1112     }
1113   }{
1114     \exp_args:NNx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1115     %\seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1116       \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1117     }
1118     \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1119   }
1120
1121   %\str_set:Nn \l_tmpa_str {#1?#2}
1122   %\str_gset_eq:cn{sref_#1?#2_type}\c__stex_refs_ref_str
1123   %\seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1124     % \seq_new:c {g__stex_refs_labels_#2_seq}
1125   }
1126   %\seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1127     % \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1128   }
1129 }

```

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

```

1130 \AtEndDocument{
1131   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1132 }

```

`\stex_ref_new_sym_target:n`

```

1133 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1134
1135   % \stex_if_smsmode:TF {
1136   %   \str_if_exist:cF{sref_sym_#1_type}{
1137   %     \stex_get_document_url:
1138   %     \str_gset_eq:cn {sref_sym_url_#1_str}\l_stex_current_docurl_str
1139   %     \str_gset_eq:cn {sref_sym_#1_type}\c__stex_refs_url_str
1140   %   }
1141   % }{
1142   %   \str_if_empty:NF \l__stex_refs_curr_label_str {

```

```

1143 %      \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1144 %      \immediate\write\@auxout{
1145 %          \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label
1146 %              \l__stex_refs_curr_label_str
1147 %          }
1148 %      }
1149 %  }
1150 % }
1151 }

```

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

26.3 Using References

`\sref` Optional arguments:

```

1152
1153 \keys_define:nn { stex / sref / 1 } {
1154   archive .str_set_x:N = \l__stex_refs_repo_str,
1155   file    .str_set_x:N = \l__stex_refs_file_str
1156 }
1157 \cs_new_protected:Nn \__stex_refs_args_i:n {
1158   \str_clear:N \l__stex_refs_repo_str
1159   \str_clear:N \l__stex_refs_file_str
1160   \keys_set:nn { stex / sref / 1 } { #1 }
1161 }
1162 \keys_define:nn { stex / sref / 2 } {
1163   in .str_set_x:N = \l__stex_refs_in_str,
1164   archive .str_set_x:N = \l__stex_refs_repob_str,
1165   cite .str_set_x:N = \l__stex_refs_cite_str
1166 }
1167 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1168   \str_clear:N \l__stex_refs_in_str
1169   \str_clear:N \l__stex_refs_cite_str
1170   \str_clear:N \l__stex_refs_repob_str
1171   \keys_set:nn { stex / sref / 2 } { #1 }
1172 }

```

The actual macro:

```

1173 \NewDocumentCommand \sref { 0{} m 0{} }{
1174   \__stex_refs_args_i:n{#1}
1175   \__stex_refs_args_ii:n{#3}
1176   \str_clear:N \l__stex_refs_uri_str
1177   \__stex_refs_find_uri:n{#2}
1178   \__stex_refs_do_sref:n{#2}
1179 }
1180 \NewDocumentCommand \extref { 0{} m m }{
1181   \__stex_refs_args_i:n{#1}
1182   \__stex_refs_args_ii:n{#3}
1183   \str_if_empty:NT \l__stex_refs_in_str {
1184     \msg_error:nn{stex}{error/extrefmissing}
1185   }
1186   \str_clear:N \l__stex_refs_uri_str
1187   \__stex_refs_find_uri:n{#2}
1188   \__stex_refs_do_sref_in:n{#2}

```

```

1189 }
1190
1191 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1192   \stex_debug:nn{sref}{File: \l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1193   \str_if_empty:NTF \l__stex_refs_file_str {
1194     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str_seq}{
1195       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str_seq}{
1196         \str_if_eq:nnT{#1}{##1}{
1197           \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1198           \seq_map_break:
1199         }
1200       }
1201     }
1202     \str_if_empty:NF \l__stex_refs_uri_str {
1203       \seq_map_inline:Nn \g_stex_ref_files_seq {
1204         \seq_map_inline:cn{g_stex_ref_##1_seq}{
1205           \str_if_eq:nnT{#1}{####1}{
1206             \str_set:Nn \l__stex_refs_uri_str {##1}
1207             \seq_map_break:n{\seq_map_break:}
1208           }
1209         }
1210       }
1211     }
1212   }{
1213     \str_if_empty:NTF \l__stex_refs_repo_str {
1214       \prop_if_exist:NTF \l_stex_current_repository_prop {
1215         \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1216         \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1217         \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1218         \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1219       }{
1220         \stex_path_from_string:Nn \l_tmpb_seq {
1221           \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1222         }
1223         \str_set:Nx \l__stex_refs_uri_str {file:\stex_path_to_string:N \l_tmpb_seq}
1224       }
1225     }{
1226       \stex_require_repository:n \l__stex_refs_repo_str
1227       \prop_get:cnN { c_stex_mathhub_\l__stex_refs_repo_str_manifest_prop } { ns } \l__stex_refs_uri_str
1228       \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1229       \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1230       \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1231     }
1232   }
1233 }
1234
1235 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1236   \cs_if_exist:cTF{autoref}{
1237     \exp_args:Nx\autoref{sref_#1}
1238   }{
1239     \exp_args:Nx\ref{sref_#1}
1240   }
1241 }
1242

```



```

1243 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1244   \str_if_empty:NTF \l__stex_refs_uri_str {
1245     \str_if_empty:NTF \l__stex_refs_in_str {
1246       \__stex_refs_do_autoref:n{#1}
1247     }{
1248       \__stex_refs_do_sref_in:n{#1}
1249     }
1250   }{
1251     \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1252       \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref\l__stex_refs_uri_str_seq}{\detokenize{#1}}{
1253         \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1254       }{
1255         \str_if_empty:NTF \l__stex_refs_in_str {
1256           \__stex_refs_do_autoref:n{#1}
1257         }{
1258           \__stex_refs_do_sref_in:n{#1}
1259         }
1260       }
1261     }{
1262       \str_if_empty:NTF \l__stex_refs_in_str {
1263         \__stex_refs_do_autoref:n{#1}
1264       }{
1265         \__stex_refs_do_sref_in:n{#1}
1266       }
1267     }
1268   }
1269 }
1270
1271 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1272   \str_if_empty:NTF \l__stex_refs_uri_str {
1273     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1274       \tl_set:Nn \l__stex_refs_return_tl {
1275         \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(#5)}~in~
1276         \str_if_empty:NTF \l__stex_refs_cite_str{\tl_if_empty:nTF\l__stex_refs_docname_tl{
1277           ???
1278         }\l__stex_refs_docname_tl}{\cite{\l__stex_refs_cite_str}}
1279       }
1280     }
1281   }{
1282     \stex_debug:nn{sref}{\l__stex_refs_uri_str}{~ == ~ #1 ~ ?}
1283     \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1284       \stex_debug:nn{sref}{\l__stex_refs_id_str}{~ == ~ #2 ~ ?}
1285       \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1286         \stex_debug:nn{sref}{success!}
1287         \tl_set:Nn \l__stex_refs_return_tl {
1288           \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(#5)}~in~
1289           \str_if_empty:NTF \l__stex_refs_cite_str{\tl_if_empty:nTF\l__stex_refs_docname_tl{
1290             ???
1291           }\l__stex_refs_docname_tl}{\cite{\l__stex_refs_cite_str}}
1292         }
1293       }
1294     }
1295   }
1296 }

```

```

1297 }
1298 \cs_new_protected:Nn \__stex_refs_document_name:n {
1299   \tl_set:Nn \l__stex_refs_docname_tl {#1}
1300 }
1301 % TODO cite
1302 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1303   \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1304   \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1305   %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1306   \begingroup\catcode13=9\relax\catcode10=9\relax
1307     \str_if_empty:NTF \l__stex_refs_repob_str {
1308       \prop_if_exist:NTF \l_stex_current_repository_prop {
1309         \str_set:Nx \l_tmpa_str {
1310           \c_stex_mathhub_str /
1311           \prop_item:Nn \l_stex_current_repository_prop { id }
1312           / source / \l__stex_refs_in_str .sref
1313         }
1314       }{
1315         \str_set:Nx \l_tmpa_str {
1316           \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str .sref
1317         }
1318       }
1319     }{
1320       \str_set:Nx \l_tmpa_str {
1321         \c_stex_mathhub_str / \l__stex_refs_repob_str
1322         / source / \l__stex_refs_in_str .sref
1323       }
1324     }
1325   \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1326   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1327   \stex_debug:nn{sref}{File: \l_tmpa_str}
1328   \exp_args:No \IfFileExists \l_tmpa_str {
1329     \tl_clear:N \l__stex_refs_docname_tl
1330     \tl_clear:N \l__stex_refs_return_tl
1331     \str_set:Nn \l__stex_refs_id_str {#1}
1332     \let\STEXInternalSrefDocumentName\__stex_refs_document_name:n
1333     \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1334     \use:c{@ @ input}{\l_tmpa_str}
1335     \exp_args:No \tl_if_empty:NTF \l__stex_refs_return_tl {
1336       \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1337       \__stex_refs_do_autoref:n{
1338         \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1339       }
1340     }{
1341       \l__stex_refs_return_tl
1342     }
1343   }{
1344     \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l_tmpa_str
1345     \__stex_refs_do_autoref:n{
1346       \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1347     }
1348   }
1349   \endgroup
1350 }

```

```

1351 % \__stex_refs_args:n { #1 }
1352 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1353 %   \str_set:Nx \l_tmpa_str { #2 }
1354 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1355 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1356 %     \seq_if_exist:CTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1357 %       \seq_get_left:CNF {g__stex_refs_labels_\l_tmpa_str_seq} \l_tmpa_str {
1358 %         \str_clear:N \l_tmpa_str
1359 %       }
1360 %     }{
1361 %       \str_clear:N \l_tmpa_str
1362 %     }
1363 %   }{
1364 %     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1365 %     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1366 %     \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1367 %     \seq_if_exist:CTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1368 %       \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1369 %       \str_clear:N \l_tmpa_str
1370 %       \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1371 %         \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1372 %           \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1373 %         }{
1374 %           \seq_map_break:n {
1375 %             \str_set:Nn \l_tmpa_str { ##1 }
1376 %           }
1377 %         }
1378 %       }
1379 %     }{
1380 %       \str_clear:N \l_tmpa_str
1381 %     }
1382 %   }
1383 %   \str_if_empty:NTF \l_tmpa_str {
1384 %     \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_li
1385 %   }{
1386 %     \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1387 %       \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1388 %         \cs_if_exist:CTF{autoref}{
1389 %           \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1390 %         }{
1391 %           \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1392 %         }
1393 %       }
1394 %     }{
1395 %       \ltx@ifpackageloaded{hyperref}{
1396 %         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1397 %       }{
1398 %         \l__stex_refs_linktext_tl
1399 %       }
1400 %     }
1401 %   }{
1402 %     \ltx@ifpackageloaded{hyperref}{
1403 %       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_
1404 %     }{

```

```

1405 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_fallback_tl
1406 % }
1407 % }
1408 % }
1409 % }{
1410 % TODO
1411 % }
1412 %}

```

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

`\srefsym`

```

1413 \NewDocumentCommand \srefsym { 0{} m}{
1414   \stex_get_symbol:n { #2 }
1415   \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1416 }
1417
1418 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1419
1420 % \str_if_exist:cTF {sref_sym_#2 _label_str }{
1421 %   \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1422 % }{
1423 %   \__stex_refs_args:n { #1 }
1424 %   \str_if_empty:NTF \l__stex_refs_indocument_str {
1425 %     \tl_if_exist:cTF{sref_sym_#2 _type}{
1426 %       % doc uri in \l_tmpb_str
1427 %       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1428 %       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1429 %         % reference
1430 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1431 %           \cs_if_exist:cTF{autoref}{
1432 %             \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1433 %           }{
1434 %             \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1435 %           }
1436 %         }{
1437 %           \ltx@ifpackageloaded{hyperref}{
1438 %             \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1439 %           }{
1440 %             \l__stex_refs_linktext_tl
1441 %           }
1442 %         }
1443 %       }{
1444 %         % URL
1445 %         \ltx@ifpackageloaded{hyperref}{
1446 %           \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1447 %             }{
1448 %               \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_fallback_tl
1449 %             }
1450 %         }
1451 %       }{
1452 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_fallback_tl
1453 %       }
1454 %     }{

```

```

1455 %      % TODO
1456 %    }
1457 %  }
1458 }

```

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

\srefsymuri

```

1459 \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1460   #2%\_stex_refs_sym_aux:nn{linktext={#2}}{#1}
1461 }

```

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

```

1462 \</package>

```

Chapter 27

STEX -Modules Implementation

```
1463 <*package>
1464
1465 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1466
1467 <@@=stex_modules>
1468
1469 Warnings and error messages
1468 \msg_new:nnn{stex}{error/unknownmodule}{
1469   No~module~#1~found
1470 }
1471 \msg_new:nnn{stex}{error/syntax}{
1472   Syntax~error:~#1
1473 }
1474 \msg_new:nnn{stex}{error/siglanguage}{
1475   Module~#1~declares~signature~#2,~but~does~not~
1476   declare~its~language
1477 }
1478 \msg_new:nnn{stex}{warning/deprecated}{
1479   #1~is~deprecated;~please~use~#2~instead!
1480 }
1481
1482 \msg_new:nnn{stex}{error/conflictingmodules}{
1483   Conflicting~imports~for~module~#1
1484 }
```

`\l_stex_current_module_str` The current module:

```
1485 \str_new:N \l_stex_current_module_str
```

(End definition for `\l_stex_current_module_str`. This variable is documented on page 79.)

`\l_stex_all_modules_seq` Stores all available modules

```
1486 \seq_new:N \l_stex_all_modules_seq
```

(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 79.)

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1487 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1488   \str_if_empty:NTF \l_stex_current_module_str
1489   \prg_return_false: \prg_return_true:
1490 }

```

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

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1491 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1492   \prop_if_exist:cTF { c_stex_module_#1_prop }
1493   \prg_return_true: \prg_return_false:
1494 }

```

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

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1495 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1496   \stex_add_to_current_module:n { #1 }
1497   \stex_do_up_to_module:n { #1 }
1498 }}
1499 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1500
1501 \cs_new_protected:Nn \stex_add_to_current_module:n {
1502   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1503 }
1504 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1505 \cs_new_protected:Npn \STEXexport {
1506   \ExplSyntaxOn
1507   \__stex_modules_export:n
1508 }
1509 \cs_new_protected:Nn \__stex_modules_export:n {
1510   \ignorespacesandpars#1\ExplSyntaxOff
1511   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1512   \stex_smsmode_do:
1513 }
1514 \let \stex_module_export_helper:n \use:n
1515 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 79.)

```

\stex_add_constant_to_current_module:n
1516 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1517   \str_set:Nx \l_tmpa_str { #1 }
1518   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1519 }

```

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

```

\stex_add_import_to_current_module:n
1520 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1521   \str_set:Nx \l_tmpa_str { #1 }
1522   \exp_args:Nno

```

```

1523 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1524 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1525 }
1526 }

```

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

`\stex_collect_imports:n`

```

1527 \cs_new_protected:Nn \stex_collect_imports:n {
1528 \seq_clear:N \l_stex_collect_imports_seq
1529 \__stex_modules_collect_imports:n {#1}
1530 }
1531 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1532 \seq_map_inline:cn {c_stex_module_#1_imports} {
1533 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1534 \__stex_modules_collect_imports:n { ##1 }
1535 }
1536 }
1537 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1538 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1539 }
1540 }

```

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

`\stex_do_up_to_module:n`

```

1541 \int_new:N \l__stex_modules_group_depth_int
1542 \cs_new_protected:Nn \stex_do_up_to_module:n {
1543 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1544 #1
1545 }{
1546 #1
1547 \expandafter \tl_gset:Nn
1548 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1549 \expandafter\expandafter\expandafter\endcsname
1550 \expandafter\expandafter\expandafter { \csname
1551 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1552 \aftergroup\__stex_modules_aftergroup_do:
1553 }
1554 }
1555 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1556 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1557 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1558 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1559 }}}
1560 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1561 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1562 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1563 }{
1564 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1565 \aftergroup\__stex_modules_aftergroup_do:
1566 }
1567 }
1568 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1569 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```


1570 }

(End definition for \stex_do_up_to_module:n. This function is documented on page 79.)

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

1571

(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.

```
1572 \str_new:N \l_stex_module_ns_str
1573 \str_new:N \l_stex_module_subpath_str
1574 \cs_new_protected:Nn __stex_modules_compute_namespace:nN {
1575   \seq_set_eq:NN \l_tmpa_seq #2
1576   % split off file extension
1577   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1578   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1579   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1580   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1581
1582   \bool_set_true:N \l_tmpa_bool
1583   \bool_while_do:Nn \l_tmpa_bool {
1584     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1585     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1586       {source} { \bool_set_false:N \l_tmpa_bool }
1587     }{}{
1588       \seq_if_empty:NT \l_tmpa_seq {
1589         \bool_set_false:N \l_tmpa_bool
1590       }
1591     }
1592   }
1593
1594   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1595   % \l_tmpa_seq <- sub-path relative to archive
1596   \str_if_empty:NTF \l_stex_module_subpath_str {
1597     \str_set:Nx \l_stex_module_ns_str {#1}
1598   }{
1599     \str_set:Nx \l_stex_module_ns_str {
1600       #1/\l_stex_module_subpath_str
1601     }
1602   }
1603 }
1604
1605 \cs_new_protected:Nn \stex_modules_current_namespace: {
1606   \str_clear:N \l_stex_module_subpath_str
1607   \prop_if_exist:NTF \l_stex_current_repository_prop {
1608     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1609     __stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1610   }{
1611     % split off file extension
1612     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1613     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

1614 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1615 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1616 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1617 \str_set:Nx \l_stex_module_ns_str {
1618   file:/\stex_path_to_string:N \l_tmpa_seq
1619 }
1620 }
1621 }

```

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

27.1 The smodule environment

smodule arguments:

```

1622 \keys_define:nn { stex / module } {
1623   title      .tl_set:N      = \smodulename ,
1624   type       .str_set_x:N   = \smodulename ,
1625   id         .str_set_x:N   = \smoduleid ,
1626   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1627   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1628   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1629   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1630   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1631   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1632   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1633   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1634 }
1635
1636 \cs_new_protected:Nn \__stex_modules_args:n {
1637   \str_clear:N \smodulename
1638   \str_clear:N \smodulename
1639   \str_clear:N \smoduleid
1640   \str_clear:N \l_stex_module_ns_str
1641   \str_clear:N \l_stex_module_deprecate_str
1642   \str_clear:N \l_stex_module_lang_str
1643   \str_clear:N \l_stex_module_sig_str
1644   \str_clear:N \l_stex_module_creators_str
1645   \str_clear:N \l_stex_module_contributors_str
1646   \str_clear:N \l_stex_module_meta_str
1647   \str_clear:N \l_stex_module_srccite_str
1648   \keys_set:nn { stex / module } { #1 }
1649 }
1650
1651 % module parameters here? In the body?
1652

```

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

```

1653 \cs_new_protected:Nn \stex_module_setup:nn {
1654   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1655   \str_set:Nx \l_stex_module_name_str { #2 }
1656   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1657 \stex_if_in_module:TF {
1658   % Nested module
1659   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1660   { ns } \l_stex_module_ns_str
1661   \str_set:Nx \l_stex_module_name_str {
1662     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1663     { name } / \l_stex_module_name_str
1664   }
1665   \str_if_empty:NT \l_stex_module_lang_str {
1666     \str_set:Nx \l_stex_module_lang_str {
1667       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1668       { lang }
1669     }
1670   }
1671 }{
1672   % not nested:
1673   \str_if_empty:NT \l_stex_module_ns_str {
1674     \stex_modules_current_namespace:
1675     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1676       / {\l_stex_module_ns_str}
1677     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1678     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1679       \str_set:Nx \l_stex_module_ns_str {
1680         \stex_path_to_string:N \l_tmpa_seq
1681       }
1682     }
1683   }
1684 }

```

Next, we determine the language of the module:

```

1685 \str_if_empty:NT \l_stex_module_lang_str {
1686   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1687   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1688   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1689   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1690     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1691       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1692     }
1693   }
1694   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1695   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1696     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1697     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1698       inferred~from~file~name}
1699   }
1700 }
1701
1702 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1703   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1704 }}

```

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

```

1705 \str_if_empty:NTF \l_stex_module_sig_str {
1706   \exp_args:Nnx \prop_gset_from_keyval:cn {
1707     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1708   } {
1709     name      = \l_stex_module_name_str ,
1710     ns        = \l_stex_module_ns_str ,
1711     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1712     lang      = \l_stex_module_lang_str ,
1713     sig       = \l_stex_module_sig_str ,
1714     deprecate = \l_stex_module_deprecate_str ,
1715     meta      = \l_stex_module_meta_str
1716   }
1717   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1718   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1719   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1720   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1721   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1722 \str_if_empty:NT \l_stex_module_meta_str {
1723   \str_set:Nx \l_stex_module_meta_str {
1724     \c_stex_metatheory_ns_str ? Metatheory
1725   }
1726 }
1727 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1728   \bool_set_true:N \l_stex_in_meta_bool
1729   \exp_args:Nx \stex_add_to_current_module:n {
1730     \bool_set_true:N \l_stex_in_meta_bool
1731     \stex_activate_module:n {\l_stex_module_meta_str}
1732     \bool_set_false:N \l_stex_in_meta_bool
1733   }
1734   \stex_activate_module:n {\l_stex_module_meta_str}
1735   \bool_set_false:N \l_stex_in_meta_bool
1736 }
1737 }{
1738   \str_if_empty:NT \l_stex_module_lang_str {
1739     \msg_error:nnxx{stex}{error/siglanguage}{
1740       \l_stex_module_ns_str?\l_stex_module_name_str
1741     }\{\l_stex_module_sig_str}
1742   }
1743   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1744   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1745     \stex_debug:nn{modules}{(already exists)}
1746   }{
1747     \stex_debug:nn{modules}{(needs loading)}
1748     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1749     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1750     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1751     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1752     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1753     \str_set:Nx \l_tmpa_str {
1754       \stex_path_to_string:N \l_tmpa_seq /

```

```

1755     \l_tmpa_str . \l_stex_module_sig_str .tex
1756   }
1757   \IfFileExists \l_tmpa_str {
1758     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1759       \str_clear:N \l_stex_current_module_str
1760       \seq_clear:N \l_stex_all_modules_seq
1761       \stex_debug:nn{modules}{Loading~signature}
1762     }
1763   }{
1764     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1765   }
1766 }
1767 \stex_if_smsmode:F {
1768   \stex_activate_module:n {
1769     \l_stex_module_ns_str ? \l_stex_module_name_str
1770   }
1771 }
1772 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1773 }
1774 \str_if_empty:NF \l_stex_module_deprecate_str {
1775   \msg_warning:nnxx{stex}{warning/deprecated}{
1776     Module~\l_stex_current_module_str
1777   }{
1778     \l_stex_module_deprecate_str
1779   }
1780 }
1781 \seq_put_right:Nx \l_stex_all_modules_seq {
1782   \l_stex_module_ns_str ? \l_stex_module_name_str
1783 }
1784 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1785 }

```

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

smodule (*env.*) The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}
1786 \cs_new_protected:Nn \__stex_modules_begin_module: {
1787   \stex_reactivate_macro:N \STEXexport
1788   \stex_reactivate_macro:N \importmodule
1789   \stex_reactivate_macro:N \symdecl
1790   \stex_reactivate_macro:N \notation
1791   \stex_reactivate_macro:N \symdef
1792 }
1793 \stex_debug:nn{modules}{
1794   New~module:\\
1795   Namespace:~\l_stex_module_ns_str\\
1796   Name:~\l_stex_module_name_str\\
1797   Language:~\l_stex_module_lang_str\\
1798   Signature:~\l_stex_module_sig_str\\
1799   Metatheory:~\l_stex_module_meta_str\\
1800   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1801 }
1802

```

```

1803 \stex_if_do_html:T{
1804   \begin{stex_annotate_env} {theory} {
1805     \l_stex_module_ns_str ? \l_stex_module_name_str
1806   }
1807
1808   \stex_annotate_invisible:nnn{header}{} {
1809     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1810     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1811     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1812       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1813     }
1814     \str_if_empty:NF \smoduletype {
1815       \stex_annotate:nnn{type}{\smoduletype}{}
1816     }
1817   }
1818 }
1819 % TODO: Inherit metatheory for nested modules?
1820 }
1821 \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}`

```

1822 \cs_new_protected:Nn \_stex_modules_end_module: {
1823   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module}
1824   \stex_reset_up_to_module:n \l_stex_current_module_str
1825   \stex_if_smsmode:T {
1826     \stex_persist:x {
1827       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1828         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1829       }
1830       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1831         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1832       }
1833       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1834         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1835       }
1836       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1837     }
1838     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1839     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1840   }
1841 }

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1842 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1843 \NewDocumentEnvironment { smodule } { 0 } { m } {
1844   \stex_module_setup:nn{#1}{#2}
1845   %\par
1846   \stex_if_smsmode:F{
1847     \tl_if_empty:NF \smoduletitle {
1848       \exp_args:No \stex_document_title:n \smoduletitle
1849     }

```

```

1850 \tl_clear:N \l_tmpa_tl
1851 \clist_map_inline:Nn \smodulotype {
1852   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1853     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1854   }
1855 }
1856 \tl_if_empty:NTF \l_tmpa_tl {
1857   \__stex_modules_smodule_start:
1858 }{
1859   \l_tmpa_tl
1860 }
1861 }
1862 \__stex_modules_begin_module:
1863 \str_if_empty:NF \smoduleid {
1864   \stex_ref_new_doc_target:n \smoduleid
1865 }
1866 \stex_smsmode_do:
1867 } {
1868   \__stex_modules_end_module:
1869   \stex_if_smsmode:F {
1870     \end{stex_annotate_env}
1871     \clist_set:Nn \l_tmpa_clist \smodulotype
1872     \tl_clear:N \l_tmpa_tl
1873     \clist_map_inline:Nn \l_tmpa_clist {
1874       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1875         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1876       }
1877     }
1878     \tl_if_empty:NTF \l_tmpa_tl {
1879       \__stex_modules_smodule_end:
1880     }{
1881       \l_tmpa_tl
1882     }
1883   }
1884 }

```

\stexpatchmodule

```

1885 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1886 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1887
1888 \newcommand\stexpatchmodule[3] [] {
1889   \str_set:Nx \l_tmpa_str{ #1 }
1890   \str_if_empty:NTF \l_tmpa_str {
1891     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1892     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1893   }{
1894     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1895     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1896   }
1897 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1898 \NewDocumentCommand \STEXModule { m } {
1899   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1900   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1901   \tl_set:Nn \l_tmpa_tl {
1902     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1903   }
1904   \seq_map_inline:Nn \l_stex_all_modules_seq {
1905     \str_set:Nn \l_tmpb_str { ##1 }
1906     \str_if_eq:eeT { \l_tmpa_str } {
1907       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1908     } {
1909       \seq_map_break:n {
1910         \tl_set:Nn \l_tmpa_tl {
1911           \stex_invoke_module:n { ##1 }
1912         }
1913       }
1914     }
1915   }
1916   \l_tmpa_tl
1917 }
1918
1919 \cs_new_protected:Nn \stex_invoke_module:n {
1920   \stex_debug:nn{modules}{Invoking~module~#1}
1921   \peek_charcode_remove:NTF ! {
1922     \__stex_modules_invoke_uri:nN { #1 }
1923   } {
1924     \peek_charcode_remove:NTF ? {
1925       \__stex_modules_invoke_symbol:nn { #1 }
1926     } {
1927       \msg_error:nnx{stex}{error/syntax}{
1928         ?~or~!~expected~after~
1929         \c_backslash_str STEXModule{#1}
1930       }
1931     }
1932   }
1933 }
1934
1935 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1936   \str_set:Nn #2 { #1 }
1937 }
1938
1939 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1940   \stex_invoke_symbol:n{#1?#2}
1941 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 80.)

```

\stex_activate_module:n
1942 \bool_new:N \l_stex_in_meta_bool
1943 \bool_set_false:N \l_stex_in_meta_bool

```



```

1944 \cs_new_protected:Nn \stex_activate_module:n {
1945   \stex_debug:nn{modules}{Activating~module~#1}
1946   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1947     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1948     \use:c{ c_stex_module_#1_code }
1949   }
1950 }

```

(End definition for \stex_activate_module:n. This function is documented on page 81.)

```

1951 \</package>

```

Chapter 28

STEX -Module Inheritance Implementation

```
1952 <*package>
1953
1954 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1955
```

28.1 SMS Mode

```
1956 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1957 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1958 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1959 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1960
1961 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1962   \makeatletter
1963   \makeatother
1964   \ExplSyntaxOn
1965   \ExplSyntaxOff
1966   \rustexBREAK
1967 }
1968
1969 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1970   \symdef
1971   \importmodule
1972   \notation
1973   \symdecl
1974   \STEXexport
1975   \inlineass
1976   \inlinedef
1977   \inlineex
1978   \endinput
1979   \setnotation
```

```

1980 \copynotation
1981 \assign
1982 \renamedekl
1983 \donotcopy
1984 \instantiate
1985 \textsymdecl
1986 }
1987
1988 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1989   \tl_to_str:n {
1990     smodule,
1991     copymodule,
1992     interpretmodule,
1993     realization,
1994     sdefinition,
1995     sexample,
1996     sassertion,
1997     sparagraph,
1998     mathstructure
1999   }
2000 }

```

(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 82.)

`\stex_if_smsmode_p:`

```

\stex_if_smsmode:TF
2001 \bool_new:N \g__stex_smsmode_bool
2002 \bool_set_false:N \g__stex_smsmode_bool
2003 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2004   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2005 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 82.)

`_stex_smsmode_in_smsmode:nn`

```

2006 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
2007   \vbox_set:Nn \l_tmpa_box {
2008     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
2009     \bool_gset_true:N \g__stex_smsmode_bool
2010     #2
2011     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2012   }
2013   \box_clear:N \l_tmpa_box
2014 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

2015 \quark_new:N \q__stex_smsmode_break
2016
2017 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
2018   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2019   \stex_smsmode_do:
2020 }
2021

```

```

2022 \cs_new_protected:Nn \__stex_smsmode_module:nn {
2023   \__stex_modules_args:n{#1}
2024   \stex_if_in_module:F {
2025     \str_if_empty:NF \l_stex_module_sig_str {
2026       \stex_modules_current_namespace:
2027       \str_set:Nx \l_stex_module_name_str { #2 }
2028       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2029         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2030         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2031         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2032         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2033         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2034         \str_set:Nx \l_tmpa_str {
2035           \stex_path_to_string:N \l_tmpa_seq /
2036           \l_tmpa_str . \l_stex_module_sig_str .tex
2037         }
2038         \IfFileExists \l_tmpa_str {
2039           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2040         }{
2041           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2042         }
2043       }
2044     }
2045   }
2046 }
2047
2048 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2049   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
2050   \tl_if_empty:nTF{#1}{
2051     \prop_if_exist:NTF \l_stex_current_repository_prop
2052     {
2053       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2054       \prg_return_true:
2055     } {
2056       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2057       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2058       \tl_if_empty:NT \l_tmpa_tl {
2059         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2060       }
2061       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
2062       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2063       \prg_return_true: \prg_return_false:
2064     }
2065   }\prg_return_true:
2066 }
2067
2068 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2069   \stex_filestack_push:n{#1}
2070   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2071   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2072   % ----- new -----
2073   \__stex_smsmode_in_smsmode:nn{#1}{
2074     \let\importmodule\__stex_smsmode_importmodule:
2075     \let\stex_module_setup:nn\__stex_smsmode_module:nn

```

```

2076 \let\__stex_modules_begin_module:\relax
2077 \let\__stex_modules_end_module:\relax
2078 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2079 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2080 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2081 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2082 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2083 \everyeof{\q__stex_smsmode_break\noexpand}
2084 \expandafter\expandafter\expandafter
2085 \stex_smsmode_do:
2086 \csname @ @ input\endcsname "#1"\relax
2087
2088 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2089   \stex_filestack_push:n{##1}
2090   \expandafter\expandafter\expandafter
2091   \stex_smsmode_do:
2092   \csname @ @ input\endcsname "##1"\relax
2093   \stex_filestack_pop:
2094 }
2095 }
2096 % ----- new -----
2097 \__stex_smsmode_in_smsmode:nn{#1} {
2098   #2
2099   % ----- new -----
2100   \begingroup
2101   \%stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2102   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2103     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
2104       \stex_import_module_uri:nn ##1
2105       \stex_import_require_module:nnnn
2106       \l_stex_import_ns_str
2107       \l_stex_import_archive_str
2108       \l_stex_import_path_str
2109       \l_stex_import_name_str \endgroup
2110     }
2111   }
2112   \endgroup
2113   \%stex_debug:nn{smsmode}{Actually~loading~file~#1}
2114   % ----- new -----
2115   \everyeof{\q__stex_smsmode_break\noexpand}
2116   \expandafter\expandafter\expandafter
2117   \stex_smsmode_do:
2118   \csname @ @ input\endcsname "#1"\relax
2119 }
2120 \stex_filestack_pop:
2121 }

```

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

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

```

2122 \cs_new_protected:Npn \stex_smsmode_do: {
2123   \stex_if_smsmode:T {
2124     \__stex_smsmode_do:w

```

```

2125 }
2126 }
2127 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2128   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2129     \expandafter\if\expandafter\relax\noexpand#1
2130     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2131     \else\expandafter\__stex_smsmode_do:w\fi
2132   }{
2133     \__stex_smsmode_do:w % #1
2134   }
2135 }
2136 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2137   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
2138     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2139       #1\__stex_smsmode_do:w
2140     }{
2141       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2142         #1
2143       }{
2144         \cs_if_eq:NNTF \begin #1 {
2145           \__stex_smsmode_check_begin:n
2146         }{
2147           \cs_if_eq:NNTF \end #1 {
2148             \__stex_smsmode_check_end:n
2149           }{
2150             \__stex_smsmode_do:w
2151           }
2152         }
2153       }
2154     }
2155   }
2156 }
2157
2158 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2159   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2160     \begin{#1}
2161   }{
2162     \__stex_smsmode_do:w
2163   }
2164 }
2165 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2166   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2167     \end{#1}\__stex_smsmode_do:w
2168   }{
2169     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2170   }
2171 }

```

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

28.2 Inheritance

```

2172 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```
2173 \cs_new_protected:Nn \stex_import_module_uri:nn {
2174   \str_set:Nx \l_stex_import_archive_str { #1 }
2175   \str_set:Nn \l_stex_import_path_str { #2 }
2176
2177   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2178   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2179   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2180
2181   \stex_modules_current_namespace:
2182   \bool_lazy_all:nTF {
2183     {\str_if_empty_p:N \l_stex_import_archive_str}
2184     {\str_if_empty_p:N \l_stex_import_path_str}
2185     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2186   }{
2187     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2188     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2189   }{
2190     \str_if_empty:NT \l_stex_import_archive_str {
2191       \prop_if_exist:NT \l_stex_current_repository_prop {
2192         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2193       }
2194     }
2195     \str_if_empty:NTF \l_stex_import_archive_str {
2196       \str_if_empty:NF \l_stex_import_path_str {
2197         \stex_path_from_string:Nn \l_tmpb_seq {
2198           \l_stex_module_ns_str / .. / \l_stex_import_path_str
2199         }
2200         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2201         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file:///}
2202       }
2203     }{
2204       \stex_require_repository:n \l_stex_import_archive_str
2205       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
2206       \l_stex_import_ns_str
2207       \str_if_empty:NF \l_stex_import_path_str {
2208         \str_set:Nx \l_stex_import_ns_str {
2209           \l_stex_import_ns_str / \l_stex_import_path_str
2210         }
2211       }
2212     }
2213   }
2214 }
```

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

`\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`.

```
2215 \str_new:N \l_stex_import_name_str
2216 \str_new:N \l_stex_import_archive_str
2217 \str_new:N \l_stex_import_path_str
2218 \str_new:N \l_stex_import_ns_str
```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 84.)

```

\stex_import_require_module:nnnnn {{\ns}} {{\archive-ID}} {{\path}} {{\name}}
2219 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
2220 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2221
2222 \stex_debug:nn{requiremodule}{Here:\~1:\~2:\~3:\~4}
2223
2224 \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2225 \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2226
2227 %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
2228
2229 % archive
2230 \str_set:Nx \l_tmpa_str { #2 }
2231 \str_if_empty:NTF \l_tmpa_str {
2232 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2233 \seq_put_right:Nn \l_tmpa_seq {...}
2234 } {
2235 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2236 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2237 \seq_put_right:Nn \l_tmpa_seq { source }
2238 }
2239
2240 % path
2241 \str_set:Nx \l_tmpb_str { #3 }
2242 \str_if_empty:NTF \l_tmpb_str {
2243 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2244
2245 \ltx@ifpackageloaded{babel} {
2246 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2247 { \language } \l_tmpb_str {
2248 \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2249 }
2250 } {
2251 \str_clear:N \l_tmpb_str
2252 }
2253
2254 \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2255 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2256 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2257 }{
2258 \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2259 \IfFileExists{ \l_tmpa_str.tex }{
2260 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2261 }{
2262 % try english as default
2263 \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2264 \IfFileExists{ \l_tmpa_str.en.tex }{
2265 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2266 }{
2267 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2268 }
2269 }
2270 }
2271

```



```

2272 } {
2273   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2274   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2275
2276   \ltx@ifpackageloaded{babel} {
2277     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2278       { \language } \l_tmpb_str {
2279       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2280     }
2281   } {
2282     \str_clear:N \l_tmpb_str
2283   }
2284
2285   \stex_path_canonicalize:N \l_tmpb_seq
2286   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2287
2288   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2289   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2290     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2291   }{
2292     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2293     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2294       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2295     }{
2296       % try english as default
2297       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2298       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2299         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2300       }{
2301         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2302         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2303           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2304         }{
2305           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2306           \IfFileExists{ \l_tmpa_str.tex }{
2307             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2308           }{
2309             % try english as default
2310             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2311             \IfFileExists{ \l_tmpa_str.en.tex }{
2312               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2313             }{
2314               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2315             }
2316           }
2317         }
2318       }
2319     }
2320   }
2321 }
2322
2323 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2324   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2325     \seq_clear:N \l_stex_all_modules_seq

```

```

2326     \str_clear:N \l_stex_current_module_str
2327     \str_set:Nx \l_tmpb_str { #2 }
2328     \str_if_empty:NF \l_tmpb_str {
2329       \stex_set_current_repository:n { #2 }
2330     }
2331     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2332   }
2333
2334   \stex_if_module_exists:nF { #1 ? #4 } {
2335     \msg_error:nnx{stex}{error/unknownmodule}{
2336       #1?#4~(in~file~\g__stex_importmodule_file_str)
2337     }
2338   }
2339 }
2340
2341 }
2342 \stex_activate_module:n { #1 ? #4 }
2343 }

```

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

`\importmodule`

```

2344 \NewDocumentCommand \importmodule { 0{} m } {
2345   \stex_import_module_uri:nn { #1 } { #2 }
2346   \stex_debug:nn{modules}{Importing~module:~
2347     \l_stex_import_ns_str ? \l_stex_import_name_str
2348   }
2349   \stex_import_require_module:nnnn
2350   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2351   { \l_stex_import_path_str } { \l_stex_import_name_str }
2352   \stex_if_smsmode:F {
2353     \stex_annotate_invisible:nnn
2354     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2355   }
2356   \exp_args:Nx \stex_add_to_current_module:n {
2357     \stex_import_require_module:nnnn
2358     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2359     { \l_stex_import_path_str } { \l_stex_import_name_str }
2360   }
2361   \exp_args:Nx \stex_add_import_to_current_module:n {
2362     \l_stex_import_ns_str ? \l_stex_import_name_str
2363   }
2364   \stex_smsmode_do:
2365   \ignorespacesandpars
2366 }
2367 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

`\usemodule`

```

2368 \NewDocumentCommand \usemodule { 0{} m } {
2369   \stex_if_smsmode:F {
2370     \stex_import_module_uri:nn { #1 } { #2 }
2371     \stex_import_require_module:nnnn
2372     { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```

```

2373 { \l_stex_import_path_str } { \l_stex_import_name_str }
2374 \stex_annotate_invisible:nnn
2375 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2376 }
2377 \stex_smsmode_do:
2378 \ignorespacesandpars
2379 }

```

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

```

2380 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2381   \tl_if_empty:nF{#2}{
2382     \clist_set:Nn \l_tmpa_clist {#2}
2383     \clist_map_inline:Nn \l_tmpa_clist {
2384       \tl_if_head_eq_charcode:nNTF {##1} [{
2385         #1 ##1
2386       }{
2387         #1{##1}
2388       }
2389     }
2390   }
2391 }
2392 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2393
2394
2395 \endpackage

```

Chapter 29

STEX -Symbols Implementation

```
2396 <*package>
2397
2398 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2399
    Warnings and error messages
2400 \msg_new:nnn{stex}{error/wrongargs}{
2401   args~value~in~symbol~declaration~for~#1~
2402   needs~to~be~i,~a,~b~or~B,~but~#2~given
2403 }
2404 \msg_new:nnn{stex}{error/unknownsymbol}{
2405   No~symbol~#1~found!
2406 }
2407 \msg_new:nnn{stex}{error/seqlength}{
2408   Expected~#1~arguments;~got~#2!
2409 }
2410 \msg_new:nnn{stex}{error/unknownnotation}{
2411   Unknown~notation~#1~for~#2!
2412 }
```

29.1 Symbol Declarations

```
2413 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2414 \cs_new_protected:Nn \stex_all_symbols:n {
2415   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2416   \seq_map_inline:Nn \l_stex_all_modules_seq {
2417     \seq_map_inline:cn{c_stex_module_##1_constants}{
2418       \__stex_symdecl_all_symbols_cs{##1?####1}
2419     }
2420   }
2421 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page [86](#).)

\STEXsymbol

```
2422 \NewDocumentCommand \STEXsymbol { m } {
2423   \stex_get_symbol:n { #1 }
2424   \exp_args:No
2425   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2426 }
```

(End definition for \STEXsymbol. This function is documented on page 87.)

symdecl arguments:

```
2427 \keys_define:nn { stex / symdecl } {
2428   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2429   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2430   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2431   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2432   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2433   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2434   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2435   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2436   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2437   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2438   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
2439   assoc     .choices:nn =
2440     {bin,binl,binr,pre,conj,pwconj}
2441     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2442 }
2443
2444 \bool_new:N \l_stex_symdecl_make_macro_bool
2445
2446 \cs_new_protected:Nn \__stex_symdecl_args:n {
2447   \str_clear:N \l_stex_symdecl_name_str
2448   \str_clear:N \l_stex_symdecl_args_str
2449   \str_clear:N \l_stex_symdecl_deprecate_str
2450   \str_clear:N \l_stex_symdecl_reorder_str
2451   \str_clear:N \l_stex_symdecl_assoctype_str
2452   \bool_set_false:N \l_stex_symdecl_local_bool
2453   \tl_clear:N \l_stex_symdecl_type_tl
2454   \tl_clear:N \l_stex_symdecl_definiens_tl
2455   \clist_clear:N \l_stex_symdecl_argnames_clist
2456
2457   \keys_set:nn { stex / symdecl } { #1 }
2458 }
```

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

```
2459
2460 \NewDocumentCommand \symdecl { s m O{} } {
2461   \__stex_symdecl_args:n { #3 }
2462   \IfBooleanTF #1 {
2463     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2464   } {
2465     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2466   }
2467   \stex_symdecl_do:n { #2 }
```

```

2468 \stex_smsmode_do:
2469 }
2470
2471 \cs_new_protected:Nn \stex_symdecl_do:nn {
2472   \__stex_symdecl_args:n{#1}
2473   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2474   \stex_symdecl_do:n{#2}
2475 }
2476
2477 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2478 \cs_new_protected:Nn \stex_symdecl_do:n {
2479   \stex_if_in_module:F {
2480     % TODO throw error? some default namespace?
2481   }
2482
2483   \str_if_empty:NT \l_stex_symdecl_name_str {
2484     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2485   }
2486
2487   \prop_if_exist:cT { \l_stex_symdecl_
2488     \l_stex_current_module_str ?
2489     \l_stex_symdecl_name_str
2490     _prop
2491   }{
2492     % TODO throw error (beware of circular dependencies)
2493   }
2494
2495   \prop_clear:N \l_tmpa_prop
2496   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2497   \seq_clear:N \l_tmpa_seq
2498   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2499   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2500
2501   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2502     \str_if_empty:NF \l_stex_module_deprecate_str {
2503       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2504     }
2505   }
2506   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2507
2508   \exp_args:No \stex_add_constant_to_current_module:n {
2509     \l_stex_symdecl_name_str
2510   }
2511
2512   % arity/args
2513   \int_zero:N \l_tmpb_int
2514
2515   \bool_set_true:N \l_tmpa_bool
2516   \str_map_inline:Nn \l_stex_symdecl_args_str {
2517     \token_case_meaning:NnF ##1 {

```

```

2518     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2519     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2520     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2521     {\tl_to_str:n a} {
2522         \bool_set_false:N \l_tmpa_bool
2523         \int_incr:N \l_tmpb_int
2524     }
2525     {\tl_to_str:n B} {
2526         \bool_set_false:N \l_tmpa_bool
2527         \int_incr:N \l_tmpb_int
2528     }
2529 }{
2530     \msg_error:nnxx{stex}{error/wrongargs}{
2531         \l_stex_current_module_str ?
2532         \l_stex_symdecl_name_str
2533     }{##1}
2534 }
2535 }
2536
2537 \bool_if:NTF \l_tmpa_bool {
2538     % possibly numeric
2539     \str_if_empty:NTF \l_stex_symdecl_args_str {
2540         \prop_put:Nnn \l_tmpa_prop { args } {}
2541         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2542     }{
2543         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2544         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2545         \str_clear:N \l_tmpa_str
2546         \int_step_inline:nn \l_tmpa_int {
2547             \str_put_right:Nn \l_tmpa_str i
2548         }
2549         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2550     }
2551 } {
2552     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2553     \prop_put:Nnx \l_tmpa_prop { arity }
2554     { \str_count:N \l_stex_symdecl_args_str }
2555 }
2556 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2557
2558 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2559     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2560 }{
2561     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2562 }
2563
2564 % argnames
2565
2566 \clist_clear:N \l_tmpa_clist
2567 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
2568     \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
2569         \clist_put_right:Nn \l_tmpa_clist {##1}
2570     }{
2571         \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl

```

```

2572     \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
2573   }
2574 }
2575 \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
2576
2577 % semantic macro
2578
2579 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2580   \exp_args:Nx \stex_do_up_to_module:n {
2581     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2582       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2583     }}
2584   }
2585 }
2586
2587 \stex_debug:nn{symbols}{New~symbol:~
2588   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2589   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2590   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2591   Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2592 }
2593
2594 % circular dependencies require this:
2595 \stex_if_do_html:T {
2596   \stex_annotate_invisible:nnn {symdecl} {
2597     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2598   } {
2599     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2600       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2601     }
2602     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2603     \stex_annotate_invisible:nnn{macroname}{#1}{}
2604     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2605       \stex_annotate_invisible:nnn{definiens}{}
2606       {\l_stex_symdecl_definiens_tl$}
2607     }
2608     \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2609       \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2610     }
2611     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2612       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2613     }
2614   }
2615 }
2616 \prop_if_exist:cF {
2617   l_stex_symdecl_
2618   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2619   _prop
2620 } {
2621   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2622     \_stex_symdecl_restore_symbol:nnnnnnn
2623     {\l_stex_symdecl_name_str}
2624     { \prop_item:Nn \l_tmpa_prop {args} }
2625     { \prop_item:Nn \l_tmpa_prop {arity} }

```



```

2626     { \prop_item:Nn \l_tmpa_prop {assocs} }
2627     { \prop_item:Nn \l_tmpa_prop {defined} }
2628     {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2629     {\l_stex_current_module_str}
2630     { \prop_item:Nn \l_tmpa_prop {argnames} }
2631   }
2632 }
2633 }
2634 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2635   \prop_clear:N \l_tmpa_prop
2636   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2637   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2638   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2639   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2640   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2641   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2642   \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2643   \tl_if_empty:nF{#6}{
2644     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2645   }
2646   \prop_set_eq:cN{\l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2647   \seq_clear:c{\l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2648 }

```

(End definition for \stex_symdecl_do:n. This function is documented on page 86.)

\textsymdecl

```

2649
2650 \keys_define:nn { stex / textsymdecl } {
2651   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2652   type      .tl_set:N    = \l__stex_symdecl_type_tl
2653 }
2654
2655 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2656   \str_clear:N \l__stex_symdecl_name_str
2657   \tl_clear:N \l__stex_symdecl_type_tl
2658   \clist_clear:N \l_stex_symdecl_argnames_clist
2659   \keys_set:nn { stex / textsymdecl } { #1 }
2660 }
2661
2662 \NewDocumentCommand \textsymdecl {m O{} m} {
2663   \_stex_textsymdecl_args:n { #2 }
2664   \str_if_empty:NTF \l__stex_symdecl_name_str {
2665     \__stex_symdecl_args:n{name=#1,#2}
2666   }{
2667     \__stex_symdecl_args:n{#2}
2668   }
2669   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2670   \stex_symdecl_do:n{#1-sym}
2671   \stex_execute_in_module:n{
2672     \cs_set_nopar:cpn{#1name}{
2673       \ifvmode\hbox_unpack:N\c_empty_box\fi
2674       \ifmmode\hbox{#3}\else#3\fi\xspace
2675     }

```

```

2676 \cs_set_nopar:cpn{#1}{
2677   \ifmmode\csname#1-sym\expandafter\endcsname\else
2678   \ifvmode\hbox_unpack:N\c_empty_box\fi
2679   \symref{#1-sym}{#3}\expandafter\xspace
2680   \fi
2681 }
2682 }
2683 \stex_execute_in_module:x{
2684   \__stex_notation_restore_notation:nnnnn
2685   {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl_name_str{#1}}{0}
2686   {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfpref}{\neginfpref}}{
2687     \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2688   }}
2689   {}
2690 }
2691 }
2692 \stex_smsmode_do:
2693 }

```

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

\stex_get_symbol:n

```

2694 \str_new:N \l_stex_get_symbol_uri_str
2695
2696 \cs_new_protected:Nn \stex_get_symbol:n {
2697   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2698     \tl_set:Nn \l_tmpa_tl { #1 }
2699     \__stex_symdecl_get_symbol_from_cs:
2700   }{
2701     % argument is a string
2702     % is it a command name?
2703     \cs_if_exist:cTF { #1 }{
2704       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2705       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2706       \str_if_empty:NTF \l_tmpa_str {
2707         \exp_args:Nx \cs_if_eq:NTF {
2708           \tl_head:N \l_tmpa_tl
2709         } \stex_invoke_symbol:n {
2710           \__stex_symdecl_get_symbol_from_cs:
2711         }{
2712           \__stex_symdecl_get_symbol_from_string:n { #1 }
2713         }
2714       } {
2715         \__stex_symdecl_get_symbol_from_string:n { #1 }
2716       }
2717     }{
2718       % argument is not a command name
2719       \__stex_symdecl_get_symbol_from_string:n { #1 }
2720       % \l_stex_all_symbols_seq
2721     }
2722   }
2723   \str_if_eq:eeF {
2724     \prop_item:cn {
2725       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop

```

```

2726     }{ deprecate }
2727 }{}{
2728     \msg_warning:nxxx{stex}{warning/deprecated}{
2729         Symbol~\l_stex_get_symbol_uri_str
2730     }{
2731         \prop_item:cn {l_stex_symdecl_l\l_stex_get_symbol_uri_str _prop}{ deprecate }
2732     }
2733 }
2734 }
2735
2736 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2737     \tl_set:Nn \l_tmpa_tl {
2738         \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2739     }
2740     \str_set:Nn \l_tmpa_str { #1 }
2741
2742     %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2743
2744     \str_if_in:NnTF \l_tmpa_str ? {
2745         \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2746         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2747         \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2748     }{
2749         \str_clear:N \l_tmpb_str
2750     }
2751     \str_if_empty:NNTF \l_tmpb_str {
2752         \seq_map_inline:Nn \l_stex_all_modules_seq {
2753             \seq_map_inline:cn{c_stex_module_##1_constants}{
2754                 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2755                     \seq_map_break:n{\seq_map_break:n{
2756                         \tl_set:Nn \l_tmpa_tl {
2757                             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2758                         }
2759                     }}
2760                 }
2761             }
2762         }
2763     }{
2764         \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2765         \seq_map_inline:Nn \l_stex_all_modules_seq {
2766             \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2767                 \seq_map_inline:cn{c_stex_module_##1_constants}{
2768                     \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2769                         \seq_map_break:n{\seq_map_break:n{
2770                             \tl_set:Nn \l_tmpa_tl {
2771                                 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2772                             }
2773                         }}
2774                     }
2775                 }
2776             }
2777         }
2778     }
2779

```

```

2780 \l_tmpa_tl
2781 }
2782
2783 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2784   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2785     { \tl_tail:N \l_tmpa_tl }
2786   \tl_if_single:NTF \l_tmpa_tl {
2787     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2788       \exp_after:wN \str_set:Nn \exp_after:wN
2789         \l_stex_get_symbol_uri_str \l_tmpa_tl
2790     }{
2791       % TODO
2792       % tail is not a single group
2793     }
2794   }{
2795     % TODO
2796     % tail is not a single group
2797   }
2798 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 86.)

29.2 Notations

```

2799 <@@=stex_notation>
      notation arguments:
2800 \keys_define:nn { stex / notation } {
2801   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2802   variant .tl_set_x:N   = \l__stex_notation_variant_str ,
2803   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2804   op       .tl_set:N    = \l__stex_notation_op_tl ,
2805   primary  .bool_set:N  = \l__stex_notation_primary_bool ,
2806   primary  .default:n   = {true} ,
2807   hints    .str_set_x:N = \l__stex_notation_hints_str,
2808   unknown  .code:n      = \str_set:Nx
2809     \l__stex_notation_variant_str \l_keys_key_str
2810 }
2811
2812 \cs_new_protected:Nn \_stex_notation_args:n {
2813   % \str_clear:N \l__stex_notation_lang_str
2814   \str_clear:N \l__stex_notation_variant_str
2815   \str_clear:N \l__stex_notation_prec_str
2816   \str_clear:N \l__stex_notation_hints_str
2817   \tl_clear:N \l__stex_notation_op_tl
2818   \bool_set_false:N \l__stex_notation_primary_bool
2819
2820   \keys_set:nn { stex / notation } { #1 }
2821 }

```

\notation

```

2822 \NewDocumentCommand \notation { s m O{}} {
2823   \_stex_notation_args:n { #3 }
2824   \tl_clear:N \l_stex_symdecl_definiens_tl

```

```

2825 \stex_get_symbol:n { #2 }
2826 \tl_set:Nn \l_stex_notation_after_do_tl {
2827   \__stex_notation_final:
2828   \IfBooleanTF#1{
2829     \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2830   }{}
2831   \stex_smsmode_do:\ignorespacesandpars
2832 }
2833 \stex_notation_do:nnnnn
2834 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2835 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2836 { \l__stex_notation_variant_str }
2837 { \l__stex_notation_prec_str }
2838 }
2839 \stex_deactivate_macro:Nn \notation {module~environments}

```

(End definition for \notation. This function is documented on page 86.)

\stex_notation_do:nnnnn

```

2840 \seq_new:N \l__stex_notation_precedences_seq
2841 \tl_new:N \l__stex_notation_opprec_tl
2842 \int_new:N \l__stex_notation_currarg_int
2843 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2844
2845 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2846   \let\STEXInternalCurrentSymbolStr\relax
2847   \seq_clear:N \l__stex_notation_precedences_seq
2848   \tl_clear:N \l__stex_notation_opprec_tl
2849   \str_set:Nx \l__stex_notation_args_str { #1 }
2850   \str_set:Nx \l__stex_notation_arity_str { #2 }
2851   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2852   \str_set:Nx \l__stex_notation_prec_str { #4 }
2853
2854   % precedences
2855   \str_if_empty:NTF \l__stex_notation_prec_str {
2856     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2857       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2858     }{
2859       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2860     }
2861   } {
2862     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2863       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2864       \int_step_inline:nn { \l__stex_notation_arity_str } {
2865         \exp_args:NNo
2866         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2867       }
2868     }{
2869       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2870       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2871         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2872         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2873           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2874             \l_tmpa_seq {\tl_to_str:n{x}} } { \l_tmpa_str }

```

```

2875         \seq_map_inline:Nn \l_tmpa_seq {
2876             \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }
2877         }
2878     }
2879 }{
2880     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2881         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2882     }{
2883         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2884     }
2885 }
2886 }
2887 }
2888
2889 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2890 \int_step_inline:nn { \l__stex_notation_arity_str } {
2891     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_seq {
2892         \exp_args:NNo
2893         \seq_put_right:No \l__stex_notation_precedences_seq {
2894             \l__stex_notation_opprec_tl
2895         }
2896     }
2897 }
2898 \tl_clear:N \l_stex_notation_dummyargs_tl
2899
2900 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2901     \exp_args:NNe
2902     \cs_set:Npn \l_stex_notation_macrocode_cs {
2903         \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2904         { \l__stex_notation_suffix_str }
2905         { \l__stex_notation_opprec_tl }
2906         { \exp_not:n { #5 } }
2907     }
2908     \l_stex_notation_after_do_tl
2909 }{
2910     \str_if_in:NnTF \l__stex_notation_args_str b {
2911         \exp_args:Nne \use:nn
2912         {
2913             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2914             \cs_set:Npn \l__stex_notation_arity_str } { {
2915                 \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2916                 { \l__stex_notation_suffix_str }
2917                 { \l__stex_notation_opprec_tl }
2918                 { \exp_not:n { #5 } }
2919             }}
2920     }{
2921         \str_if_in:NnTF \l__stex_notation_args_str B {
2922             \exp_args:Nne \use:nn
2923             {
2924                 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2925                 \cs_set:Npn \l__stex_notation_arity_str } { {
2926                     \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2927                     { \l__stex_notation_suffix_str }
2928                     { \l__stex_notation_opprec_tl }

```

```

2929         { \exp_not:n { #5 } }
2930     } }
2931 }{
2932   \exp_args:Nne \use:nn
2933   {
2934     \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2935     \cs_set:Npn \l__stex_notation_arity_str } { {
2936       \STEXInternalTermMathOMAiiai { \STEXInternalCurrentSymbolStr }
2937       { \l__stex_notation_suffix_str }
2938       { \l__stex_notation_opprec_tl }
2939       { \exp_not:n { #5 } }
2940     } }
2941   }
2942 }
2943
2944 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2945 \int_zero:N \l__stex_notation_currarg_int
2946 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2947 \__stex_notation_arguments:
2948 }
2949 }

```

(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

```

2950 \cs_new_protected:Nn \__stex_notation_arguments: {
2951   \int_incr:N \l__stex_notation_currarg_int
2952   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2953     \l_stex_notation_after_do_tl
2954   }{
2955     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2956     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2957     \str_if_eq:VnTF \l_tmpa_str a {
2958       \__stex_notation_argument_assoc:nn{a}
2959     }{
2960       \str_if_eq:VnTF \l_tmpa_str B {
2961         \__stex_notation_argument_assoc:nn{B}
2962       }{
2963         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2964         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2965           { \STEXInternalTermMathArgiii
2966             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2967             { \l_tmpb_str }
2968             { ###\int_use:N \l__stex_notation_currarg_int }
2969           }
2970         }
2971         \__stex_notation_arguments:
2972       }
2973     }
2974   }
2975 }

```

(End definition for `__stex_notation_arguments:.`)

_stex_notation_argument_assoc:nn

```

2976 \cs_new_protected:Nn \_stex_notation_argument_assoc:nn {
2977
2978   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2979     {\l_stex_notation_arity_str}{
2980       #2
2981     }
2982   \int_zero:N \l_tmpa_int
2983   \tl_clear:N \l_tmpa_tl
2984   \str_map_inline:Nn \l_stex_notation_args_str {
2985     \int_incr:N \l_tmpa_int
2986     \tl_put_right:Nx \l_tmpa_tl {
2987       \str_if_eq:nnTF {##1}{a}{ {} }{
2988         \str_if_eq:nnTF {##1}{B}{ {} }{
2989           {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_int}
2990         }
2991       }
2992     }
2993   }
2994   \exp_after:wN\exp_after:wN\exp_after:wN \def
2995   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2996   \exp_after:wN\exp_after:wN\exp_after:wN ##
2997   \exp_after:wN\exp_after:wN\exp_after:wN 1
2998   \exp_after:wN\exp_after:wN\exp_after:wN ##
2999   \exp_after:wN\exp_after:wN\exp_after:wN 2
3000   \exp_after:wN\exp_after:wN\exp_after:wN {
3001     \exp_after:wN \exp_after:wN \exp_after:wN
3002     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
3003       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3004     }
3005   }
3006
3007   \seq_pop_left:NN \l_stex_notation_remaining_precs_seq \l_tmpa_str
3008   \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
3009     \STEXInternalTermMathAssocArgiiii
3010     { \int_use:N \l_stex_notation_currarg_int }
3011     { \l_tmpa_str }
3012     { ####\int_use:N \l_stex_notation_currarg_int }
3013     { \l_tmpa_cs {####1} {####2} }
3014     {#1}
3015   } }
3016   \_stex_notation_arguments:
3017 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

3018 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
3019   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
3020   \cs_set_nopar:Npn {#3}{#4}
3021   \tl_if_empty:nF {#5}{
3022     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
3023   }
3024   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{

```



```

3025     \seq_put_right:cx { \l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
3026   }
3027 }
3028
3029 \cs_new_protected:Nn \__stex_notation_final: {
3030
3031   \stex_execute_in_module:x {
3032     \__stex_notation_restore_notation:nnnnn
3033     {\l_stex_get_symbol_uri_str}
3034     {\l__stex_notation_suffix_str}
3035     {\l__stex_notation_arity_str}
3036     {
3037       \exp_after:wN \exp_after:wN \exp_after:wN
3038       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3039       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3040     }
3041     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3042   }
3043
3044   \stex_debug:nn{symbols}{
3045     Notation~\l__stex_notation_suffix_str
3046     ~for~\l_stex_get_symbol_uri_str^^J
3047     Operator~precedence:~\l__stex_notation_opprec_tl^^J
3048     Argument~precedences:~
3049     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3050     Notation: \cs_meaning:c {
3051       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3052       \l__stex_notation_suffix_str
3053       _cs
3054     }
3055   }
3056   % HTML annotations
3057   \stex_if_do_html:T {
3058     \stex_annotate_invisible:nnn { notation }
3059     { \l_stex_get_symbol_uri_str } {
3060       \stex_annotate_invisible:nnn { notationfragment }
3061       { \l__stex_notation_suffix_str }{}
3062     }
3063     \stex_annotate_invisible:nnn { precedence }
3064     { \l__stex_notation_prec_str }{}
3065
3066     \int_zero:N \l_tmpa_int
3067     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3068     \tl_clear:N \l_tmpa_tl
3069     \int_step_inline:nn { \l__stex_notation_arity_str }{
3070       \int_incr:N \l_tmpa_int
3071       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3072       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
3073       \str_if_eq:VnTF \l_tmpb_str a {
3074         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3075           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{},
3076           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{},
3077         } }
3078       }{
3079         \str_if_eq:VnTF \l_tmpb_str B {

```

```

3079         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3080             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
3081             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
3082         } }
3083     }{
3084         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3085             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{ }
3086         } }
3087     }
3088 }
3089 }
3090 \stex_annotate_invisible:nnn { notationcomp }{ }{
3091     \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3092     $ \exp_args:Nno \use:nn { \use:c {
3093         stex_notation_ \STEXInternalCurrentSymbolStr
3094         \c_hash_str \l__stex_notation_suffix_str _cs
3095     } } { \l_tmpa_tl } $
3096 }
3097 \tl_if_empty:NF \l__stex_notation_op_tl {
3098     \stex_annotate_invisible:nnn { notationopcomp }{ }{
3099         $\l__stex_notation_op_tl$
3100     }
3101 }
3102 }
3103 }
3104 }

```

(End definition for `_stex_notation_final:.`)

`\setnotation`

```

3105 \keys_define:nn { stex / setnotation } {
3106     % lang .tl_set_x:N = \l__stex_notation_lang_str ,
3107     variant .tl_set_x:N = \l__stex_notation_variant_str ,
3108     unknown .code:n = \str_set:Nx
3109         \l__stex_notation_variant_str \l_keys_key_str
3110 }
3111
3112 \cs_new_protected:Nn \_stex_setnotation_args:n {
3113     % \str_clear:N \l__stex_notation_lang_str
3114     \str_clear:N \l__stex_notation_variant_str
3115     \keys_set:nn { stex / setnotation } { #1 }
3116 }
3117
3118 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
3119     \seq_if_exist:CT{l_stex_symdecl_#1_notations}{
3120         \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
3121         \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
3122     }
3123 }
3124
3125 \cs_new_protected:Nn \stex_setnotation:n {
3126     \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
3127         { \l__stex_notation_variant_str }{
3128         \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari

```

```

3129 \stex_debug:nn {notations}{
3130   Setting~default~notation~
3131   {\l__stex_notation_variant_str }~for~
3132   #1 \
3133   \expandafter\meaning\csname
3134   l_stex_symdecl_#1 _notations\endcsname
3135 }
3136 }{
3137   \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
3138 }
3139 }
3140
3141 \NewDocumentCommand \setnotation {m m} {
3142   \stex_get_symbol:n { #1 }
3143   \_stex_setnotation_args:n { #2 }
3144   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3145   \stex_smsmode_do:\ignorespacesandpars
3146 }
3147
3148 \cs_new_protected:Nn \stex_copy_notations:nn {
3149   \stex_debug:nn {notations}{
3150     Copying~notations~from~#2~to~#1\
3151     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3152   }
3153   \tl_clear:N \l_tmpa_tl
3154   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3155     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
3156   }
3157   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3158     \stex_debug:nn{Here}{Here:~##1}
3159     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3160     \edef \l_tmpa_tl {
3161       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3162       \exp_after:wN\exp_after:wN\exp_after:wN {
3163         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3164       }
3165     }
3166
3167     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3168     \exp_after:wN ####\exp_after:wN 1 \exp_after:wN ####\exp_after:wN 2
3169     \exp_after:wN { \l_tmpa_tl }
3170
3171     \edef \l_tmpa_tl {
3172       \exp_after:wN \exp_not:n \exp_after:wN {
3173         \l_tmpa_tl {##### 1}{##### 2}
3174       }
3175     }
3176
3177     \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3178
3179   \stex_execute_in_module:x {
3180     \__stex_notation_restore_notation:nnnnn
3181     {#1}{##1}
3182     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }

```

```

3183     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
3184     {
3185         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3186             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3187         }
3188     }
3189 } \endgroup
3190 }
3191 }
3192
3193 \NewDocumentCommand \copynotation {m m} {
3194     \stex_get_symbol:n { #1 }
3195     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3196     \stex_get_symbol:n { #2 }
3197     \exp_args:Noo
3198     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3199     \stex_smsmode_do:\ignorespacesandpars
3200 }
3201

```

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

\symdef

```

3202 \keys_define:nn { stex / symdef } {
3203     name .str_set_x:N = \l_stex_symdecl_name_str ,
3204     local .bool_set:N = \l_stex_symdecl_local_bool ,
3205     args .str_set_x:N = \l_stex_symdecl_args_str ,
3206     type .tl_set:N = \l_stex_symdecl_type_tl ,
3207     def .tl_set:N = \l_stex_symdecl_definiens_tl ,
3208     reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3209     op .tl_set:N = \l__stex_notation_op_tl ,
3210     % lang .str_set_x:N = \l__stex_notation_lang_str ,
3211     variant .str_set_x:N = \l__stex_notation_variant_str ,
3212     prec .str_set_x:N = \l__stex_notation_prec_str ,
3213     argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3214     assoc .choices:nn =
3215         {bin,binl,binr,pre,conj,pwconj}
3216         {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3217     unknown .code:n = \str_set:Nx
3218         \l__stex_notation_variant_str \l_keys_key_str
3219 }
3220
3221 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3222     \str_clear:N \l_stex_symdecl_name_str
3223     \str_clear:N \l_stex_symdecl_args_str
3224     \str_clear:N \l_stex_symdecl_assoctype_str
3225     \str_clear:N \l_stex_symdecl_reorder_str
3226     \bool_set_false:N \l_stex_symdecl_local_bool
3227     \tl_clear:N \l_stex_symdecl_type_tl
3228     \tl_clear:N \l_stex_symdecl_definiens_tl
3229     \clist_clear:N \l_stex_symdecl_argnames_clist
3230     % \str_clear:N \l__stex_notation_lang_str
3231     \str_clear:N \l__stex_notation_variant_str
3232     \str_clear:N \l__stex_notation_prec_str

```

```

3233 \tl_clear:N \l__stex_notation_op_tl
3234
3235 \keys_set:nn { stex / symdef } { #1 }
3236 }
3237
3238 \NewDocumentCommand \symdef { m O{} } {
3239   \__stex_notation_symdef_args:n { #2 }
3240   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3241   \stex_symdecl_do:n { #1 }
3242   \tl_set:Nn \l_stex_notation_after_do_tl {
3243     \__stex_notation_final:
3244     \stex_smsmode_do:\ignorespacesandpars
3245   }
3246   \str_set:Nx \l_stex_get_symbol_uri_str {
3247     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3248   }
3249   \exp_args:Nx \stex_notation_do:nnnnn
3250     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { args } }
3251     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { arity } }
3252     { \l__stex_notation_variant_str }
3253     { \l__stex_notation_prec_str }
3254 }
3255 \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for `\symdef`. This function is documented on page [86](#).)

29.3 Variables

```

3256 <@@=stex_variables>
3257
3258 \keys_define:nn { stex / vardef } {
3259   name .str_set_x:N = \l__stex_variables_name_str ,
3260   args .str_set_x:N = \l__stex_variables_args_str ,
3261   type .tl_set:N = \l__stex_variables_type_tl ,
3262   def .tl_set:N = \l__stex_variables_def_tl ,
3263   op .tl_set:N = \l__stex_variables_op_tl ,
3264   prec .str_set_x:N = \l__stex_variables_prec_str ,
3265   reorder .str_set_x:N = \l__stex_variables_reorder_str ,
3266   argnames .clist_set:N = \l__stex_variables_argnames_clist ,
3267   assoc .choices:nn =
3268     {bin,binl,binr,pre,conj,pwconj}
3269     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3270   bind .choices:nn =
3271     {forall,exists}
3272     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3273 }
3274
3275 \cs_new_protected:Nn \__stex_variables_args:n {
3276   \str_clear:N \l__stex_variables_name_str
3277   \str_clear:N \l__stex_variables_args_str
3278   \str_clear:N \l__stex_variables_prec_str
3279   \str_clear:N \l__stex_variables_assoctype_str
3280   \str_clear:N \l__stex_variables_reorder_str
3281   \str_clear:N \l__stex_variables_bind_str

```

```

3282 \tl_clear:N \l__stex_variables_type_tl
3283 \tl_clear:N \l__stex_variables_def_tl
3284 \tl_clear:N \l__stex_variables_op_tl
3285 \clist_clear:N \l__stex_variables_argnames_clist
3286
3287 \keys_set:nn { stex / vardef } { #1 }
3288 }
3289
3290 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3291   \__stex_variables_args:n {#2}
3292   \str_if_empty:NT \l__stex_variables_name_str {
3293     \str_set:Nx \l__stex_variables_name_str { #1 }
3294   }
3295   \prop_clear:N \l_tmpa_prop
3296   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3297
3298   \int_zero:N \l_tmpb_int
3299   \bool_set_true:N \l_tmpa_bool
3300   \str_map_inline:Nn \l__stex_variables_args_str {
3301     \token_case_meaning:NnF ##1 {
3302       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3303       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3304       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3305       {\tl_to_str:n a} {
3306         \bool_set_false:N \l_tmpa_bool
3307         \int_incr:N \l_tmpb_int
3308       }
3309       {\tl_to_str:n B} {
3310         \bool_set_false:N \l_tmpa_bool
3311         \int_incr:N \l_tmpb_int
3312       }
3313     }{
3314       \msg_error:nnxx{stex}{error/wrongargs}{
3315         variable~\l__stex_variables_name_str
3316       }{##1}
3317     }
3318   }
3319   \bool_if:NTF \l_tmpa_bool {
3320     % possibly numeric
3321     \str_if_empty:NTF \l__stex_variables_args_str {
3322       \prop_put:Nnn \l_tmpa_prop { args } {}
3323       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3324     }{
3325       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3326       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3327       \str_clear:N \l_tmpa_str
3328       \int_step_inline:nn \l_tmpa_int {
3329         \str_put_right:Nn \l_tmpa_str i
3330       }
3331       \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3332       \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3333     }
3334   } {
3335     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }

```

```

3336     \prop_put:Nnx \l_tmpa_prop { arity }
3337     { \str_count:N \l__stex_variables_args_str }
3338 }
3339 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
3340 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3341
3342 % argnames
3343
3344 \clist_clear:N \l_tmpa_clist
3345 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
3346   \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3347     \clist_put_right:Nn \l_tmpa_clist {##1}
3348   }{
3349     \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3350     \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
3351   }
3352 }
3353 \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3354
3355
3356 \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop} \l_tmpa_prop
3357
3358 \tl_if_empty:NF \l__stex_variables_op_tl {
3359   \cs_set:cpx {
3360     stex_var_op_notation_ \l__stex_variables_name_str _cs
3361   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3362 }
3363
3364 \tl_set:Nn \l_stex_notation_after_do_tl {
3365   \exp_args:Nne \use:nn {
3366     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3367     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3368   } {{
3369     \exp_after:wN \exp_after:wN \exp_after:wN
3370     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3371     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3372   }}
3373 \stex_if_do_html:T {
3374   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3375     \stex_annotate_invisible:nnn { precedence }
3376     { \l__stex_variables_prec_str }{}
3377   \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{
3378     \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }{}
3379     \stex_annotate_invisible:nnn{macroname}{#1}{
3380     \tl_if_empty:NF \l__stex_variables_def_tl {
3381       \stex_annotate_invisible:nnn{definiens}{
3382         {\l__stex_variables_def_tl$}
3383       }
3384     \str_if_empty:NF \l__stex_variables ASSOCTYPE_str {
3385       \stex_annotate_invisible:nnn{ ASSOCTYPE}{\l__stex_variables ASSOCTYPE_str}{
3386     }
3387     \str_if_empty:NF \l__stex_variables REORDER_str {
3388       \stex_annotate_invisible:nnn{ REORDERargs}{\l__stex_variables REORDER_str}{
3389     }

```

```

3390 \int_zero:N \l_tmpa_int
3391 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3392 \tl_clear:N \l_tmpa_tl
3393 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3394   \int_incr:N \l_tmpa_int
3395   \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3396   \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3397   \str_if_eq:VnTF \l_tmpb_str a {
3398     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3399       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3400       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3401     } }
3402   }{
3403     \str_if_eq:VnTF \l_tmpb_str B {
3404       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3405         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3406         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3407       } }
3408     }{
3409       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3410         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3411       } }
3412     }
3413   }
3414 }
3415 \stex_annotate_invisible:nnn { notationcomp }{}{
3416   \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3417   $ \exp_args:Nno \use:nn { \use:c {
3418     stex_var_notation_\l__stex_variables_name_str _cs
3419   } } { \l_tmpa_tl } $
3420 }
3421 \tl_if_empty:NF \l__stex_variables_op_tl {
3422   \stex_annotate_invisible:nnn { notationopcomp }{}{
3423     $\l__stex_variables_op_tl$
3424   }
3425 }
3426 }
3427 \str_if_empty:NF \l__stex_variables_bind_str {
3428   \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3429 }
3430 }\ignorespacesandpars
3431 }
3432
3433 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3434 }
3435
3436 \cs_new:Nn \_stex_reset:N {
3437   \tl_if_exist:NTF #1 {
3438     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3439   }{
3440     \let \exp_not:N #1 \exp_not:N \undefined
3441   }
3442 }
3443

```



```

3444 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3445   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3446   \exp_args:Nnx \use:nn {
3447     % TODO
3448     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3449       #2
3450     }
3451   }{
3452     \stex_reset:N \varnot
3453     \stex_reset:N \vartype
3454     \stex_reset:N \vardefi
3455   }
3456 }
3457
3458 \NewDocumentCommand \vardef { s } {
3459   \IfBooleanTF#1 {
3460     \__stex_variables_do_complex:nn
3461   }{
3462     \__stex_variables_do_simple:nnn
3463   }
3464 }
3465
3466 \NewDocumentCommand \svar { 0{} m }{
3467   \tl_if_empty:nTF {#1}{
3468     \str_set:Nn \l_tmpa_str { #2 }
3469   }{
3470     \str_set:Nn \l_tmpa_str { #1 }
3471   }
3472   \stex_term_omv:nn {
3473     var://\l_tmpa_str
3474   }{
3475     \exp_args:Nnx \use:nn {
3476       \def\comp{\_varcomp}
3477       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3478       \comp{ #2 }
3479     }{
3480       \stex_reset:N \comp
3481       \stex_reset:N \STEXInternalCurrentSymbolStr
3482     }
3483   }
3484 }
3485
3486
3487
3488 \keys_define:nn { stex / varseq } {
3489   name .str_set_x:N = \l__stex_variables_name_str ,
3490   args .int_set:N = \l__stex_variables_args_int ,
3491   type .tl_set:N = \l__stex_variables_type_tl ,
3492   mid .tl_set:N = \l__stex_variables_mid_tl ,
3493   bind .choices:nn =
3494     {forall,exists}
3495     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3496 }
3497

```

```

3498 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3499   \str_clear:N \l__stex_variables_name_str
3500   \int_set:Nn \l__stex_variables_args_int 1
3501   \tl_clear:N \l__stex_variables_type_tl
3502   \str_clear:N \l__stex_variables_bind_str
3503
3504   \keys_set:nn { stex / varseq } { #1 }
3505 }
3506
3507 \NewDocumentCommand \varseq {m O{}} m m m m){
3508   \__stex_variables_seq_args:n { #2 }
3509   \str_if_empty:NT \l__stex_variables_name_str {
3510     \str_set:Nx \l__stex_variables_name_str { #1 }
3511   }
3512   \prop_clear:N \l_tmpa_prop
3513   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3514
3515   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3516   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3517     \msg_error:nnxx{stex}{error/seqlength}
3518     {\int_use:N \l__stex_variables_args_int}
3519     {\seq_count:N \l_tmpa_seq}
3520   }
3521   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3522   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3523     \msg_error:nnxx{stex}{error/seqlength}
3524     {\int_use:N \l__stex_variables_args_int}
3525     {\seq_count:N \l_tmpb_seq}
3526   }
3527   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3528   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3529
3530   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3531   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3532
3533   % argnames
3534
3535   \clist_clear:N \l_tmpa_clist
3536   \int_step_inline:nn {\l__stex_variables_args_int} {
3537     \clist_put_right:Nn \l_tmpa_clist {##1}
3538   }
3539   \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3540
3541
3542
3543
3544   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3545   \int_step_inline:nn \l__stex_variables_args_int {
3546     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3547   }
3548   \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3549   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3550   \tl_if_empty:NF \l__stex_variables_mid_tl {
3551     \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl

```

```

3552     \tl_put_right:Nn \l_tmpa_tl {\,ellipses,}
3553   }
3554   \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}
3555   \int_step_inline:nn \l__stex_variables_args_int {
3556     \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3557   }
3558   \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3559   \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3560
3561
3562   \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3563
3564   \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3565
3566   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}
3567
3568   \int_step_inline:nn \l__stex_variables_args_int {
3569     \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3570       \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3571     }}
3572   }
3573
3574   \tl_set:Nx \l_tmpa_tl {
3575     \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{\l__stex_variables_name_str}{0}{
3576       \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3577     }
3578   }
3579
3580   \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3581
3582   \exp_args:Nno \use:nn {
3583     \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str_cs}
3584     \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3585
3586     \stex_debug:nn{sequences}{New~Sequence:~
3587       \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str_cs\endcsname\\~\\
3588       \prop_to_keyval:N \l_tmpa_prop
3589     }
3590     \prop_set_eq:cN {\l_stex_symdecl_varseq://\l__stex_variables_name_str_prop}\l_tmpa_prop
3591
3592     \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3593       \tl_if_empty:NF \l__stex_variables_type_tl {
3594         \stex_annotate:nnn {type}{\l__stex_variables_type_tl}
3595       }
3596       \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{\l__stex_variables_type_tl}
3597       \str_if_empty:NF \l__stex_variables_bind_str {
3598         \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{\l__stex_variables_type_tl}
3599       }
3600       \stex_annotate:nnn{startindex}{\l__stex_variables_startindex}{\l__stex_variables_type_tl}
3601       \stex_annotate:nnn{endindex}{\l__stex_variables_endindex}{\l__stex_variables_type_tl}
3602     }
3603     \tl_clear:N \l_tmpa_tl
3604     \int_step_inline:nn \l__stex_variables_args_int {
3605       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {

```

```

3606     \stex_annotate:nnn{argmarker}{##1}{-}
3607   } }
3608 }
3609 \stex_annotate_invisible:nnn { notationcomp }{}{
3610   \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3611   $ \exp_args:Nno \use:nn { \use:c {
3612     stex_varseq_\l__stex_variables_name_str _cs
3613   } } { \l_tmpa_tl } $
3614 }
3615 \stex_annotate_invisible:nnn { notationopcomp }{}{
3616   $ \prop_item:Nn \l_tmpa_prop { notation } $
3617 }
3618
3619 }}
3620
3621 \ignorespacesandpars
3622 }
3623
3624 \end{package}

```

Chapter 30

STEX -Terms Implementation

```
3625 <*package>
3626
3627 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3628
3629 <@@=stex_terms>
3630
3631 Warnings and error messages
3632 \msg_new:nnn{stex}{error/nonotation}{
3633   Symbol~#1~invoked,~but~has~no~notation#2!
3634 }
3635 \msg_new:nnn{stex}{error/notationarg}{
3636   Error~in~parsing~notation~#1
3637 }
3638 \msg_new:nnn{stex}{error/noop}{
3639   Symbol~#1~has~no~operator~notation~for~notation~#2
3640 }
3641 \msg_new:nnn{stex}{error/notallowed}{
3642   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3643 }
3644 \msg_new:nnn{stex}{error/doubleargument}{
3645   Argument~#1~of~symbol~#2~already~assigned
3646 }
3647 \msg_new:nnn{stex}{error/overarity}{
3648   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3649 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3649
3650
3651 \bool_new:N \l_stex_allow_semantic_bool
3652 \bool_set_true:N \l_stex_allow_semantic_bool
3653
```

```

3654 \cs_new_protected:Nn \stex_invoke_symbol:n {
3655   \ifvmode\indent\fi
3656   \bool_if:NTF \l_stex_allow_semantic_bool {
3657     \str_if_eq:eeF {
3658       \prop_item:cn {
3659         l_stex_symdecl_#1_prop
3660       }{ deprecate }
3661     }{}{
3662       \msg_warning:nxxx{stex}{warning/deprecated}{
3663         Symbol~#1
3664       }{
3665         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3666       }
3667     }
3668     \if_mode_math:
3669       \exp_after:wN \__stex_terms_invoke_math:n
3670     \else:
3671       \exp_after:wN \__stex_terms_invoke_text:n
3672     \fi: { #1 }
3673   }{
3674     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3675   }
3676 }
3677
3678 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3679   \peek_charcode_remove:NTF ! {
3680     \__stex_terms_invoke_op_custom:nn {#1}
3681   }{
3682     \__stex_terms_invoke_custom:nn {#1}
3683   }
3684 }
3685
3686 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3687   \peek_charcode_remove:NTF ! {
3688     % operator
3689     \peek_charcode_remove:NTF * {
3690       % custom op
3691       \__stex_terms_invoke_op_custom:nn {#1}
3692     }{
3693       % op notation
3694       \peek_charcode:NTF [ {
3695         \__stex_terms_invoke_op_notation:nw {#1}
3696       }{
3697         \__stex_terms_invoke_op_notation:nw {#1}[]
3698       }
3699     }
3700   }{
3701     \peek_charcode_remove:NTF * {
3702       \__stex_terms_invoke_custom:nn {#1}
3703       % custom
3704     }{
3705       % normal
3706       \peek_charcode:NTF [ {
3707         \__stex_terms_invoke_notation:nw {#1}

```

```

3708     }{
3709         \_stex_terms_invoke_notation:nw {#1}[]
3710     }
3711 }
3712 }
3713 }
3714
3715
3716 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3717     \exp_args:Nnx \use:nn {
3718         \def\comp{\_comp}
3719         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3720         \bool_set_false:N \l_stex_allow_semantic_bool
3721         \stex_mathml_intent:nn{#1}{
3722             \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3723                 \comp{ #2 }
3724             }
3725         }
3726     }{
3727         \_stex_reset:N \comp
3728         \_stex_reset:N \STEXInternalCurrentSymbolStr
3729         \bool_set_true:N \l_stex_allow_semantic_bool
3730     }
3731 }
3732
3733 \keys_define:nn { stex / terms } {
3734     % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3735     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3736     unknown .code:n = \str_set:Nx
3737         \l_stex_notation_variant_str \l_keys_key_str
3738 }
3739
3740 \cs_new_protected:Nn \_stex_terms_args:n {
3741     % \str_clear:N \l_stex_notation_lang_str
3742     \str_clear:N \l_stex_notation_variant_str
3743
3744     \keys_set:nn { stex / terms } { #1 }
3745 }
3746
3747 \cs_new_protected:Nn \stex_find_notation:nn {
3748     \_stex_terms_args:n { #2 }
3749     \seq_if_empty:cTF {
3750         l_stex_symdecl_ #1 _notations
3751     } {
3752         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3753     } {
3754         \str_if_empty:NTF \l_stex_notation_variant_str {
3755             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3756         }{
3757             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3758                 \l_stex_notation_variant_str
3759             }{
3760                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3761             }{

```

```

3762         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3763         ~\l_stex_notation_variant_str
3764     }
3765 }
3766 }
3767 }
3768 }
3769
3770 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3771     \exp_args:Nnx \use:nn {
3772         \def\comp{\_comp}
3773         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3774         \stex_find_notation:nn { #1 }{ #2 }
3775         \bool_set_false:N \l_stex_allow_semantic_bool
3776         \cs_if_exist:cTF {
3777             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3778         }{
3779             \_stex_term_oms:nnn { #1 }{
3780                 #1 \c_hash_str \l_stex_notation_variant_str
3781             }{
3782                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3783             }
3784         }{
3785             \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3786                 \cs_if_exist:cTF {
3787                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3788                 }{
3789                     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3790                         \_stex_reset:N \comp
3791                         \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3792                         \_stex_reset:N \STEXInternalCurrentSymbolStr
3793                         \bool_set_true:N \l_stex_allow_semantic_bool
3794                     }
3795                     \def\comp{\_comp}
3796                     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3797                     \bool_set_false:N \l_stex_allow_semantic_bool
3798                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3799                 }{
3800                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3801                     ~\l_stex_notation_variant_str
3802                     }
3803                 }
3804             }{
3805                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3806             }
3807         }
3808     }{
3809         \_stex_reset:N \comp
3810         \_stex_reset:N \STEXInternalCurrentSymbolStr
3811         \bool_set_true:N \l_stex_allow_semantic_bool
3812     }
3813 }
3814
3815 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```



```

3816 \stex_find_notation:nn { #1 }{ #2 }
3817 \cs_if_exist:cTF {
3818   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3819 }{
3820   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3821     \_stex_reset:N \comp
3822     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3823     \_stex_reset:N \STEXInternalCurrentSymbolStr
3824     \bool_set_true:N \l_stex_allow_semantic_bool
3825   }
3826   \def\comp{\_comp}
3827   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3828   \bool_set_false:N \l_stex_allow_semantic_bool
3829   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3830 }{
3831   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3832     ~\l_stex_notation_variant_str
3833   }
3834 }
3835 }
3836
3837 \prop_new:N \l__stex_terms_custom_args_prop
3838 \clist_new:N \l_stex_argnames_seq
3839 \seq_new:N \l__stex_terms_tmp_seq
3840
3841 \cs_new_protected:Nn \__stex_terms_custom_comp:n{ \bool_set_false:N \l_stex_allow_semantic_bo
3842
3843 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3844   \exp_args:Nnx \use:nn {
3845     \def\comp{\__stex_terms_custom_comp:n}
3846     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3847     \prop_clear:N \l__stex_terms_custom_args_prop
3848     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3849     \prop_get:cnN {
3850       l_stex_symdecl_#1 _prop
3851     }{ args } \l_tmpa_str
3852     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
3853       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
3854     }
3855     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3856     \tl_set:Nn \arg { \__stex_terms_arg: }
3857     \str_if_empty:NTF \l_tmpa_str {
3858       \stex_mathml_intent:nn{#1}{
3859         \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3860       }
3861     }{
3862       \seq_clear:N \l__stex_terms_tmp_seq
3863       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
3864         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
3865         \bool_lazy_or:nnT{
3866           \str_if_eq_p:nn{a}{\str_item:Nn \l_tmpa_str{##1}}
3867         }{
3868           \str_if_eq_p:nn{B}{\str_item:Nn \l_tmpa_str{##1}}
3869         }{

```

```

3870         \tl_put_right:Nn \l__stex_terms_tmp_tl +
3871     }
3872     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
3873 }
3874 \stex_mathml_intent:nn{
3875     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}]{ args }(
3876     \seq_use:Nn \l__stex_terms_tmp_seq ,
3877 )
3878 }{
3879     \str_if_in:NnTF \l_tmpa_str b {
3880         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3881     }{
3882         \str_if_in:NnTF \l_tmpa_str B {
3883             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3884         }{
3885             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3886         }
3887     }
3888 }
3889 }
3890 % TODO check that all arguments exist
3891 }{
3892     \_stex_reset:N \l_stex_argnames_seq
3893     \_stex_reset:N \STEXInternalCurrentSymbolStr
3894     \_stex_reset:N \arg
3895     \_stex_reset:N \comp
3896     \_stex_reset:N \l__stex_terms_custom_args_prop
3897     %\bool_set_true:N \l_stex_allow_semantic_bool
3898 }
3899 }
3900
3901 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3902     \tl_if_empty:nTF {#2}{
3903         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3904         \bool_set_true:N \l_tmpa_bool
3905         \bool_do_while:Nn \l_tmpa_bool {
3906             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3907             \int_incr:N \l_tmpa_int
3908         }{
3909             \bool_set_false:N \l_tmpa_bool
3910         }
3911     }
3912     }{
3913         \int_set:Nn \l_tmpa_int { #2 }
3914     }
3915     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3916     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3917         \msg_error:nnxxx{stex}{error/overarity}
3918         {\int_use:N \l_tmpa_int}
3919         {\STEXInternalCurrentSymbolStr}
3920         {\str_count:N \l_tmpa_str}
3921     }
3922     \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3923     \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {

```

```

3924 \bool_lazy_any:nF {
3925   {\str_if_eq_p:Vn \l_tmpa_str {a}}
3926   {\str_if_eq_p:Vn \l_tmpa_str {B}}
3927 }{
3928   \msg_error:nnxx{stex}{error/doubleargument}
3929   {\int_use:N \l_tmpa_int}
3930   {\STEXInternalCurrentSymbolStr}
3931 }
3932 }
3933 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3934 \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
3935   \bool_set_true:N \l_stex_allow_semantic_bool
3936   \use:nn
3937 }
3938 {
3939 \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
3940   \IfBooleanTF#1{
3941     \stex_annotate_invisible:n { %TODO
3942       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3943     }
3944   }{ %TODO
3945     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3946   }
3947 }}
3948 {\bool_set_false:N \l_stex_allow_semantic_bool}
3949 }
3950
3951
3952 \cs_new_protected:Nn \_stex_term_arg:nn {
3953   \bool_set_true:N \l_stex_allow_semantic_bool
3954   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3955   \bool_set_false:N \l_stex_allow_semantic_bool
3956 }
3957
3958 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
3959   \exp_args:Nnx \use:nn
3960   { \int_set:Nn \l__stex_terms_downprec { #2 }
3961     \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
3962       \_stex_term_arg:nn { #1 }{ #3 }
3963     }
3964   }
3965   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3966 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 87.)

`\STEXInternalTermMathAssocArgiiii`

```

3967 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4#5 {
3968   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3969   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
3970   \tl_if_empty:nTF { #3 }{
3971     \STEXInternalTermMathArgiii{#5#1}{#2}{}
3972   }{
3973     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{

```

```

3974 \expandafter\if\expandafter\relax\noexpand#3
3975 \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
3976 \else
3977 \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
3978 \fi
3979 \l_tmpa_tl
3980 }{
3981 \_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
3982 }
3983 }
3984 }
3985
3986 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_maybe_sequence:Nnn {
3987 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3988 \str_if_empty:NTF \l_tmpa_str {
3989 \exp_args:Nx \cs_if_eq:NNTF {
3990 \tl_head:N #1
3991 } \stex_invoke_sequence:n {
3992 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3993 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3994 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
3995 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3996 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3997 \exp_not:n{\exp_args:Nnx \use:nn} {
3998 \exp_not:n {
3999 \def\comp{\_varcomp}
4000 \str_set:Nn \STEXInternalCurrentSymbolStr
4001 } {varseq://\l_tmpa_str}
4002 \exp_not:n{ ##1 }
4003 }{
4004 \exp_not:n {
4005 \_stex_reset:N \comp
4006 \_stex_reset:N \STEXInternalCurrentSymbolStr
4007 }
4008 }
4009 }}}
4010 \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4011 \seq_reverse:N \l_tmpa_seq
4012 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4013 \seq_map_inline:Nn \l_tmpa_seq {
4014 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4015 \exp_args:Nno
4016 \l_tmpa_cs { ##1 } \l_tmpa_tl
4017 }
4018 }
4019 \tl_set:Nx \l_tmpa_tl {
4020 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4021 \exp_args:No \exp_not:n \l_tmpa_tl
4022 }
4023 }
4024 \exp_args:No\l_tmpb_tl\l_tmpa_tl
4025 }{
4026 \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4027 }

```

```

4028 } {
4029   \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4030 }
4031
4032 }
4033
4034 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nnn {
4035   \clist_set:Nn \l_tmpa_clist{ #2 }
4036   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4037     \tl_set:Nn \l_tmpa_tl {
4038       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4039         \_stex_term_arg:nn{A#3#1}{ #2 } }
4040     }
4041   }{
4042     \clist_reverse:N \l_tmpa_clist
4043     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4044     \tl_set:Nx \l_tmpa_tl {
4045       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4046         \_stex_term_arg:nn{A#3#1}{
4047           \exp_args:No \exp_not:n \l_tmpa_tl
4048         }
4049       }
4050     }
4051     \clist_map_inline:Nn \l_tmpa_clist {
4052       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4053         \exp_args:Nno
4054         \l_tmpa_cs {
4055           \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4056             \_stex_term_arg:nn{A#3#1}{##1}
4057           }
4058         } \l_tmpa_tl
4059       }
4060     }
4061     \exp_args:No\l_tmpb_tl\l_tmpa_tl
4062   }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 88.)

30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
4063 \tl_const:Nx \infprec {\int_use:N \c_max_int}
4064 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4065 \int_new:N \l__stex_terms_downprec
4066 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_terms_downprec`. These variables are documented on page 88.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
4067 \tl_set:Nn \l__stex_terms_left_bracket_str (
4068 \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

```

4069 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4070   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4071     \bool_set_false:N \l__stex_terms_brackets_done_bool
4072     #2
4073   } {
4074     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4075       \bool_if:NTF \l__stex_inarray_bool { #2 }{
4076         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4077         \dobrackets { #2 }
4078       }
4079     }{ #2 }
4080   }
4081 }

```

(End definition for _stex_terms_maybe_brackets:nn.)

\dobrackets

```

4082 \bool_new:N \l__stex_terms_brackets_done_bool
4083 %\RequirePackage{scalereel}
4084 \cs_new_protected:Npn \dobrackets #1 {
4085   %\ThisStyle{\if D\m@switch
4086   %   \exp_args:Nnx \use:nn
4087   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4088   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
4089   % \else
4090   \exp_args:Nnx \use:nn
4091   {
4092     \bool_set_true:N \l__stex_terms_brackets_done_bool
4093     \int_set:Nn \l__stex_terms_downprec \infprec
4094     \l__stex_terms_left_bracket_str
4095     #1
4096   }
4097   {
4098     \bool_set_false:N \l__stex_terms_brackets_done_bool
4099     \l__stex_terms_right_bracket_str
4100     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4101   }
4102   %\fi}
4103 }

```

(End definition for \dobrackets. This function is documented on page 88.)

\withbrackets

```

4104 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4105   \exp_args:Nnx \use:nn
4106   {
4107     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4108     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4109     #3
4110   }
4111   {

```

```

4112 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4113 {\l__stex_terms_left_bracket_str}
4114 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4115 {\l__stex_terms_right_bracket_str}
4116 }
4117 }

```

(End definition for `\withbrackets`. This function is documented on page 88.)

`\STEXinvisible`

```

4118 \cs_new_protected:Npn \STEXinvisible #1 {
4119 \stex_annotate_invisible:n { #1 }
4120 }

```

(End definition for `\STEXinvisible`. This function is documented on page 88.)

OMDoc terms:

`\STEXInternalTermMathOMSiiii`

```

4121 \cs_new_protected:Nn \_stex_term_oms:nnn {
4122 \stex_annotate:nnn{ OMID }{ #2 }{
4123 #3
4124 }
4125 }
4126
4127 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4128 \__stex_terms_maybe_brackets:nn { #3 }{
4129 \stex_mathml_intent:nn{#1} {
4130 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4131 }
4132 }
4133 }

```

(End definition for `\STEXInternalTermMathOMSiiii`. This function is documented on page 87.)

`_stex_term_math_omv:nn`

```

4134 \cs_new_protected:Nn \_stex_term_omv:nn {
4135 \stex_annotate:nnn{ OMV }{ #1 }{
4136 #2
4137 }
4138 }

```

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

`\STEXInternalTermMathOMAiiai`

```

4139 \cs_new_protected:Nn \_stex_term_oma:nnn {
4140 \stex_annotate:nnn{ OMA }{ #2 }{
4141 #3
4142 }
4143 }
4144
4145 \cs_new_protected:Npn \STEXInternalTermMathOMAiiai #1#2#3#4 {
4146 \exp_args:Nnx \use:nn {
4147 \seq_clear:N \l__stex_terms_tmp_seq
4148 \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4149 \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {

```

```

4150     \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4151   }
4152   \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4153     \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4154     \bool_lazy_or:nnT{
4155       \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4156     }{
4157       \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4158     }{
4159       \tl_put_right:Nn \l__stex_terms_tmp_tl +
4160     }
4161     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4162   }
4163 }
4164 \__stex_terms_maybe_brackets:nn { #3 }{
4165   \stex_mathml_intent:nn{
4166     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4167       \seq_use:Nn \l__stex_terms_tmp_seq ,
4168     )
4169   }{
4170     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4171   }
4172 }
4173 }{
4174   \_stex_reset:N \l_stex_argnames_seq
4175 }
4176 }

```

(End definition for `\STEXInternalTermMathOMAi`. This function is documented on page 87.)

`\STEXInternalTermMathOMBiiii`

```

4177 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4178   \stex_annotate:nnn{ OMBIND }{ #2 }{
4179     #3
4180   }
4181 }
4182
4183 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
4184   \exp_args:Nnx \use:nn {
4185     \seq_clear:N \l__stex_terms_tmp_seq
4186     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4187       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4188         \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4189       }
4190       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4191         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4192         \bool_lazy_or:nnT{
4193           \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4194         }{
4195           \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4196         }{
4197           \tl_put_right:Nn \l__stex_terms_tmp_tl +
4198         }
4199         \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl

```



```

4200     }
4201   }
4202   \__stex_terms_maybe_brackets:nn { #3 }{
4203     \stex_mathml_intent:nn{
4204       #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4205         \seq_use:Nn \l__stex_terms_tmp_seq ,
4206       )
4207     }{
4208       \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4209     }
4210   }
4211   }{
4212     \stex_reset:N \l_stex_argnames_seq
4213   }
4214 }

```

(End definition for `\STEXInternalTermMathOMBiiii`. This function is documented on page 87.)

`\symref`
`\symname`

```

4215 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4216
4217 \keys_define:nn { stex / symname } {
4218   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
4219   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
4220   root     .tl_set_x:N      = \l__stex_terms_root_tl
4221 }
4222
4223 \cs_new_protected:Nn \stex_symname_args:n {
4224   \tl_clear:N \l__stex_terms_post_tl
4225   \tl_clear:N \l__stex_terms_pre_tl
4226   \tl_clear:N \l__stex_terms_root_str
4227   \keys_set:nn { stex / symname } { #1 }
4228 }
4229
4230 \NewDocumentCommand \symref { m m }{
4231   \let\compemph_uri_prev:\compemph@uri
4232   \let\compemph@uri\symrefemph@uri
4233   \STEXsymbol{#1}!\{ #2 }
4234   \let\compemph@uri\compemph_uri_prev:
4235 }
4236
4237 \NewDocumentCommand \synonym { 0{} m m }{
4238   \stex_symname_args:n { #1 }
4239   \let\compemph_uri_prev:\compemph@uri
4240   \let\compemph@uri\symrefemph@uri
4241   % TODO
4242   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4243   \let\compemph@uri\compemph_uri_prev:
4244 }
4245
4246 \NewDocumentCommand \symname { 0{} m }{
4247   \stex_symname_args:n { #1 }
4248   \stex_get_symbol:n { #2 }
4249   \str_set:Nx \l_tmpa_str {

```

```

4250   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4251 }
4252 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4253
4254 \let\compemph_uri_prev:\compemph@uri
4255 \let\compemph@uri\symrefemph@uri
4256 \exp_args:Nnx \use:nn
4257 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4258   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4259 } }
4260 \let\compemph@uri\compemph_uri_prev:
4261 }
4262
4263 \NewDocumentCommand \Symname { 0{} m }{
4264   \stex_symname_args:n { #1 }
4265   \stex_get_symbol:n { #2 }
4266   \str_set:Nx \l_tmpa_str {
4267     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4268   }
4269   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4270   \let\compemph_uri_prev:\compemph@uri
4271   \let\compemph@uri\symrefemph@uri
4272   \exp_args:Nnx \use:nn
4273   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4274     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4275     \l__stex_terms_post_tl
4276   } }
4277   \let\compemph@uri\compemph_uri_prev:
4278 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 87.)

30.3 Notation Components

```

4279 <@@=stex_notationcomps>

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

4280 \cs_new_protected:Npn \_comp #1 {
4281   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4282     \stex_html_backend:TF {
4283       \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4284     }{
4285       \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4286     }
4287   }
4288 }
4289
4290 \cs_new_protected:Npn \_varcomp #1 {
4291   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4292     \stex_html_backend:TF {
4293       \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4294     }{
4295       \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
4296     }

```

```

4297 }
4298 }
4299
4300 \def\comp{\_comp}
4301
4302 \cs_new_protected:Npn \compemph@uri #1 #2 {
4303   \compemph{ #1 }
4304 }
4305
4306
4307 \cs_new_protected:Npn \compemph #1 {
4308   #1
4309 }
4310
4311 \cs_new_protected:Npn \defemph@uri #1 #2 {
4312   \defemph{#1}
4313 }
4314
4315 \cs_new_protected:Npn \defemph #1 {
4316   \textbf{#1}
4317 }
4318
4319 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4320   \symrefemph{#1}
4321 }
4322
4323 \cs_new_protected:Npn \symrefemph #1 {
4324   \emph{#1}
4325 }
4326
4327 \cs_new_protected:Npn \varemp@uri #1 #2 {
4328   \varemp{#1}
4329 }
4330
4331 \cs_new_protected:Npn \varemp #1 {
4332   #1
4333 }

```

(End definition for `\comp` and others. These functions are documented on page 88.)

`\ellipses`

```

4334 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 88.)

```

\parray
\prmatrix 4335 \bool_new:N \l_stex_inarray_bool
\parrayline 4336 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 4337 \NewDocumentCommand \parray { m m } {
\parraycell 4338   \begingroup
4339   \bool_set_true:N \l_stex_inarray_bool
4340   \begin{array}{#1}
4341     #2
4342   \end{array}
4343   \endgroup

```

```

4344 }
4345
4346 \NewDocumentCommand \prmatrix { m } {
4347   \begingroup
4348   \bool_set_true:N \l_stex_inarray_bool
4349   \begin{matrix}
4350     #1
4351   \end{matrix}
4352   \endgroup
4353 }
4354
4355 \def \maybepline {
4356   \bool_if:NT \l_stex_inarray_bool {\hline}
4357 }
4358
4359 \def \parrayline #1 #2 {
4360   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
4361 }
4362
4363 \def \pmrow #1 { \parrayline{}{ #1 } }
4364
4365 \def \parraylineh #1 #2 {
4366   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
4367 }
4368
4369 \def \parraycell #1 {
4370   #1 \bool_if:NT \l_stex_inarray_bool {&}
4371 }

```

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

30.4 Variables

```

4372 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

4373 \cs_new_protected:Nn \stex_invoke_variable:n {
4374   \if_mode_math:
4375     \exp_after:wN \__stex_variables_invoke_math:n
4376   \else:
4377     \exp_after:wN \__stex_variables_invoke_text:n
4378   \fi: {#1}
4379 }
4380
4381 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4382   \peek_charcode_remove:NTF ! {
4383     \__stex_variables_invoke_op_custom:nn {#1}
4384   }{
4385     \__stex_variables_invoke_custom:nn {#1}
4386   }
4387 }
4388
4389
4390 \cs_new_protected:Nn \__stex_variables_invoke_math:n {

```

```

4391 \peek_charcode_remove:NTF ! {
4392   \peek_charcode_remove:NTF ! {
4393     \peek_charcode:NTF [ {
4394       % TODO throw error
4395     }{
4396       \__stex_variables_invoke_op_custom:nn
4397     }
4398   }{
4399     \__stex_variables_invoke_op:n { #1 }
4400   }
4401 }{
4402   \peek_charcode_remove:NTF * {
4403     \__stex_variables_invoke_custom:nn { #1 }
4404   }{
4405     \__stex_variables_invoke_math_ii:n { #1 }
4406   }
4407 }
4408 }
4409
4410 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4411   \exp_args:Nnx \use:nn {
4412     \def\comp{\_varcomp}
4413     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4414     \bool_set_false:N \l_stex_allow_semantic_bool
4415     \stex_term_omv:nn {var://#1}{
4416       \comp{ #2 }
4417     }
4418   }{
4419     \stex_reset:N \comp
4420     \stex_reset:N \STEXInternalCurrentSymbolStr
4421     \bool_set_true:N \l_stex_allow_semantic_bool
4422   }
4423 }
4424
4425 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4426   \cs_if_exist:cTF {
4427     stex_var_op_notation_ #1 _cs
4428   }{
4429     \exp_args:Nnx \use:nn {
4430       \def\comp{\_varcomp}
4431       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4432       \stex_term_omv:nn { var://#1 }{
4433         \use:c{stex_var_op_notation_ #1 _cs }
4434       }
4435     }{
4436       \stex_reset:N \comp
4437       \stex_reset:N \STEXInternalCurrentSymbolStr
4438     }
4439   }{
4440     \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4441       \__stex_variables_invoke_math_ii:n {#1}
4442     }{
4443       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
4444     }

```

```

4445 }
4446 }
4447
4448 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4449   \cs_if_exist:cTF {
4450     stex_var_notation_#1_cs
4451   }{
4452     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4453       \_stex_reset:N \comp
4454       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4455       \_stex_reset:N \STEXInternalCurrentSymbolStr
4456       \bool_set_true:N \l_stex_allow_semantic_bool
4457     }
4458     \def\comp{\_varcomp}
4459     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4460     \bool_set_false:N \l_stex_allow_semantic_bool
4461     \use:c{stex_var_notation_#1_cs}
4462   }{
4463     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
4464   }
4465 }
4466
4467 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4468   \exp_args:Nnx \use:nn {
4469     \def\comp{\_varcomp}
4470     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4471     \prop_clear:N \l__stex_terms_custom_args_prop
4472     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4473     \prop_get:cnN {
4474       l_stex_symdecl_var://#1 _prop
4475     }{ args } \l_tmpa_str
4476     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4477     \tl_set:Nn \arg { \_stex_terms_arg: }
4478     \str_if_empty:NTF \l_tmpa_str {
4479       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4480     }{
4481       \str_if_in:NnTF \l_tmpa_str b {
4482         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4483       }{
4484         \str_if_in:NnTF \l_tmpa_str B {
4485           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4486         }{
4487           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4488         }
4489       }
4490     }
4491     % TODO check that all arguments exist
4492   }{
4493     \_stex_reset:N \STEXInternalCurrentSymbolStr
4494     \_stex_reset:N \arg
4495     \_stex_reset:N \comp
4496     \_stex_reset:N \l__stex_terms_custom_args_prop
4497     %\bool_set_true:N \l_stex_allow_semantic_bool
4498   }

```

```
4499 }
```

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

30.5 Sequences

```
4500 <@@=stex_sequences>
4501
4502 \cs_new_protected:Nn \stex_invoke_sequence:n {
4503   \peek_charcode_remove:NTF ! {
4504     \stex_term_omv:nn {varseq://#1}{
4505       \exp_args:Nnx \use:nn {
4506         \def\comp{\_varcomp}
4507         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4508         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4509       }{
4510         \stex_reset:N \comp
4511         \stex_reset:N \STEXInternalCurrentSymbolStr
4512       }
4513     }
4514   }{
4515     \bool_set_false:N \l_stex_allow_semantic_bool
4516     \def\comp{\_varcomp}
4517     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4518     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4519       \stex_reset:N \comp
4520       \stex_reset:N \STEXInternalSymbolAfterInvokationTL
4521       \stex_reset:N \STEXInternalCurrentSymbolStr
4522       \bool_set_true:N \l_stex_allow_semantic_bool
4523     }
4524     \use:c { stex_varseq_#1_cs }
4525   }
4526 }
4527 </package>
```

Chapter 31

STEX -Structural Features Implementation

```
4528 ⟨*package⟩
4529
4530 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4531
4532 Warnings and error messages
4533 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4534   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4535 }
4536 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4537   Symbol~#1~not~assigned~in~interpretmodule~#2
4538 }
4539 \msg_new:nnn{stex}{error/unknownstructure}{
4540   No~structure~#1~found!
4541 }
4542
4543 \msg_new:nnn{stex}{error/unknownfield}{
4544   No~field~#1~in~instance~#2~found!\#3
4545 }
4546
4547 \msg_new:nnn{stex}{error/keyval}{
4548   Invalid~key=value~pair:#1
4549 }
4550 \msg_new:nnn{stex}{error/instantiate/missing}{
4551   Assignments~missing~in~instantiate:~#1
4552 }
4553 \msg_new:nnn{stex}{error/incompatible}{
4554   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4555 }
4556
```


31.1 Imports with modification

```

4557 <@@=stex_copymodule>
4558 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4559   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4560     \tl_set:Nn \l_tmpa_tl { #1 }
4561     \__stex_copymodule_get_symbol_from_cs:
4562   }{
4563     % argument is a string
4564     % is it a command name?
4565     \cs_if_exist:cTF { #1 }{
4566       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4567       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4568       \str_if_empty:NNTF \l_tmpa_str {
4569         \exp_args:Nx \cs_if_eq:NNTF {
4570           \tl_head:N \l_tmpa_tl
4571         } \stex_invoke_symbol:n {
4572           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4573         }{
4574           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4575         }
4576       } {
4577         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4578       }
4579     }{
4580       % argument is not a command name
4581       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4582       % \l_stex_all_symbols_seq
4583     }
4584   }
4585 }
4586
4587 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4588   \str_set:Nn \l_tmpa_str { #1 }
4589   \bool_set_false:N \l_tmpa_bool
4590   \bool_if:NF \l_tmpa_bool {
4591     \tl_set:Nn \l_tmpa_tl {
4592       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4593     }
4594     \str_set:Nn \l_tmpa_str { #1 }
4595     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4596     \seq_map_inline:Nn #2 {
4597       \str_set:Nn \l_tmpb_str { ##1 }
4598       \str_if_eq:eeT { \l_tmpa_str } {
4599         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4600       } {
4601         \seq_map_break:n {
4602           \tl_set:Nn \l_tmpa_tl {
4603             \str_set:Nn \l_stex_get_symbol_uri_str {
4604               ##1
4605             }
4606           }
4607         }
4608       }

```

```

4609     }
4610     \l_tmpa_tl
4611   }
4612 }
4613
4614 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4615   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4616     { \tl_tail:N \l_tmpa_tl }
4617   \tl_if_single:NTF \l_tmpa_tl {
4618     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4619       \exp_after:wN \str_set:Nn \exp_after:wN
4620         \l_stex_get_symbol_uri_str \l_tmpa_tl
4621       \__stex_copymodule_get_symbol_check:n { #1 }
4622     }{
4623       % TODO
4624       % tail is not a single group
4625     }
4626   }{
4627     % TODO
4628     % tail is not a single group
4629   }
4630 }
4631
4632 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4633   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4634     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4635       :~\seq_use:Nn #1 {,~}
4636     }
4637   }
4638 }
4639
4640 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4641   % import module
4642   \stex_import_module_uri:nn { #1 } { #2 }
4643   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4644   \stex_import_require_module:nnnn
4645     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4646     { \l_stex_import_path_str } { \l_stex_import_name_str }
4647
4648   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4649   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4650
4651   % fields
4652   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4653   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4654     \seq_map_inline:cn {c_stex_module_###1_constants}{
4655       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4656         ###1 ? #####1
4657       }
4658     }
4659   }
4660
4661   % setup prop
4662   \seq_clear:N \l_tmpa_seq

```

```

4663 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4664   name      = \l_stex_current_copymodule_name_str ,
4665   module    = \l_stex_current_module_str ,
4666   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4667   includes  = \l_tmpa_seq %,
4668 % fields    = \l_tmpa_seq
4669 }
4670 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4671   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4672 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {
4673 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {
4674
4675 \stex_if_do_html:T {
4676   \begin{stex_annotate_env} {#4} {
4677     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4678   }
4679   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{
4680 }
4681 }
4682
4683 \cs_new_protected:Nn \stex_copymodule_end:n {
4684   % apply to every field
4685   \def \l_tmpa_cs ##1 ##2 {#1}
4686
4687   \tl_clear:N \__stex_copymodule_module_tl
4688   \tl_clear:N \__stex_copymodule_exec_tl
4689
4690   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4691   \seq_clear:N \__stex_copymodule_fields_seq
4692
4693   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4694     \seq_map_inline:cn {c_stex_module_##1_constants}{
4695
4696       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4697       \l_tmpa_cs{##1}{####1}
4698
4699       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4700         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
4701         \stex_if_do_html:T {
4702           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4703             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
4704           }
4705         }
4706       }{
4707         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4708       }
4709
4710       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4711       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4712       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4713
4714       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4715         \stex_if_do_html:T {
4716           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4717         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4718     }
4719 }
4720 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4721 }
4722
4723 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4724 \tl_put_right:Nx \__stex_copymodule_module_tl {
4725     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4726     \prop_set_from_keyval:cn {
4727         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4728     }{
4729         \prop_to_keyval:N \l_tmpa_prop
4730     }
4731 }
4732
4733 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4734     \stex_if_do_html:T {
4735         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4736             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4737         }
4738     }
4739     \tl_put_right:Nx \__stex_copymodule_module_tl {
4740         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4741             \stex_invoke_symbol:n {
4742                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4743             }
4744         }
4745     }
4746 }
4747
4748 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4749
4750 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4751     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4752 }
4753
4754 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4755     \stex_if_do_html:TF{
4756         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4757     }{
4758         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4759     }
4760 }
4761 }
4762 }
4763
4764
4765 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4766 \tl_put_left:Nx \__stex_copymodule_module_tl {
4767     \prop_set_from_keyval:cn {
4768         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4769 }{
4770     \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4771   }
4772 }
4773
4774 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4775   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4776 }
4777
4778 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4779 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4780 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4781
4782 \__stex_copymodule_exec_tl
4783 \stex_if_do_html:T {
4784   \end{stex_annotate_env}
4785 }
4786 }
4787
4788 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4789   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4790   \stex_deactivate_macro:Nn \symdecl {module~environments}
4791   \stex_deactivate_macro:Nn \symdef {module~environments}
4792   \stex_deactivate_macro:Nn \notation {module~environments}
4793   \stex_reactivate_macro:N \assign
4794   \stex_reactivate_macro:N \renamedekl
4795   \stex_reactivate_macro:N \donotcopy
4796   \stex_smsmode_do:
4797 }{
4798   \stex_copymodule_end:n {}
4799 }
4800
4801 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4802   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4803   \stex_deactivate_macro:Nn \symdecl {module~environments}
4804   \stex_deactivate_macro:Nn \symdef {module~environments}
4805   \stex_deactivate_macro:Nn \notation {module~environments}
4806   \stex_reactivate_macro:N \assign
4807   \stex_reactivate_macro:N \renamedekl
4808   \stex_reactivate_macro:N \donotcopy
4809   \stex_smsmode_do:
4810 }{
4811   \stex_copymodule_end:n {
4812     \tl_if_exist:cF {
4813       l__stex_copymodule_copymodule_##1?##2_def_tl
4814     }{
4815       \str_if_eq:eeF {
4816         \prop_item:cn{
4817           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4818         }{ true }{
4819           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4820             ##1?##2
4821           }{\l_stex_current_copymodule_name_str}
4822         }
4823       }
4824     }

```

```

4825 }
4826
4827 \iffalse \begin{stex_annotate_env} \fi
4828 \NewDocumentEnvironment {realization} { 0 } { m } {
4829   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4830   \stex_deactivate_macro:Nn \symdecl {module~environments}
4831   \stex_deactivate_macro:Nn \symdef {module~environments}
4832   \stex_deactivate_macro:Nn \notation {module~environments}
4833   \stex_reactivate_macro:N \donotcopy
4834   \stex_reactivate_macro:N \assign
4835   \stex_smsmode_do:
4836 } {
4837   \stex_import_module_uri:nn { #1 } { #2 }
4838   \tl_clear:N \__stex_copymodule_exec_tl
4839   \tl_set:Nx \__stex_copymodule_module_tl {
4840     \stex_import_require_module:nnnn
4841     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4842     { \l_stex_import_path_str } { \l_stex_import_name_str }
4843   }
4844
4845   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4846     \seq_map_inline:cn {c_stex_module_##1_constants}{
4847       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4848       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4849         \stex_if_do_html:T {
4850           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4851             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4852               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4853             }
4854           }
4855         }
4856         \tl_put_right:Nx \__stex_copymodule_module_tl {
4857           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4858         }
4859       }
4860     }
4861
4862     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4863
4864     \__stex_copymodule_exec_tl
4865     \stex_if_do_html:T {\end{stex_annotate_env}}
4866   }
4867
4868 \NewDocumentCommand \donotcopy { m } {
4869   \str_clear:N \l_stex_import_name_str
4870   \str_set:Nn \l_tmpa_str { #1 }
4871   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4872   \seq_map_inline:Nn \l_stex_all_modules_seq {
4873     \str_set:Nn \l_tmpb_str { ##1 }
4874     \str_if_eq:eeT { \l_tmpa_str } {
4875       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4876     } {
4877       \seq_map_break:n {
4878         \stex_if_do_html:T {

```

```

4879         \stex_if_smsmode:F {
4880             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4881                 \stex_annotate:nnn{domain}{##1}{}}
4882             }
4883         }
4884     }
4885     \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4886 }
4887 }
4888 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4889     \str_set:Nn \l_tmpb_str { #####1 }
4890     \str_if_eq:eeT { \l_tmpa_str } {
4891         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4892     } {
4893         \seq_map_break:n {\seq_map_break:n {
4894             \stex_if_do_html:T {
4895                 \stex_if_smsmode:F {
4896                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4897                         \stex_annotate:nnn{domain}{
4898                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4899                         }{}
4900                     }
4901                 }
4902             }
4903             \str_set:Nx \l_stex_import_name_str {
4904                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4905             }
4906         }}
4907     }
4908 }
4909 }
4910 \str_if_empty:NTF \l_stex_import_name_str {
4911     % TODO throw error
4912 }{
4913     \stex_collect_imports:n {\l_stex_import_name_str }
4914     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4915         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4916         \seq_map_inline:cn {c_stex_module_###1_constants}{
4917             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4918             \bool_lazy_any:nT {
4919                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4920                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4921                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4922             }{
4923                 % TODO throw error
4924             }
4925         }
4926     }
4927     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4928     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4929     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4930 }
4931 \stex_smsmode_do:
4932 }

```

```

4933
4934 \NewDocumentCommand \assign { m m }{
4935   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4936   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4937   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4938   \stex_smsmode_do:
4939 }
4940
4941 \keys_define:nn { stex / renamedekl } {
4942   name          .str_set_x:N = \l_stex_renamedekl_name_str
4943 }
4944 \cs_new_protected:Nn \__stex_copymodule_renamedekl_args:n {
4945   \str_clear:N \l_stex_renamedekl_name_str
4946   \keys_set:nn { stex / renamedekl } { #1 }
4947 }
4948
4949 \NewDocumentCommand \renamedekl { O{} m m }{
4950   \__stex_copymodule_renamedekl_args:n { #1 }
4951   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4952   \stex_debug:nn{renamedekl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4953   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4954   \str_if_empty:NTF \l_stex_renamedekl_name_str {
4955     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4956       \l_stex_get_symbol_uri_str
4957     } }
4958   } {
4959     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4960       \stex_debug:nn{renamedekl}{@~\l_stex_current_module_str ? \l_stex_renamedekl_name_str}
4961       \prop_set_eq:cc {l_stex_symdecl_
4962         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4963       }_prop
4964       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4965       \seq_set_eq:cc {l_stex_symdecl_
4966         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4967       }_notations
4968       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4969       \prop_put:cnx {l_stex_symdecl_
4970         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4971       }_prop
4972       }{ name }{ \l_stex_renamedekl_name_str }
4973       \prop_put:cnx {l_stex_symdecl_
4974         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4975       }_prop
4976       }{ module }{ \l_stex_current_module_str }
4977       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4978         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4979       }
4980       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4981         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4982       } }
4983     }
4984     \stex_smsmode_do:
4985   }
4986

```



```

4987 \stex_deactivate_macro:Nn \assign {copymodules}
4988 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4989 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4990
4991

```

31.2 The feature environment

`structural@feature (env.)`

```

4992 <@@=stex_features>
4993
4994 \NewDocumentEnvironment{structural_feature_module}{m m m}{
4995   \stex_if_in_module:F {
4996     \msg_set:nnn{stex}{error/nomodule}{
4997       Structural~Feature~has~to~occur~in~a~module:\\
4998       Feature~#2~of~type~#1\\
4999       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
5000     }
5001     \msg_error:nn{stex}{error/nomodule}
5002   }
5003
5004   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
5005
5006   \stex_module_setup:nn{meta=NONE}{#2 - #1}
5007
5008   \stex_if_do_html:T {
5009     \begin{stex_annotate_env}{feature:#1}{\l_stex_feature_parent_str ? #2 - #1}
5010     \stex_annotate_invisible:nnn{header}{\l_stex_feature_parent_str ? #2 - #1}
5011   }
5012 }{
5013   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5014   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5015   \stex_debug:nn{features}{
5016     Feature: \l_stex_last_feature_str
5017   }
5018   \stex_if_do_html:T {
5019     \end{stex_annotate_env}
5020   }
5021 }

```

31.3 Structure

`structure (env.)`

```

5022 <@@=stex_structures>
5023 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5024   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
5025     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
5026   }
5027   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
5028   {#1}{#2}
5029 }
5030

```

```

5031 \keys_define:nn { stex / features / structure } {
5032   name          .str_set_x:N = \l__stex_structures_name_str ,
5033 }
5034
5035 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5036   \str_clear:N \l__stex_structures_name_str
5037   \keys_set:nn { stex / features / structure } { #1 }
5038 }
5039
5040 \NewDocumentEnvironment{mathstructure}{m O{}}{
5041   \__stex_structures_structure_args:n { #2 }
5042   \str_if_empty:NT \l__stex_structures_name_str {
5043     \str_set:Nx \l__stex_structures_name_str { #1 }
5044   }
5045   \stex_suppress_html:n {
5046     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5047     \exp_args:Nx \stex_symdecl_do:nn {
5048       name = \l__stex_structures_name_str ,
5049       def = {\STEXsymbol{module-type}}{
5050         \STEXInternalTermMathOMSiiii {
5051           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5052             { ns } ?
5053           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5054             { name } / \l__stex_structures_name_str - structure
5055         }{}{0}{}
5056       }}
5057     }{ #1 }
5058   }
5059   \exp_args:Nnnx
5060   \begin{structural_feature_module}{ structure }
5061     { \l__stex_structures_name_str }{}
5062   \stex_smsmode_do:
5063 }{
5064   \end{structural_feature_module}
5065   \stex_reset_up_to_module:n \l_stex_last_feature_str
5066   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5067   \seq_clear:N \l_tmpa_seq
5068   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5069     \seq_map_inline:cn{c_stex_module_##1_constants}{
5070       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5071     }
5072   }
5073   \exp_args:Nnno
5074   \prop_gput:cn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5075   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5076   \stex_add_structure_to_current_module:nn
5077     \l__stex_structures_name_str
5078     \l_stex_last_feature_str
5079
5080   \stex_execute_in_module:x {
5081     \tl_set:cn { #1 }{
5082       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
5083     }
5084   }

```

```

5085 }
5086
5087 \cs_new:Nn \stex_invoke_structure:nn {
5088   \stex_invoke_symbol:n { #1?#2 }
5089 }
5090
5091 \cs_new_protected:Nn \stex_get_structure:n {
5092   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5093     \tl_set:Nn \l_tmpa_tl { #1 }
5094     \__stex_structures_get_from_cs:
5095   }{
5096     \cs_if_exist:cTF { #1 }{
5097       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5098       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5099       \str_if_empty:NNTF \l_tmpa_str {
5100         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
5101           \__stex_structures_get_from_cs:
5102         }{
5103           \__stex_structures_get_from_string:n { #1 }
5104         }
5105       }{
5106         \__stex_structures_get_from_string:n { #1 }
5107       }
5108     }{
5109       \__stex_structures_get_from_string:n { #1 }
5110     }
5111   }
5112 }
5113
5114 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5115   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5116     { \tl_tail:N \l_tmpa_tl }
5117   \str_set:Nx \l_tmpa_str {
5118     \exp_after:wN \use_i:nn \l_tmpa_tl
5119   }
5120   \str_set:Nx \l_tmpb_str {
5121     \exp_after:wN \use_ii:nn \l_tmpa_tl
5122   }
5123   \str_set:Nx \l_stex_get_structure_str {
5124     \l_tmpa_str ? \l_tmpb_str
5125   }
5126   \str_set:Nx \l_stex_get_structure_module_str {
5127     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5128   }
5129 }
5130
5131 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5132   \tl_set:Nn \l_tmpa_tl {
5133     \msg_error:nnn{stex}{error/unknownstructure}{#1}
5134   }
5135   \str_set:Nn \l_tmpa_str { #1 }
5136   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5137
5138   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

5139 \prop_if_exist:cT {c_stex_module_##1_structures} {
5140 \prop_map_inline:cn {c_stex_module_##1_structures} {
5141 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5142 \prop_map_break:n{\seq_map_break:n{
5143 \tl_set:Nn \l_tmpa_tl {
5144 \str_set:Nn \l_stex_get_structure_str {##1?####1}
5145 \str_set:Nn \l_stex_get_structure_module_str {####2}
5146 }
5147 }}
5148 }
5149 }
5150 }
5151 }
5152 \l_tmpa_tl
5153 }

```

\instantiate

```

5154
5155 \keys_define:nn { stex / instantiate } {
5156 name .str_set_x:N = \l__stex_structures_name_str
5157 }
5158 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5159 \str_clear:N \l__stex_structures_name_str
5160 \keys_set:nn { stex / instantiate } { #1 }
5161 }
5162
5163 \NewDocumentCommand \instantiate {m O{} m m O{}}{
5164 \beginingroup
5165 \stex_get_structure:n {#3}
5166 \__stex_structures_instantiate_args:n { #2 }
5167 \str_if_empty:NT \l__stex_structures_name_str {
5168 \str_set:Nn \l__stex_structures_name_str { #1 }
5169 }
5170 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5171 \seq_clear:N \l__stex_structures_fields_seq
5172 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5173 \seq_map_inline:Nn \l_stex_collect_imports_seq {
5174 \seq_map_inline:cn {c_stex_module_##1_constants}{
5175 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5176 }
5177 }
5178
5179 \tl_if_empty:nF{#5}{
5180 \seq_set_split:Nnn \l_tmpa_seq , {#5}
5181 \prop_clear:N \l_tmpa_prop
5182 \seq_map_inline:Nn \l_tmpa_seq {
5183 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5184 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5185 \msg_error:nnn{stex}{error/keyval}{##1}
5186 }
5187 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5188 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5189 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5190 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```

```

5191     \exp_args:Nxx \str_if_eq:nnF
5192     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5193     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5194     \msg_error:nnxxxx{stex}{error/incompatible}
5195     {\l__stex_structures_dom_str}
5196     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5197     {\l_stex_get_symbol_uri_str}
5198     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5199   }
5200   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
5201 }
5202 }
5203
5204 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5205   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5206   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5207
5208   \stex_add_constant_to_current_module:n {\l_tmpa_str}
5209   \stex_execute_in_module:x {
5210     \prop_set_from_keyval:cn { l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _p
5211     name   = \l_tmpa_str ,
5212     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5213     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5214     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5215     argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5216   }
5217   \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5218 }
5219
5220 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5221   \stex_find_notation:nn{##1}{}
5222   \stex_execute_in_module:x {
5223     \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
5224   }
5225
5226   \stex_copy_control_sequence_ii:ccN
5227   {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5228   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5229   \l_tmpa_tl
5230   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5231
5232
5233   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5234     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5235     \stex_execute_in_module:x {
5236       \tl_set:cn
5237       {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
5238       { \exp_args:No \exp_not:n \l_tmpa_cs}
5239     }
5240   }
5241
5242 }
5243
5244 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur

```

```

5245 }
5246
5247 \stex_execute_in_module:x {
5248   \prop_set_from_keyval:cn {l_stex_instance_ \l_stex_current_module_str? \l__stex_structur
5249     domain = \l_stex_get_structure_module_str ,
5250     \prop_to_keyval:N \l_tmpa_prop
5251   }
5252   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str? \l__stex_structur
5253 }
5254 \stex_debug:nn{instantiate}{
5255   Instance~\l_stex_current_module_str? \l__stex_structures_name_str \
5256   \prop_to_keyval:N \l_tmpa_prop
5257 }
5258 \exp_args:Nxx \stex_symdecl_do:nn {
5259   type={\STEXsymbol{module-type}{
5260     \STEXInternalTermMathOMSiiii {
5261       \l_stex_get_structure_module_str
5262     }{}{0}{}}
5263   }}
5264 }{\l__stex_structures_name_str}
5265 % {
5266   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str? \l__stex_structures
5267   \tl_set:Nn \l_stex_notation_after_do_tl {\_stex_notation_final:}
5268   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
5269 % }
5270 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
5271 \endgroup
5272 \stex_smsmode_do:\ignorespacesandpars
5273 }
5274
5275 \cs_new_protected:Nn \stex_symbol_or_var:n {
5276   \cs_if_exist:cTF{#1}{
5277     \cs_set_eq:Nc \l_tmpa_tl { #1 }
5278     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
5279     \str_if_empty:NTF \l_tmpa_str {
5280       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
5281       \stex_invoke_variable:n {
5282         \bool_set_true:N \l_stex_symbol_or_var_bool
5283         \bool_set_false:N \l_stex_instance_or_symbol_bool
5284         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5285         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5286         \str_set:Nx \l_stex_get_symbol_uri_str {
5287           \exp_after:wN \use:n \l_tmpa_tl
5288         }
5289       }{ % TODO \stex_invoke_varinstance:n
5290         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
5291           \bool_set_true:N \l_stex_symbol_or_var_bool
5292           \bool_set_true:N \l_stex_instance_or_symbol_bool
5293           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5294           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5295           \str_set:Nx \l_stex_get_symbol_uri_str {
5296             \exp_after:wN \use:n \l_tmpa_tl
5297           }
5298         }{

```

```

5299         \bool_set_false:N \l_stex_symbol_or_var_bool
5300         \stex_get_symbol:n{#1}
5301     }
5302 }
5303 }{
5304     \__stex_structures_symbolorvar_from_string:n{ #1 }
5305 }
5306 }{
5307     \__stex_structures_symbolorvar_from_string:n{ #1 }
5308 }
5309 }
5310
5311 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
5312     \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5313         \bool_set_true:N \l_stex_symbol_or_var_bool
5314         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5315     }{
5316         \bool_set_false:N \l_stex_symbol_or_var_bool
5317         \stex_get_symbol:n{#1}
5318     }
5319 }
5320
5321 \keys_define:nn { stex / varinstantiate } {
5322     name .str_set_x:N = \l__stex_structures_name_str,
5323     bind .choices:nn =
5324         {forall,exists}
5325         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5326 }
5327 }
5328 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5329     \str_clear:N \l__stex_structures_name_str
5330     \str_clear:N \l__stex_structures_bind_str
5331     \keys_set:nn { stex / varinstantiate } { #1 }
5332 }
5333
5334 \NewDocumentCommand \varinstantiate {m O{}} m m O{}{}{
5335     \beginingroup
5336         \stex_get_structure:n {#3}
5337         \__stex_structures_varinstantiate_args:n { #2 }
5338         \str_if_empty:NT \l__stex_structures_name_str {
5339             \str_set:Nn \l__stex_structures_name_str { #1 }
5340         }
5341         \stex_if_do_html:TF{
5342             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5343         }{\use:n}
5344         {
5345             \stex_if_do_html:T{
5346                 \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
5347             }
5348             \seq_clear:N \l__stex_structures_fields_seq
5349             \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5350             \seq_map_inline:Nn \l_stex_collect_imports_seq {
5351                 \seq_map_inline:cn {c_stex_module_##1_constants}{
5352                     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }

```

```

5353     }
5354 }
5355 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5356 \prop_clear:N \l_tmpa_prop
5357 \tl_if_empty:nF {#5} {
5358   \seq_set_split:Nnn \l_tmpa_seq , {#5}
5359   \seq_map_inline:Nn \l_tmpa_seq {
5360     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5361     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5362       \msg_error:nnn{stex}{error/keyval}{##1}
5363     }
5364     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
5365     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5366     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq
5367     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5368     \stex_if_do_html:T{
5369       \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5370         \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\l_stex_get_symbol_uri_str}}{ }
5371     }
5372     \bool_if:NTF \l_stex_symbol_or_var_bool {
5373       \exp_args:Nxx \str_if_eq:nnF
5374         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5375         {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}{
5376         \msg_error:nnxxx{stex}{error/incompatible}
5377         {\l__stex_structures_dom_str}
5378         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5379         {\l_stex_get_symbol_uri_str}
5380         {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}
5381       }
5382       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
5383     }}{
5384       \exp_args:Nxx \str_if_eq:nnF
5385         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5386         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5387         \msg_error:nnxxx{stex}{error/incompatible}
5388         {\l__stex_structures_dom_str}
5389         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5390         {\l_stex_get_symbol_uri_str}
5391         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5392       }
5393       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
5394     }}
5395     }
5396   }
5397   \tl_gclear:N \g__stex_structures_aftergroup_tl
5398   \seq_map_inline:Nn \l__stex_structures_fields_seq {
5399     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_
5400     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5401     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5402       \stex_find_notation:nn{##1}{ }
5403       \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
5404         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5405       \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
5406       \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{

```



```

5407         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5408         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
5409         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5410     }
5411 }
5412
5413 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5414     \prop_set_from_keyval:cn { \l_stex_symdecl_ var://\l_tmpa_str_prop}{
5415         name = \l_tmpa_str ,
5416         args = \prop_item:cn { \l_stex_symdecl_ ##1_prop } {args} ,
5417         arity = \prop_item:cn { \l_stex_symdecl_ ##1_prop } {arity} ,
5418         assocs = \prop_item:cn { \l_stex_symdecl_ ##1_prop } {assocs} ,
5419         argnames = { \prop_item:cn { \l_stex_symdecl_ ##1_prop } {argnames} } ,
5420     }
5421     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
5422     {g__stex_structures_tmpa_\l_tmpa_str_cs}
5423     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
5424     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5425 }
5426 \prop_put:Nxx \l_tmpa_prop { \prop_item:cn { \l_stex_symdecl_ ##1_prop } {name} } { \stex_inv
5427 }
5428 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5429     \prop_set_from_keyval:cn { \l_stex_varinstance_\l__stex_structures_name_str_prop } {
5430         domain = \l_stex_get_structure_module_str ,
5431         \prop_to_keyval:N \l_tmpa_prop
5432     }
5433     \tl_set:cn { #1 } { \stex_invoke_varinstance:n { \l__stex_structures_name_str } }
5434     \tl_set:cn { \l_stex_varinstance_\l__stex_structures_name_str_op_tl } {
5435         \exp_args:NNx \exp_not:N \use:nn {
5436             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5437             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5438                 \exp_not:n{
5439                     \_varcomp{#4}
5440                 }
5441             }
5442         } {
5443             \exp_not:n{ \stex_reset:N \STEXInternalCurrentSymbolStr }
5444         }
5445     }
5446 }
5447 }
5448 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{ \g__stex_structures_a
5449 \aftergroup \g__stex_structures_aftergroup_tl
5450 \endgroup
5451 \stex_smsmode_do:\ignorespacesandpars
5452 }
5453
5454 \cs_new_protected:Nn \stex_invoke_instance:n {
5455     \peek_charcode_remove:NTF ! {
5456         \stex_invoke_symbol:n{#1}
5457     } {
5458         \stex_invoke_instance:nn {#1}
5459     }
5460 }

```

```

5461
5462
5463 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5464   \peek_charcode_remove:NTF ! {
5465     \exp_args:Nnx \use:nn {
5466       \def\comp{\_varcomp}
5467       \use:c{l_stex_varinstance_#1_op_tl}
5468     }{
5469       \_stex_reset:N \comp
5470     }
5471   }{
5472     \_stex_invoke_varinstance:nn {#1}
5473   }
5474 }
5475
5476 \cs_new_protected:Nn \stex_invoke_instance:nn {
5477   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5478     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5479   }{
5480     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5481     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5482       \prop_to_keyval:N \l_tmpa_prop
5483     }
5484   }
5485 }
5486
5487 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
5488   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5489     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5490     \l_tmpa_tl
5491   }{
5492     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
5493     }
5494   }

```

(End definition for \instantiate. This function is documented on page 34.)

\stex_invoke_structure:nnn

```

5495 % #1: URI of the instance
5496 % #2: URI of the instantiated module
5497 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5498   \tl_if_empty:nTF{ #3 }{
5499     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5500       c_stex_feature_ #2 _prop
5501     }
5502     \tl_clear:N \l_tmpa_tl
5503     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5504     \seq_map_inline:Nn \l_tmpa_seq {
5505       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5506       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5507       \cs_if_exist:cT {
5508         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5509       }{
5510         \tl_if_empty:NF \l_tmpa_tl {

```

```

5511         \tl_put_right:Nn \l_tmpa_tl {,}
5512     }
5513     \tl_put_right:Nx \l_tmpa_tl {
5514         \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5515     }
5516 }
5517 }
5518 \exp_args:No \mathstruct \l_tmpa_tl
5519 }{
5520     \stex_invoke_symbol:n{#1/#3}
5521 }
5522 }

```

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

```

5523 </package>

```

Chapter 32

STEX -Statements Implementation

```
5524 <*package>
5525
5526 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5527
5528 <@@=stex_statements>
    Warnings and error messages
5529

\titleemph
5530 \def\titleemph#1{\textbf{#1}}

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

32.1 Definitions

definiendum

```
5531 \keys_define:nn {stex / definiendum }{
5532   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5533   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5534   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5535   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5536 }
5537 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5538   \str_clear:N \l__stex_statements_definiendum_root_str
5539   \tl_clear:N \l__stex_statements_definiendum_post_tl
5540   \str_clear:N \l__stex_statements_definiendum_gfa_str
5541   \keys_set:nn { stex / definiendum }{ #1 }
5542 }
5543 \NewDocumentCommand \definiendum { O{} m m } {
5544   \__stex_statements_definiendum_args:n { #1 }
5545   \stex_get_symbol:n { #2 }
5546   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5547   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5548     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5549     \tl_set:Nn \l_tmpa_tl { #3 }
5550   } {
5551     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5552     \tl_set:Nn \l_tmpa_tl {
5553       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5554     }
5555   }
5556 } {
5557   \tl_set:Nn \l_tmpa_tl { #3 }
5558 }
5559
5560 % TODO root
5561 \stex_html_backend:TF {
5562   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5563 } {
5564   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5565 }
5566 }
5567 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 44.)

definame

```

5568
5569 \NewDocumentCommand \definame { 0{ } m } {
5570   \__stex_statements_definiendum_args:n { #1 }
5571   % TODO: root
5572   \stex_get_symbol:n { #2 }
5573   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5574   \str_set:Nx \l_tmpa_str {
5575     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5576   }
5577   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5578   \stex_html_backend:TF {
5579     \stex_if_do_html:T {
5580       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5581         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5582       }
5583     }
5584   } {
5585     \exp_args:Nnx \defemph@uri {
5586       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5587     } { \l_stex_get_symbol_uri_str }
5588   }
5589 }
5590 \stex_deactivate_macro:Nn \definame {definition~environments}
5591
5592 \NewDocumentCommand \Definame { 0{ } m } {
5593   \__stex_statements_definiendum_args:n { #1 }
5594   \stex_get_symbol:n { #2 }
5595   \str_set:Nx \l_tmpa_str {
5596     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5597   }
5598   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5599 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5600 \stex_html_backend:TF {
5601   \stex_if_do_html:T {
5602     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5603       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5604     }
5605   }
5606 } {
5607   \exp_args:Nnx \defemph@uri {
5608     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5609   } { \l_stex_get_symbol_uri_str }
5610 }
5611 }
5612 \stex_deactivate_macro:Nn \Definame {definition-environments}
5613
5614 \NewDocumentCommand \premise { m }{
5615   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5616 }
5617 \NewDocumentCommand \conclusion { m }{
5618   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5619 }
5620 \NewDocumentCommand \definiens { 0{} m }{
5621   \str_clear:N \l_stex_get_symbol_uri_str
5622   \tl_if_empty:nF {#1} {
5623     \stex_get_symbol:n { #1 }
5624   }
5625   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5626     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5627       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5628     }{
5629       % TODO throw error
5630     }
5631   }
5632   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5633   {\l_stex_current_module_str}{
5634     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5635   }{true}{
5636     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5637     \exp_args:Nx \stex_add_to_current_module:n {
5638       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5639     }
5640   }
5641 }
5642 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5643 }
5644
5645 \NewDocumentCommand \varbindforall {m}{
5646   \stex_symbol_or_var:n {#1}
5647   \bool_if:NTF\l_stex_symbol_or_var_bool{
5648     \stex_if_do_html:T {
5649       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5650     }
5651   }{
5652     % todo throw error

```

```

5653 }
5654 }
5655
5656 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5657 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5658 \stex_deactivate_macro:Nn \definiens {definition~environments}
5659 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5660

```

(End definition for definame. This function is documented on page 44.)

sdefinition (env.)

```

5661
5662 \keys_define:nn {stex / sdefinition }{
5663   type      .str_set_x:N = \sdefinitiontype,
5664   id        .str_set_x:N = \sdefinitionid,
5665   name      .str_set_x:N = \sdefinitionname,
5666   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5667   title     .tl_set:N    = \sdefinitiontitle
5668 }
5669 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5670   \str_clear:N \sdefinitiontype
5671   \str_clear:N \sdefinitionid
5672   \str_clear:N \sdefinitionname
5673   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5674   \tl_clear:N \sdefinitiontitle
5675   \keys_set:nn { stex / sdefinition }{ #1 }
5676 }
5677
5678 \NewDocumentEnvironment{sdefinition}{0{}}{
5679   \__stex_statements_sdefinition_args:n{ #1 }
5680   \stex_reactivate_macro:N \definiendum
5681   \stex_reactivate_macro:N \definame
5682   \stex_reactivate_macro:N \Definame
5683   \stex_reactivate_macro:N \premise
5684   \stex_reactivate_macro:N \definiens
5685   \stex_reactivate_macro:N \varbindforall
5686   \stex_if_smsmode:F{
5687     \seq_clear:N \l_tmpb_seq
5688     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5689       \tl_if_empty:nF{ ##1 }{
5690         \stex_get_symbol:n { ##1 }
5691         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5692           \l_stex_get_symbol_uri_str
5693         }
5694       }
5695     }
5696     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5697     \exp_args:Nnnx
5698     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5699     \str_if_empty:NF \sdefinitiontype {
5700       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5701     }
5702     \str_if_empty:NF \sdefinitionname {

```

```

5703     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5704   }
5705   \clist_set:No \l_tmpa_clist \sdefinitiontype
5706   \tl_clear:N \l_tmpa_tl
5707   \clist_map_inline:Nn \l_tmpa_clist {
5708     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5709       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5710     }
5711   }
5712   \tl_if_empty:NTF \l_tmpa_tl {
5713     \__stex_statements_sdefinition_start:
5714   }{
5715     \l_tmpa_tl
5716   }
5717 }
5718 \stex_ref_new_doc_target:n \sdefinitionid
5719 \stex_smsmode_do:
5720 }{
5721   \stex_suppress_html:n {
5722     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5723   }
5724   \stex_if_smsmode:F {
5725     \clist_set:No \l_tmpa_clist \sdefinitiontype
5726     \tl_clear:N \l_tmpa_tl
5727     \clist_map_inline:Nn \l_tmpa_clist {
5728       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5729         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5730       }
5731     }
5732     \tl_if_empty:NTF \l_tmpa_tl {
5733       \__stex_statements_sdefinition_end:
5734     }{
5735       \l_tmpa_tl
5736     }
5737     \end{stex_annotate_env}
5738   }
5739 }

```

\stexpatchdefinition

```

5740 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5741   \stex_par:\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
5742     ~(\sdefinitiontitle)
5743   }~}
5744 }
5745 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5746
5747 \newcommand\stexpatchdefinition[3]{} {
5748   \str_set:Nx \l_tmpa_str{ #1 }
5749   \str_if_empty:NTF \l_tmpa_str {
5750     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5751     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5752   }{
5753     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5754     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```



```

5755     }
5756 }

```

(End definition for `\stexpatchdefinition`. This function is documented on page 51.)

`\inlinedef` inline:

```

5757 \keys_define:nn {stex / inlinedef }{
5758   type      .str_set_x:N = \sdefinitiontype,
5759   id        .str_set_x:N = \sdefinitionid,
5760   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5761   name      .str_set_x:N = \sdefinitionname
5762 }
5763 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5764   \str_clear:N \sdefinitiontype
5765   \str_clear:N \sdefinitionid
5766   \str_clear:N \sdefinitionname
5767   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5768   \keys_set:nn { stex / inlinedef }{ #1 }
5769 }
5770 \NewDocumentCommand \inlinedef { 0{} m } {
5771   \begingroup
5772   \__stex_statements_inlinedef_args:n{ #1 }
5773   \stex_reactivate_macro:N \definiendum
5774   \stex_reactivate_macro:N \definame
5775   \stex_reactivate_macro:N \Definame
5776   \stex_reactivate_macro:N \premise
5777   \stex_reactivate_macro:N \definiens
5778   \stex_reactivate_macro:N \varbindforall
5779   \stex_ref_new_doc_target:n \sdefinitionid
5780   \stex_if_smsmode:TF{\stex_suppress_html:n {
5781     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5782   }}{
5783     \seq_clear:N \l_tmpb_seq
5784     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5785       \tl_if_empty:nF{ ##1 }{
5786         \stex_get_symbol:n { ##1 }
5787         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5788           \l_stex_get_symbol_uri_str
5789         }
5790       }
5791     }
5792     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5793     \exp_args:Nnx
5794     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5795       \str_if_empty:NF \sdefinitiontype {
5796         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5797       }
5798       #2
5799       \str_if_empty:NF \sdefinitionname {
5800         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5801         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5802       }
5803     }
5804   }

```



```

5852 \clist_map_inline:Nn \l_tmpa_clist {
5853   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5854     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5855   }
5856 }
5857 \tl_if_empty:NTF \l_tmpa_tl {
5858   \__stex_statements_sassertion_start:
5859 }{
5860   \l_tmpa_tl
5861 }
5862 }
5863 \str_if_empty:NTF \sassertionid {
5864   \str_if_empty:NF \sassertionname {
5865     \stex_ref_new_doc_target:n {}
5866   }
5867 } {
5868   \stex_ref_new_doc_target:n \sassertionid
5869 }
5870 \stex_smsmode_do:
5871 ){
5872   \str_if_empty:NF \sassertionname {
5873     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5874     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5875   }
5876   \stex_if_smsmode:F {
5877     \clist_set:No \l_tmpa_clist \sassertiontype
5878     \tl_clear:N \l_tmpa_tl
5879     \clist_map_inline:Nn \l_tmpa_clist {
5880       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5881         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5882       }
5883     }
5884     \tl_if_empty:NTF \l_tmpa_tl {
5885       \__stex_statements_sassertion_end:
5886     }{
5887       \l_tmpa_tl
5888     }
5889     \end{stex_annotate_env}
5890   }
5891 }

```

\stexpatchassertion

```

5892
5893 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5894   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5895     (\sassertiontitle)
5896   }~}
5897 }
5898 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5899
5900 \newcommand\stexpatchassertion[3]{} {
5901   \str_set:Nx \l_tmpa_str{ #1 }
5902   \str_if_empty:NTF \l_tmpa_str {
5903     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

5904     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5905   }{
5906     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5907     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5908   }
5909 }

```

(End definition for `\stexpatchassertion`. This function is documented on page 51.)

`\inlineass` inline:

```

5910 \keys_define:nn {stex / inlineass }{
5911   type      .str_set_x:N = \sassertiontype,
5912   id        .str_set_x:N = \sassertionid,
5913   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5914   name      .str_set_x:N = \sassertionname
5915 }
5916 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5917   \str_clear:N \sassertiontype
5918   \str_clear:N \sassertionid
5919   \str_clear:N \sassertionname
5920   \clist_clear:N \l__stex_statements_sassertion_for_clist
5921   \keys_set:nn { stex / inlineass }{ #1 }
5922 }
5923 \NewDocumentCommand \inlineass { 0{} m } {
5924   \beginngroup
5925   \stex_reactivate_macro:N \premise
5926   \stex_reactivate_macro:N \conclusion
5927   \stex_reactivate_macro:N \varbindforall
5928   \__stex_statements_inlineass_args:n{ #1 }
5929   \str_if_empty:NTF \sassertionid {
5930     \str_if_empty:NF \sassertionname {
5931       \stex_ref_new_doc_target:n {}
5932     }
5933   } {
5934     \stex_ref_new_doc_target:n \sassertionid
5935   }
5936
5937   \stex_if_smsmode:TF{
5938     \str_if_empty:NF \sassertionname {
5939       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
5940     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5941   }
5942   }{
5943     \seq_clear:N \l_tmpb_seq
5944     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5945       \tl_if_empty:nF{ ##1 }{
5946         \stex_get_symbol:n { ##1 }
5947         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5948           \l_stex_get_symbol_uri_str
5949         }
5950       }
5951     }
5952     \exp_args:Nnx
5953     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{

```

```

5954     \str_if_empty:NF \sassassertiontype {
5955       \stex_annotate_invisible:nnn{typestrings}{\sassassertiontype}{ }
5956     }
5957     #2
5958     \str_if_empty:NF \sassassertionname {
5959       \stex_suppress_html:n{\stex_symdecl_do:nn{ }{\sassassertionname}}
5960       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassassertionname}
5961       \stex_annotate_invisible:nnn{statementname}{\sassassertionname}{ }
5962     }
5963   }
5964 }
5965 \endgroup
5966 \stex_smsmode_do:
5967 }

```

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

32.3 Examples

`sexample (env.)`

```

5968
5969 \keys_define:nn {stex / sexample }{
5970   type      .str_set_x:N = \exampletype,
5971   id        .str_set_x:N = \sexampleid,
5972   title     .tl_set:N   = \sexamplename,
5973   name      .str_set_x:N = \sexamplename ,
5974   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5975 }
5976 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5977   \str_clear:N \sexampletype
5978   \str_clear:N \sexampleid
5979   \str_clear:N \sexamplename
5980   \tl_clear:N \sexamplename
5981   \clist_clear:N \l__stex_statements_sexample_for_clist
5982   \keys_set:nn { stex / sexample }{ #1 }
5983 }
5984
5985 \NewDocumentEnvironment{sexample}{0{}}{
5986   \__stex_statements_sexample_args:n{ #1 }
5987   \stex_reactivate_macro:N \premise
5988   \stex_reactivate_macro:N \conclusion
5989   \stex_if_smsmode:F {
5990     \seq_clear:N \l_tmpb_seq
5991     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5992       \tl_if_empty:nF{ ##1 }{
5993         \stex_get_symbol:n { ##1 }
5994         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5995           \l_stex_get_symbol_uri_str
5996         }
5997       }
5998     }
5999     \exp_args:Nnnx
6000     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```

```

6001 \str_if_empty:NF \sexamplotype {
6002   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}}
6003 }
6004 \str_if_empty:NF \sexamplename {
6005   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}}
6006 }
6007 \clist_set:No \l_tmpa_clist \sexamplotype
6008 \tl_clear:N \l_tmpa_tl
6009 \clist_map_inline:Nn \l_tmpa_clist {
6010   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6011     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
6012   }
6013 }
6014 \tl_if_empty:NTF \l_tmpa_tl {
6015   \__stex_statements_sexample_start:
6016 }{
6017   \l_tmpa_tl
6018 }
6019 }
6020 \str_if_empty:NF \sexampleid {
6021   \stex_ref_new_doc_target:n \sexampleid
6022 }
6023 \stex_smsmode_do:
6024 }{
6025   \str_if_empty:NF \sexamplename {
6026     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6027   }
6028   \stex_if_smsmode:F {
6029     \clist_set:No \l_tmpa_clist \sexamplotype
6030     \tl_clear:N \l_tmpa_tl
6031     \clist_map_inline:Nn \l_tmpa_clist {
6032       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6033         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6034       }
6035     }
6036     \tl_if_empty:NTF \l_tmpa_tl {
6037       \__stex_statements_sexample_end:
6038     }{
6039       \l_tmpa_tl
6040     }
6041     \end{stex_annotate_env}
6042   }
6043 }

```

\stexpatchexample

```

6044
6045 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6046   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
6047     (\sexampltitle)
6048   }~}
6049 }
6050 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
6051
6052 \newcommand\stexpatchexample[3] [] {

```

```

6053 \str_set:Nx \l_tmpa_str{ #1 }
6054 \str_if_empty:NTF \l_tmpa_str {
6055   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6056   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6057 }{
6058   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6059   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6060 }
6061 }

```

(End definition for `\stexpatchexample`. This function is documented on page 51.)

`\inlineex` inline:

```

6062 \keys_define:nn {stex / inlineex }{
6063   type      .str_set_x:N = \sexamplotype,
6064   id        .str_set_x:N = \sexampleid,
6065   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
6066   name      .str_set_x:N = \sexamplename
6067 }
6068 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6069   \str_clear:N \sexamplotype
6070   \str_clear:N \sexampleid
6071   \str_clear:N \sexamplename
6072   \clist_clear:N \l__stex_statements_sexample_for_clist
6073   \keys_set:nn { stex / inlineex }{ #1 }
6074 }
6075 \NewDocumentCommand \inlineex { 0{} m } {
6076   \begingroup
6077   \stex_reactivate_macro:N \premise
6078   \stex_reactivate_macro:N \conclusion
6079   \__stex_statements_inlineex_args:n{ #1 }
6080   \str_if_empty:NF \sexampleid {
6081     \stex_ref_new_doc_target:n \sexampleid
6082   }
6083   \stex_if_smsmode:TF{
6084     \str_if_empty:NF \sexamplename {
6085       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6086     }
6087   }{
6088     \seq_clear:N \l_tmpb_seq
6089     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6090       \tl_if_empty:nF{ ##1 }{
6091         \stex_get_symbol:n { ##1 }
6092         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6093           \l_stex_get_symbol_uri_str
6094         }
6095       }
6096     }
6097     \exp_args:Nnx
6098     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6099       \str_if_empty:NF \sexamplotype {
6100         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
6101       }
6102       #2

```

```

6103     \str_if_empty:NF \sexamplename {
6104       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
6105       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
6106     }
6107   }
6108 }
6109 \endgroup
6110 \stex_smsmode_do:
6111 }

```

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

32.4 Logical Paragraphs

`sparagraph` (*env.*)

```

6112 \keys_define:nn { stex / sparagraph } {
6113   id      .str_set_x:N = \sparagraphid ,
6114   title   .tl_set:N   = \l_stex_sparagraph_title_tl ,
6115   type    .str_set_x:N = \sparagraphtype ,
6116   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6117   from    .tl_set:N   = \sparagraphfrom ,
6118   to      .tl_set:N   = \sparagraphto ,
6119   start   .tl_set:N   = \l_stex_sparagraph_start_tl ,
6120   name    .str_set:N   = \sparagraphname ,
6121   imports .tl_set:N   = \l__stex_statements_sparagraph_imports_tl
6122 }
6123
6124 \cs_new_protected:Nn \stex_sparagraph_args:n {
6125   \tl_clear:N \l_stex_sparagraph_title_tl
6126   \tl_clear:N \sparagraphfrom
6127   \tl_clear:N \sparagraphto
6128   \tl_clear:N \l_stex_sparagraph_start_tl
6129   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6130   \str_clear:N \sparagraphid
6131   \str_clear:N \sparagraphtype
6132   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6133   \str_clear:N \sparagraphname
6134   \keys_set:nn { stex / sparagraph } { #1 }
6135 }
6136 \newif\if@in@omtext\@in@omtextfalse
6137
6138 \NewDocumentEnvironment {sparagraph} { 0{ } } {
6139   \stex_sparagraph_args:n { #1 }
6140   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6141     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6142   }{
6143     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6144   }
6145   \@in@omtexttrue
6146   \stex_if_smsmode:F {
6147     \seq_clear:N \l_tmpb_seq
6148     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6149       \tl_if_empty:nF{ ##1 }{

```



```

6150     \stex_get_symbol:n { ##1 }
6151     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6152         \l_stex_get_symbol_uri_str
6153     }
6154 }
6155 }
6156 \exp_args:Nnnx
6157 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6158 \str_if_empty:NF \sparagraphtype {
6159     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6160 }
6161 \str_if_empty:NF \sparagraphfrom {
6162     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6163 }
6164 \str_if_empty:NF \sparagraphto {
6165     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6166 }
6167 \str_if_empty:NF \sparagraphname {
6168     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6169 }
6170 \clist_set:No \l_tmpa_clist \sparagraphtype
6171 \tl_clear:N \l_tmpa_tl
6172 \clist_map_inline:Nn \sparagraphtype {
6173     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6174         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
6175     }
6176 }
6177 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6178 \tl_if_empty:NTF \l_tmpa_tl {
6179     \__stex_statements_sparagraph_start:
6180 }{
6181     \l_tmpa_tl
6182 }
6183 }
6184 \clist_set:No \l_tmpa_clist \sparagraphtype
6185 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6186     {
6187         \stex_reactivate_macro:N \definiendum
6188         \stex_reactivate_macro:N \definame
6189         \stex_reactivate_macro:N \Definame
6190         \stex_reactivate_macro:N \premise
6191         \stex_reactivate_macro:N \definiens
6192     }
6193 \str_if_empty:NTF \sparagraphid {
6194     \str_if_empty:NTF \sparagraphname {
6195         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6196             \stex_ref_new_doc_target:n {}
6197         }
6198     } {
6199         \stex_ref_new_doc_target:n {}
6200     }
6201 } {
6202     \stex_ref_new_doc_target:n \sparagraphid
6203 }

```

```

6204 \exp_args:NNx
6205 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6206   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6207     \tl_if_empty:nF{ ##1 }{
6208       \stex_get_symbol:n { ##1 }
6209       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6210     }
6211   }
6212 }
6213 \stex_smsmode_do:
6214 \ignorespacesandpars
6215 }{
6216   \str_if_empty:NF \sparagraphname {
6217     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6218     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6219   }
6220   \stex_if_smsmode:F {
6221     \clist_set:No \l_tmpa_clist \sparagraphtype
6222     \tl_clear:N \l_tmpa_tl
6223     \clist_map_inline:Nn \l_tmpa_clist {
6224       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
6225         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
6226       }
6227     }
6228     \tl_if_empty:NTF \l_tmpa_tl {
6229       \__stex_statements_sparagraph_end:
6230     }{
6231       \l_tmpa_tl
6232     }
6233     \end{stex_annotate_env}
6234   }
6235 }

```

\stexpatchparagraph

```

6236
6237 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6238   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6239     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6240       \titleemph{\l_stex_sparagraph_title_tl}:~
6241     }
6242   }{
6243     \titleemph{\l_stex_sparagraph_start_tl}~
6244   }
6245 }
6246 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6247
6248 \newcommand\stexpatchparagraph[3] [] {
6249   \str_set:Nx \l_tmpa_str{ #1 }
6250   \str_if_empty:NTF \l_tmpa_str {
6251     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6252     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
6253   }{
6254     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
6255     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

6256     }
6257 }
6258
6259 \keys_define:nn { stex / inlinepara } {
6260   id      .str_set:N = \sparagraphid ,
6261   type    .str_set:N = \sparagraphtype ,
6262   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6263   from    .tl_set:N   = \sparagraphfrom ,
6264   to      .tl_set:N   = \sparagraphto ,
6265   name    .str_set:N   = \sparagraphname
6266 }
6267 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6268   \tl_clear:N \sparagraphfrom
6269   \tl_clear:N \sparagraphto
6270   \str_clear:N \sparagraphid
6271   \str_clear:N \sparagraphtype
6272   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6273   \str_clear:N \sparagraphname
6274   \keys_set:nn { stex / inlinepara }{ #1 }
6275 }
6276 \NewDocumentCommand \inlinepara { O{} m } {
6277   \begingroup
6278   \__stex_statements_inlinepara_args:n{ #1 }
6279   \clist_set:Nn \l_tmpa_clist \sparagraphtype
6280   \str_if_empty:NTF \sparagraphid {
6281     \str_if_empty:NTF \sparagraphname {
6282       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6283         \stex_ref_new_doc_target:n {}
6284       }
6285     } {
6286       \stex_ref_new_doc_target:n {}
6287     }
6288   } {
6289     \stex_ref_new_doc_target:n \sparagraphid
6290   }
6291   \stex_if_smsmode:TF{
6292     \str_if_empty:NF \sparagraphname {
6293       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6294     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6295   }
6296 }{
6297   \seq_clear:N \l_tmpb_seq
6298   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6299     \tl_if_empty:nF{ ##1 }{
6300       \stex_get_symbol:n { ##1 }
6301       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6302         \l_stex_get_symbol_uri_str
6303       }
6304     }
6305   }
6306   \exp_args:Nnx
6307   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6308     \str_if_empty:NF \sparagraphtype {
6309       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

6310     }
6311     \str_if_empty:NF \sparagraphfrom {
6312         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6313     }
6314     \str_if_empty:NF \sparagraphto {
6315         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6316     }
6317     \str_if_empty:NF \sparagraphname {
6318         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6319         \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6320         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6321     }
6322     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6323         \clist_map_inline:Nn \l_tmpb_seq {
6324             \stex_ref_new_sym_target:n {##1}
6325         }
6326     }
6327     #2
6328 }
6329 }
6330 \endgroup
6331 \stex_smsmode_do:
6332 }
6333

```

(End definition for `\stexpatchparagraph`. This function is documented on page 51.)

```

6334 </package>

```

Chapter 33

The Implementation

```
6335 <*package>
6336 <@@=stex_sproof>
6337
6338 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
6339
```

33.1 Proofs

We first define some keys for the proof environment.

```
6340 \keys_define:nn { stex / spf } {
6341   id          .str_set_x:N = \spfid,
6342   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
6343   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
6344   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
6345   type        .str_set_x:N = \spftype,
6346   title       .tl_set:N    = \spftitle,
6347   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
6348   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
6349   term        .tl_set:N    = \l__stex_sproof_spf_term_tl,
6350   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
6351   hide        .bool_set:N  = \l__stex_sproof_spf_hide_bool
6352 }
6353 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
6354   \str_clear:N \spfid
6355   \tl_clear:N \l__stex_sproof_spf_for_tl
6356   \tl_clear:N \l__stex_sproof_spf_from_tl
6357   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6358   \str_clear:N \spftype
6359   \tl_clear:N \spftitle
6360   \tl_clear:N \l__stex_sproof_spf_continues_tl
6361   \tl_clear:N \l__stex_sproof_spf_term_tl
6362   \tl_clear:N \l__stex_sproof_spf_functions_tl
6363   \tl_clear:N \l__stex_sproof_spf_method_tl
6364   \bool_set_false:N \l__stex_sproof_spf_hide_bool
6365   \keys_set:nn { stex / spf }{ #1 }
6366 }
6367 \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`

```
6368 \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.

```
6369 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6370 \cs_new_protected:Npn \sproofnumber {
6371   \int_set:Nn \l_tmpa_int {1}
6372   \bool_while_do:nn {
6373     \int_compare_p:nNn {
6374       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6375     } > 0
6376   }{
6377     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6378     \int_incr:N \l_tmpa_int
6379   }
6380 }
6381 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
6382   \int_set:Nn \l_tmpa_int {1}
6383   \bool_while_do:nn {
6384     \int_compare_p:nNn {
6385       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6386     } > 0
6387   }{
6388     \int_incr:N \l_tmpa_int
6389   }
6390   \int_compare:nNnF \l_tmpa_int = 1 {
6391     \int_decr:N \l_tmpa_int
6392   }
6393   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6394     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6395   }
6396 }
6397
6398 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6399   \int_set:Nn \l_tmpa_int {1}
6400   \bool_while_do:nn {
6401     \int_compare_p:nNn {
6402       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6403     } > 0
6404   }{
6405     \int_incr:N \l_tmpa_int
6406   }
6407   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6408 }
6409
```

```

6410 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6411   \int_set:Nn \l_tmpa_int {1}
6412   \bool_while_do:nn {
6413     \int_compare_p:nNn {
6414       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6415     } > 0
6416   }{
6417     \int_incr:N \l_tmpa_int
6418   }
6419   \int_decr:N \l_tmpa_int
6420   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6421 }

```

\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

```

6422 \def\sproof@box{
6423   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6424 }
6425 \def\sproofend{
6426   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6427     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6428   }
6429 }

```

(End definition for \sproofend. This function is documented on page 51.)

spf@*@kw

```

6430 \def\spf@proofsketch@kw{Proof~Sketch}
6431 \def\spf@proof@kw{Proof}
6432 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6433 \AddToHook{begindocument}{
6434   \ltx@ifpackageloaded{babel}{
6435     \makeatletter
6436     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6437     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6438       \input{sproof-ngerman.ldf}
6439     }
6440     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6441       \input{sproof-finnish.ldf}
6442     }
6443     \clist_if_in:NnT \l_tmpa_clist {french}{
6444       \input{sproof-french.ldf}
6445     }
6446     \clist_if_in:NnT \l_tmpa_clist {russian}{
6447       \input{sproof-russian.ldf}
6448     }
6449     \makeatother
6450   }{}
6451 }

```

spfsketch

```

6452 \newcommand\spfsketch[2] [] {
6453   \begin{group}
6454   \let \premise \stex_proof_premise:
6455   \stex_sproof_spf_args:n{#1}
6456   \stex_if_smsmode:TF {
6457     \str_if_empty:NF \spfid {
6458       \stex_ref_new_doc_target:n \spfid
6459     }
6460   }{
6461     \seq_clear:N \l_tmpa_seq
6462     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6463       \tl_if_empty:nF{ ##1 }{
6464         \stex_get_symbol:n { ##1 }
6465         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6466           \l_stex_get_symbol_uri_str
6467         }
6468       }
6469     }
6470     \exp_args:Nnx
6471     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6472       \str_if_empty:NF \spftype {
6473         \stex_annotate_invisible:nnn{type}{\spftype}{ }
6474       }
6475       \clist_set:Nn \l_tmpa_clist \spftype
6476       \tl_set:Nn \l_tmpa_tl {
6477         \titleemph{
6478           \tl_if_empty:NTF \spftitle {
6479             \spf@proofsketch@kw
6480           }{
6481             \spftitle
6482           }
6483         }:-
6484       }
6485       \clist_map_inline:Nn \l_tmpa_clist {
6486         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6487           \tl_clear:N \l_tmpa_tl
6488         }
6489       }
6490       \str_if_empty:NF \spfid {
6491         \stex_ref_new_doc_target:n \spfid
6492       }
6493       \l_tmpa_tl #2 \sproofend
6494     }
6495   }
6496   \endgroup
6497   \stex_smsmode_do:
6498 }
6499

```

(End definition for *spfsketch*. This function is documented on page 50.)

```

\__stex_sproof_maybe_comment:
\__stex_sproof_maybe_comment_end:
\__stex_sproof_start_comment:
6500 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```



```

6501
6502 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6503   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6504     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6505   }
6506 }
6507 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6508   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6509 }
6510 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6511   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6512 }
6513

```

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

\stexcommentfont

```

6514 \cs_new_protected:Npn \stexcommentfont {
6515   \small\itshape
6516 }

```

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

sproof (env.) In this environment, we initialize the proof depth counter \count10 to 10, and set up the description environment that will take the proof steps. At the end of the proof, we position the proof end into the last line.

```

6517 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6518   \seq_clear:N \l_tmpa_seq
6519   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6520     \tl_if_empty:NF{ ##1 }{
6521       \stex_get_symbol:n { ##1 }
6522       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6523         \l_stex_get_symbol_uri_str
6524       }
6525     }
6526   }
6527   \exp_args:Nnnx
6528   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6529   \str_if_empty:NF \spftype {
6530     \stex_annotate_invisible:nnn{type}{\spftype}{}
6531   }
6532   #3 {~\stex_annotate:nnn{spftitle}{}{#2}}
6533   \str_if_empty:NF \spfid {
6534     \stex_ref_new_doc_target:n \spfid
6535   }
6536   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6537   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6538     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6539   }
6540   \begin{list}{}{
6541     \setlength\topsep{0pt}
6542     \setlength\parsep{0pt}
6543     \setlength\rightmargin{0pt}

```

```

6544 } \_stex_sproof_maybe_comment:
6545 }
6546 \cs_new_protected:Nn \_stex_sproof_end_env:n {
6547   \stex_if_smsmode:F{
6548     \_stex_sproof_maybe_comment_end:
6549     \end{list}
6550     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6551       \stex_html_backend:F{\egroup}
6552     }
6553     \clist_set:No \l_tmpa_clist \spftype
6554     #1
6555     \end{stex_annotate_env}
6556     \end{stex_annotate_env}
6557   }
6558 }
6559 }
6560 \NewDocumentEnvironment{sproof}{s O{} m}{
6561   \intarray_gzero:N \l__stex_sproof_counter_intarray
6562   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6563   \stex_reactivate_macro:N \yield
6564   \stex_reactivate_macro:N \eqstep
6565   \stex_reactivate_macro:N \assumption
6566   \stex_reactivate_macro:N \conclude
6567   \stex_reactivate_macro:N \spfstep
6568   \_stex_sproof_spf_args:n{#2}
6569   \stex_if_smsmode:TF {
6570     \str_if_empty:NF \spfid {
6571       \stex_ref_new_doc_target:n \spfid
6572     }
6573   }{
6574     \_stex_sproof_start_env:nnn{sproof}{#3}{
6575       \clist_set:No \l_tmpa_clist \spftype
6576       \tl_clear:N \l_tmpa_tl
6577       \clist_map_inline:Nn \l_tmpa_clist {
6578         \tl_if_exist:cT {\_stex_sproof_sproof_##1_start:}{
6579           \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_start:}}
6580         }
6581         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6582           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6583         }
6584       }
6585       \tl_if_empty:NTF \l_tmpa_tl {
6586         \_stex_sproof_sproof_start:
6587       }{
6588         \l_tmpa_tl
6589       }
6590     }
6591   }
6592   \stex_smsmode_do:
6593 }{\_stex_sproof_end_env:n{
6594   \tl_clear:N \l_tmpa_tl
6595   \clist_map_inline:Nn \l_tmpa_clist {
6596     \tl_if_exist:cT {\_stex_sproof_sproof_##1_end:}{
6597       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_end:}}

```

```

6598     }
6599   }
6600   \tl_if_empty:NTF \l_tmpa_tl {
6601     \__stex_sproof_sproof_end:
6602   }{
6603     \l_tmpa_tl
6604   }
6605 }}
6606 \NewDocumentEnvironment{subproof}{s O{} m}{
6607   \__stex_sproof_spf_args:n{#2}
6608   \stex_if_smsmode:TF {
6609     \str_if_empty:NF \spfid {
6610       \stex_ref_new_doc_target:n \spfid
6611     }
6612   }{
6613     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6614   }
6615   \__stex_sproof_add_counter:
6616   \stex_smsmode_do:
6617 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6618   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6619     \__stex_sproof_inc_counter:
6620   }
6621   \aftergroup\__stex_sproof_maybe_comment:
6622 }
6623 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6624
6625 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6626   \par\noindent\titleemph{
6627     \tl_if_empty:NTF \spftype {
6628       \spf@proof@kw
6629     }{
6630       \spftype
6631     }
6632   }:
6633 }
6634 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6635
6636 \newcommand\stexpatchproof[3] [] {
6637   \str_set:Nx \l_tmpa_str{ #1 }
6638   \str_if_empty:NTF \l_tmpa_str {
6639     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6640     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6641   }{
6642     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6643     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6644   }
6645 }

\pstep
\conclude
\assumption
\have
\eqstep

```

```

6646
6647 \keys_define:nn { stex / spfsteps } {
6648   id          .str_set_x:N = \spfstepid,
6649   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,

```

```

6650 type .str_set_x:N = \spftype,
6651 title .tl_set:N = \spftitle,
6652 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6653 term .tl_set:N = \l__stex_sproof_spf_term_tl
6654 }
6655 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6656 \str_clear:N \spfstepid
6657 \clist_clear:N \l__stex_sproof_spf_for_clist
6658 \str_clear:N \spftype
6659 \tl_clear:N \l__stex_sproof_spf_method_tl
6660 \tl_clear:N \l__stex_sproof_spf_term_tl
6661 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6662 \keys_set:nn { stex / spfsteps }{ #1 }
6663 }
6664
6665 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6666 \NewDocumentCommand #1 {s O{} +m} {
6667 \__stex_sproof_maybe_comment_end:
6668
6669 \__stex_sproof_spfstep_args:n{##2}
6670 \stex_annotate:nnn{spfstep}{#2}{
6671 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6672 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6673 }
6674 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6675 #4
6676 }{
6677 \item[\IfBooleanTF ##1 {}{#3}]
6678 }
6679 \ignorespacesandpars ##3
6680 }
6681 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6682 \__stex_sproof_maybe_comment:
6683 }
6684 \stex_deactivate_macro:Nn #1 {sproof~environments}
6685 }
6686
6687 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6688 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6689 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6690
6691 \NewDocumentCommand \eqstep {s m}{
6692 \__stex_sproof_maybe_comment_end:
6693 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6694 $=$
6695 }{
6696 \item[$=$]
6697 }
6698 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6699 \__stex_sproof_maybe_comment:
6700 }
6701 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6702
6703 \NewDocumentCommand \yield {+m}{

```

```

6704 \stex_annotate:nnn{spfyield}{\}{ #1 }
6705 }
6706 \stex_deactivate_macro:Nn \yield {sproof~environments}
6707
6708 \NewDocumentEnvironment{spfblock}{\}{\{
6709 \item[]
6710 \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6711 \}{
6712 \aftergroup\__stex_sproof_maybe_comment:
6713 }
6714 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6715

```

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

\spfidea

```

6716 \NewDocumentCommand\spfidea{0{\} +m}{
6717 \__stex_sproof_spf_args:n{#1}
6718 \titleemph{
6719 \tl_if_empty:NTF \spftype {Proof~Idea}{
6720 \spftype
6721 }:
6722 }~#2
6723 \sproofend
6724 }

```

(End definition for \spfidea. This function is documented on page 50.)

```

6725 \newcommand\spfjust[1]{
6726 #1
6727 }
6728 \</package>

```

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

Chapter 34

STEX -Others Implementation

```
6729 <*package>
6730
6731 %%%%%%%%%% others.dtx %%%%%%%%%%
6732
6733 <@@=stex_others>
        Warnings and error messages
6734 % None

\MSC Math subject classifier

6735 \NewDocumentCommand \MSC {m} {
6736 % TODO
6737 }

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

6738 \@ifpackageloaded{tikzinput}{
6739 \RequirePackage{stex-tikzinput}
6740 }{}
6741
6742 \bool_if:NT \c_stex_persist_mode_bool {
6743 \let__stex_notation_restore_notation_old:nnnnn
6744 \__stex_notation_restore_notation:nnnnn
6745 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6746 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6747 \ExplSyntaxOn
6748 }
6749 \def__stex_notation_restore_notation:nnnnn{
6750 \ExplSyntaxOff
6751 \catcode'\sim10
6752 \__stex_notation_restore_notation_new:nnnnn
6753 }
6754 \input{\jobname.sms}
6755 \let__stex_notation_restore_notation:nnnnn
6756 \__stex_notation_restore_notation_old:nnnnn
6757 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```

6758     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6759     \l_tmpa_str
6760     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6761     \c_stex_mathhub_main_manifest_prop
6762     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6763   }
6764 }
6765
6766 \stex_get_document_uri:
6767 </package>

```

Chapter 35

STEX -Metatheory Implementation

```
6768 <*package>
6769 <@@=stex_modules>
6770
6771 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6772
6773 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6774 \begingroup
6775 \stex_module_setup:nn{
6776   ns=\c_stex_metatheory_ns_str,
6777   meta=NONE
6778 }{Metatheory}
6779 \stex_reactivate_macro:N \symdecl
6780 \stex_reactivate_macro:N \notation
6781 \stex_reactivate_macro:N \symdef
6782 \ExplSyntaxOff
6783 \csname stex_suppress_html:n\endcsname{
6784   % is-a (a:A, a \in A, a is an A, etc.)
6785   \symdecl{isa}[args=ai]
6786   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6787   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6788   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6789
6790   % bind (\forall, \Pi, \lambda etc.)
6791   \symdecl{bind}[args=Bi,assoc=pre]
6792   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6793   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6794   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6795
6796   % implicit bind
6797   \symdecl{implicitbind}[args=Bi,assoc=pre]
6798   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6799   \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6800   \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6801
6802   % dummy variable
```



```

6803 \symdecl{dummyvar}
6804 \notation{dummyvar}[underscore]{\comp\_}
6805 \notation{dummyvar}[dot]{\comp\cdot}
6806 \notation{dummyvar}[dash]{\comp{\rm --}}
6807
6808 %fromto (function space, Hom-set, implication etc.)
6809 \symdecl{fromto}[args=ai]
6810 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6811 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6812
6813 % mapto (lambda etc.)
6814 \symdecl{mapto}[args=Bi]
6815 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6816 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6817 %\notation{mapto}[lambdau]{\comp\lambda_#1 \comp.\; #2}{#1 \comp, #2}
6818
6819 % function/operator application
6820 \symdecl{apply}[args=ia]
6821 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6822 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6823
6824 % collection of propositions/booleans/truth values
6825 \symdecl{prop}[name=proposition]
6826 \notation{prop}[prop]{\comp{\rm prop}}
6827 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6828
6829 \symdecl{judgmentholds}[args=1]
6830 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6831
6832 % sequences
6833 \symdecl{seqtype}[args=1]
6834 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6835
6836 \symdecl{seqexpr}[args=a]
6837 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6838
6839 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
6840 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6841 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6842 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6843 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6844 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6845 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6846 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6847 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6848
6849 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6850 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
6851
6852 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\ellipses}}{##1\comp,##2}
6853 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\ellipses},#2}{##1\comp,##2}
6854 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\ellipses},#2\comp{\ellipses},#3}{##1\comp,##2,##3}
6855
6856 % nat literals

```

```

6857 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6858
6859 % letin (''let'', local definitions, variable substitution)
6860 \symdecl{letin}[args=bii]
6861 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
6862 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
6863 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
6864
6865 % structures
6866 \symdecl*{module-type}[args=1]
6867 \notation{module-type}{\comp{\mathtt{MOD}}}\;#1}
6868 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6869 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6870
6871 % objects
6872 \symdecl{object}
6873 \notation{object}{\comp{\mathtt{OBJECT}}}
6874
6875 }
6876
6877 % The following are abbreviations in the sTeX corpus that are left over from earlier
6878 % developments. They will eventually be phased out.
6879
6880 \ExplSyntaxOn
6881 \stex_add_to_current_module:n{
6882   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6883   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6884   \def\livar{\csname sequence-index\endcsname[li]}
6885   \def\uivar{\csname sequence-index\endcsname[ui]}
6886   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6887   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6888 }
6889 \__stex_modules_end_module:
6890 \endgroup
6891 </package>

```

Chapter 36

Tikzinput Implementation

```
6892 <@@=tikzinput>
6893 <*package>
6894
6895 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6896
6897 \ProvidesExplPackage{tikzinput}{2022/08/08}{3.2.0}{tikzinput package}
6898 \RequirePackage{l3keys2e}
6899
6900 \keys_define:nn { tikzinput } {
6901   image .bool_set:N = \c_tikzinput_image_bool,
6902   image .default:n = false ,
6903   unknown .code:n = {}
6904 }
6905
6906 \ProcessKeysOptions { tikzinput }
6907
6908 \bool_if:NTF \c_tikzinput_image_bool {
6909   \RequirePackage{graphicx}
6910
6911   \providecommand\usetikzlibrary[]{}
6912   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6913 }{
6914   \RequirePackage{tikz}
6915   \RequirePackage{standalone}
6916
6917   \newcommand \tikzinput [2] [] {
6918     \setkeys{Gin}{#1}
6919     \ifx \Gin@ewidth \Gin@exclamation
6920       \ifx \Gin@eheight \Gin@exclamation
6921         \input { #2 }
6922       \else
6923         \resizebox{!}{ \Gin@eheight }{
6924           \input { #2 }
6925         }
6926       \fi
6927     \else
6928       \ifx \Gin@eheight \Gin@exclamation
6929         \resizebox{ \Gin@ewidth }{!}{
```

```

6930         \input { #2 }
6931     }
6932     \else
6933         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6934             \input { #2 }
6935         }
6936     \fi
6937 \fi
6938 }
6939 }
6940
6941 \newcommand \ctikzinput [2] [] {
6942     \begin{center}
6943         \tikzinput [#1] {#2}
6944     \end{center}
6945 }
6946
6947 \@ifpackageloaded{stex}{
6948     \RequirePackage{stex-tikzinput}
6949 }{}
6950
6951 </package>
6952 <*stex>
6953 \ProvidesExplPackage{stex-tikzinput}{2022/08/08}{3.2.0}{stex-tikzinput}
6954 \RequirePackage{stex}
6955 \RequirePackage{tikzinput}
6956
6957 \newcommand\mhtikzinput[2] []{%
6958     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
6959     \stex_in_repository:nn\Gin@mhrepos{
6960         \tikzinput[#1]{\mhp@path{##1}{#2}}
6961     }
6962 }
6963 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6964
6965 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6966     \pgfkeys@spdef\pgf@temp{#1}
6967     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6968     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6969     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6970     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6971     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6972     \catcode'\@=11
6973     \catcode'\|=12
6974     \catcode'\$=3
6975     \pgfutil@InputIfFileExists{#2}{-}{-}
6976     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6977     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6978     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6979 }
6980
6981
6982 \newcommand\libusetikzlibrary[1]{

```

```

6983 \prop_if_exist:NF \l_stex_current_repository_prop {
6984   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6985 }
6986 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6987   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6988 }
6989 \seq_clear:N \l__tikzinput_libinput_files_seq
6990 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6991 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6992
6993 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6994   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6995   \IfFileExists{ \l_tmpa_str }{
6996     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6997   }{}
6998   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6999   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
7000 }
7001
7002 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
7003 \IfFileExists{ \l_tmpa_str }{
7004   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7005 }{}
7006
7007 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
7008   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
7009 }{
7010   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7011     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7012       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7013     }
7014   }{
7015     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7016   }
7017 }
7018 }
7019 </stex>

```

Chapter 37

document-structure.sty Implementation

```
7020 <*package>
7021 <@@=document_structure>
7022 \ProvidesExplPackage{document-structure}{2022/08/08}{3.2.0}{Modular Document Structure}
7023 \RequirePackage{13keys2e}
```

37.1 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).

```
7024
7025 \keys_define:nn{ document-structure }{
7026   class      .str_set_x:N = \c_document_structure_class_str,
7027   topsect    .str_set_x:N = \c_document_structure_topsect_str,
7028   unknown    .code:n      = {
7029     \PassOptionsToClass{\CurrentOption}{stex}
7030     \PassOptionsToClass{\CurrentOption}{tikzinput}
7031   }
7032   % showignores .bool_set:N = \c_document_structure_showignores_bool,
7033 }
7034 \ProcessKeysOptions{ document-structure }
7035 \str_if_empty:NT \c_document_structure_class_str {
7036   \str_set:Nn \c_document_structure_class_str {article}
7037 }
7038 \str_if_empty:NT \c_document_structure_topsect_str {
7039   \str_set:Nn \c_document_structure_topsect_str {section}
7040 }
```

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

```
7041 \RequirePackage{xspace}
7042 \RequirePackage{comment}
7043 \RequirePackage{stex}
7044 \AddToHook{begindocument}{
```

```

7045 \ltx@ifpackageloaded{babel}{
7046   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7047   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7048     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7049   }
7050 }{}
7051 }

```

`\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.

```

7052 \int_new:N \l_document_structure_section_level_int
7053 \str_case:NnF \c_document_structure_topsect_str {
7054   {part}}{
7055     \int_set:Nn \l_document_structure_section_level_int {0}
7056   }
7057   {chapter}{
7058     \int_set:Nn \l_document_structure_section_level_int {1}
7059   }
7060 }{
7061   \str_case:NnF \c_document_structure_class_str {
7062     {book}{
7063       \int_set:Nn \l_document_structure_section_level_int {0}
7064     }
7065     {report}{
7066       \int_set:Nn \l_document_structure_section_level_int {0}
7067     }
7068   }{
7069     \int_set:Nn \l_document_structure_section_level_int {2}
7070   }
7071 }

```

37.2 Document Structure

The structure of the document is given by the `sfragment` environment. The hierarchy is adjusted automatically according to the L^AT_EX class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.⁹

EdN:9

```

7072 \def\current@section@level{document}%
7073 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
7074 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

7075 \cs_new_protected:Npn \skipfragment {

```

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

```

7076 \ifcase\l_document_structure_section_level_int
7077 \or\stepcounter{part}
7078 \or\stepcounter{chapter}
7079 \or\stepcounter{section}
7080 \or\stepcounter{subsection}
7081 \or\stepcounter{subsubsection}
7082 \or\stepcounter{paragraph}
7083 \or\stepcounter{subparagraph}
7084 \fi
7085 }

```

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

`blindfragment (env.)`

```

7086 \newcommand\at@begin@blindsfragment[1]{
7087 \newenvironment{blindfragment}
7088 {
7089 \int_incr:N\l_document_structure_section_level_int
7090 \at@begin@blindsfragment\l_document_structure_section_level_int
7091 }{}

```

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

```

7092 \newcommand\sfragment@nonum[2]{
7093 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7094 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7095 }

```

(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 `rdmeta` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```

7096 \newcommand\sfragment@num[2]{
7097 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7098 \@nameuse{#1}{#2}
7099 }{
7100 \cs_if_exist:NTF\rdmeta@sectioning{
7101 \@nameuse{rdmeta@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7102 }{
7103 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7104 }
7105 }
7106 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7107 }

```

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

`sfragment (env.)`

```

7108 \keys_define:nn { document-structure / sfragment }{
7109 id .str_set_x:N = \l__document_structure_sfragment_id_str,
7110 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```



```

7111 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7112 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7113 srccite       .tl_set:N      = \l__document_structure_sfragment_srccite_tl,
7114 type         .tl_set:N      = \l__document_structure_sfragment_type_tl,
7115 short        .tl_set:N      = \l__document_structure_sfragment_short_tl,
7116 intro        .tl_set:N      = \l__document_structure_sfragment_intro_tl,
7117 imports      .tl_set:N      = \l__document_structure_sfragment_imports_tl,
7118 loadmodules   .bool_set:N    = \l__document_structure_sfragment_loadmodules_bool
7119 }
7120 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
7121   \str_clear:N \l__document_structure_sfragment_id_str
7122   \str_clear:N \l__document_structure_sfragment_date_str
7123   \clist_clear:N \l__document_structure_sfragment_creators_clist
7124   \clist_clear:N \l__document_structure_sfragment_contributors_clist
7125   \tl_clear:N \l__document_structure_sfragment_srccite_tl
7126   \tl_clear:N \l__document_structure_sfragment_type_tl
7127   \tl_clear:N \l__document_structure_sfragment_short_tl
7128   \tl_clear:N \l__document_structure_sfragment_imports_tl
7129   \tl_clear:N \l__document_structure_sfragment_intro_tl
7130   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7131   \keys_set:nn { document-structure / sfragment } { #1 }
7132 }

```

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.

```

7133 \newif\if@mainmatter\@mainmattertrue
7134 \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.

```

7135 \keys_define:nn { document-structure / sectioning }{
7136   name      .str_set_x:N = \l__document_structure_sect_name_str  ,
7137   ref       .str_set_x:N = \l__document_structure_sect_ref_str   ,
7138   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
7139   clear     .default:n   = {true}                                ,
7140   num       .bool_set:N  = \l__document_structure_sect_num_bool   ,
7141   num       .default:n   = {true}
7142 }
7143 \cs_new_protected:Nn \l__document_structure_sect_args:n {
7144   \str_clear:N \l__document_structure_sect_name_str
7145   \str_clear:N \l__document_structure_sect_ref_str
7146   \bool_set_false:N \l__document_structure_sect_clear_bool
7147   \bool_set_false:N \l__document_structure_sect_num_bool
7148   \keys_set:nn { document-structure / sectioning } { #1 }
7149 }
7150 \newcommand\omdoc@sectioning[3][]{
7151   \l__document_structure_sect_args:n {#1 }
7152   \let\omdoc@sect@name\l__document_structure_sect_name_str
7153   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7154   \if@mainmatter% numbering not overridden by frontmatter, etc.
7155     \bool_if:NTF \l__document_structure_sect_num_bool {
7156       \sfragment@num{#2}{#3}
7157     }{

```

```

7158     \sfragment@nonum{#2}{#3}
7159   }
7160   \def\current@section@level{\omdoc@sect@name}
7161   \else
7162     \sfragment@nonum{#2}{#3}
7163   \fi
7164 }% 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.

```

7165 \newcommand\sfragment@redefine@addtocontents[1]{%
7166 %\edef\__document_structureimport{#1}%
7167 %\@for\@I:=\__document_structureimport\do{%
7168 %\edef\@path{\csname module@\@I @path\endcsname}%
7169 %\@ifundefined{tf@toc}\relax%
7170 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
7171 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7172 %\def\addcontentsline##1##2##3{%
7173 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7174 %\else% hyperref.sty not loaded
7175 %\def\addcontentsline##1##2##3{%
7176 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7177 %\fi
7178 }% 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.

```

7179 \newenvironment{sfragment}[2][ ]% keys, title
7180 {
7181   \__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.

```

7182   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7183
7184   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7185     \sfragment@redefine@addtocontents{
7186       %\@ifundefined{module@id}\used@modules%
7187       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7188     }
7189   }

```

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

```

7190
7191   \stex_document_title:n { #2 }
7192
7193   \int_incr:N\l__document_structure_section_level_int
7194   \ifcase\l__document_structure_section_level_int
7195     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7196     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7197     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
7198     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

7199 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7200 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
7201 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#1}
7202 \fi
7203 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
7204 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7205   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7206 }
7207 }% for customization
7208 {}

```

and finally, we localize the sections

```

7209 \newcommand\omdoc@part@kw{Part}
7210 \newcommand\omdoc@chapter@kw{Chapter}
7211 \newcommand\omdoc@section@kw{Section}
7212 \newcommand\omdoc@subsection@kw{Subsection}
7213 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7214 \newcommand\omdoc@paragraph@kw{paragraph}
7215 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

37.3 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`

```

7216 \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).

```

7217 \cs_if_exist:NTF\frontmatter{
7218   \let\__document_structure_orig_frontmatter\frontmatter
7219   \let\frontmatter\relax
7220 }{
7221   \tl_set:Nn\__document_structure_orig_frontmatter{
7222     \clearpage
7223     \@mainmatterfalse
7224     \pagenumbering{roman}
7225   }
7226 }
7227 \cs_if_exist:NTF\backmatter{
7228   \let\__document_structure_orig_backmatter\backmatter
7229   \let\backmatter\relax
7230 }{
7231   \tl_set:Nn\__document_structure_orig_backmatter{
7232     \clearpage
7233     \@mainmatterfalse
7234     \pagenumbering{roman}
7235   }

```

7236 }

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

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

```
7237 \newenvironment{frontmatter}{
7238   \_document_structure_orig_frontmatter
7239 }{
7240   \cs_if_exist:NTF\mainmatter{
7241     \mainmatter
7242   }{
7243     \clearpage
7244     \@mainmattertrue
7245     \pagenumbering{arabic}
7246   }
7247 }
```

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

```
7248 \newenvironment{backmatter}{
7249   \_document_structure_orig_backmatter
7250 }{
7251   \cs_if_exist:NTF\mainmatter{
7252     \mainmatter
7253   }{
7254     \clearpage
7255     \@mainmattertrue
7256     \pagenumbering{arabic}
7257   }
7258 }
```

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

```
7259 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7260 \def \c__document_structure_document_str{document}
7261 \newcommand\afterprematurestop{}
7262 \def\prematurestop@endsfragment{
7263   \unless\ifx\@currenvir\c__document_structure_document_str
7264     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7265     \expandafter\prematurestop@endsfragment
7266   \fi
7267 }
7268 \providecommand\prematurestop{
7269   \message{Stopping~sTeX~processing~prematurely}
7270   \prematurestop@endsfragment
7271   \afterprematurestop
7272   \end{document}
7273 }
```

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

37.4 Global Variables

\setSGvar set a global variable

```
7274 \RequirePackage{etoolbox}
7275 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
7276 \newrobustcmd\useSGvar[1]{%
7277   \@ifundefined{sTeX@Gvar@#1}
7278   {\PackageError{document-structure}
7279     {The sTeX Global variable #1 is undefined}
7280     {set it with \protect\setSGvar}}
7281   \@nameuse{sTeX@Gvar@#1}}
```

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

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

```
7282 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7283   \@ifundefined{sTeX@Gvar@#1}
7284   {\PackageError{document-structure}
7285     {The sTeX Global variable #1 is undefined}
7286     {set it with \protect\setSGvar}}
7287   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

Chapter 38

NotesSlides – Implementation

38.1 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.

```
7288 \*cls)
7289 \@@=notesslides)
7290 \ProvidesExplClass{notesslides}{2022/08/08}{3.2.0}{notesslides Class}
7291 \RequirePackage{13keys2e}
7292
7293 \keys_define:nn{notesslides / cls}{
7294   class .str_set_x:N = \c__notesslides_class_str,
7295   notes .bool_set:N = \c__notesslides_notes_bool ,
7296   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7297   docopt .str_set_x:N = \c__notesslides_docopt_str,
7298   unknown .code:n = {
7299     \PassOptionsToPackage{\CurrentOption}{document-structure}
7300     \PassOptionsToClass{\CurrentOption}{beamer}
7301     \PassOptionsToPackage{\CurrentOption}{notesslides}
7302     \PassOptionsToPackage{\CurrentOption}{stex}
7303   }
7304 }
7305 \ProcessKeysOptions{ notesslides / cls }
7306
7307 \str_if_empty:NF \c__notesslides_class_str {
7308   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7309 }
7310
7311 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7312   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7313 }
7314 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7315   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
7316 }
7317
7318 \RequirePackage{stex}
```

```

7319 \stex_html_backend:T {
7320   \bool_set_true:N\c__notesslides_notes_bool
7321 }
7322
7323 \bool_if:NTF \c__notesslides_notes_bool {
7324   \PassOptionsToPackage{notes=true}{notesslides}
7325   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7326 }{
7327   \PassOptionsToPackage{notes=false}{notesslides}
7328   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7329 }
7330 </cls>

```

now we do the same for the notesslides package.

```

7331 <*package>
7332 \ProvidesExplPackage{notesslides}{2022/08/08}{3.2.0}{notesslides Package}
7333 \RequirePackage{l3keys2e}
7334
7335 \keys_define:nn{notesslides / pkg}{
7336   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7337   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7338   notes            .bool_set:N = \c__notesslides_notes_bool ,
7339   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7340   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7341   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7342   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7343   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
7344   unknown          .code:n      = {
7345     \PassOptionsToClass{\CurrentOption}{stex}
7346     \PassOptionsToClass{\CurrentOption}{tikzinput}
7347   }
7348 }
7349 \ProcessKeysOptions{ notesslides / pkg }
7350
7351 \RequirePackage{stex}
7352 \stex_html_backend:T {
7353   \bool_set_true:N\c__notesslides_notes_bool
7354 }
7355
7356 \newif\ifnotes
7357 \bool_if:NTF \c__notesslides_notes_bool {
7358   \notesttrue
7359 }{
7360   \notestfalse
7361 }
7362

```

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

```

7363 \str_if_empty:NTF \c__notesslides_topsect_str {
7364   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7365 }{
7366   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7367 }
7368 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
7369 </package>
```

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

```
7370 <*cls>
7371 \bool_if:NTF \c__notesslides_notes_bool {
7372   \str_if_empty:NT \c__notesslides_class_str {
7373     \str_set:Nn \c__notesslides_class_str {article}
7374   }
7375   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
7376   {\c__notesslides_class_str}
7377 }{
7378   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7379   \newcounter{Item}
7380   \newcounter{paragraph}
7381   \newcounter{subparagraph}
7382   \newcounter{Hfootnote}
7383 }
7384 \RequirePackage{document-structure}
```

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

```
7385 \RequirePackage{notesslides}
7386 </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.

```
7387 <*package>
7388 \bool_if:NT \c__notesslides_notes_bool {
7389   \RequirePackage{a4wide}
7390   \RequirePackage{marginnote}
7391   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7392   \RequirePackage{mdframed}
7393   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7394   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7395 }
7396 \RequirePackage{stex-tikzinput}
7397 \RequirePackage{comment}
7398 \RequirePackage{url}
7399 \RequirePackage{graphicx}
7400 \RequirePackage{pgf}
7401 \RequirePackage{bookmark}
```

38.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class.

```
7402 \bool_if:NT \c__notesslides_notes_bool {
7403   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7404 }
```



```

7405 \NewDocumentCommand \libusetheme {0{} m} {
7406   \libusepackage[#1]{beamertheme#2}
7407 }
7408

```

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

```

7409 \newcounter{slide}
7410 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7411 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

note (*env.*) The **note** environment is used to leave out text in the **slides** mode. It does not have a counterpart in OMDoc. So for course notes, we define the **note** environment to be a no-operation otherwise we declare the **note** environment as a comment via the **comment** package.

```

7412 \bool_if:NTF \c__notesslides_notes_bool {
7413   \renewenvironment{note}{\ignorespaces}{}
7414 }{
7415   \excludecomment{note}
7416 }

```

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.

```

7417 \bool_if:NT \c__notesslides_notes_bool {
7418   \newlength{\slideframewidth}
7419   \setlength{\slideframewidth}{1.5pt}

```

frame (*env.*) We first define the keys.

```

7420 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7421   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7422     \bool_set_true:N #1
7423   }{
7424     \bool_set_false:N #1
7425   }
7426 }
7427 \keys_define:nn{notesslides / frame}{
7428   label .str_set_x:N = \l__notesslides_frame_label_str,
7429   allowframebreaks .code:n = {
7430     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7431   },
7432   allowdisplaybreaks .code:n = {
7433     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7434   },
7435   fragile .code:n = {
7436     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7437   },
7438   shrink .code:n = {
7439     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7440   },
7441   squeeze .code:n = {
7442     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7443   },
7444   t .code:n = {

```

```

7445     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7446   },
7447   unknown    .code:n      = {}
7448 }
7449 \cs_new_protected:Nn \__notesslides_frame_args:n {
7450   \str_clear:N \l__notesslides_frame_label_str
7451   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7452   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7453   \bool_set_true:N \l__notesslides_frame_fragile_bool
7454   \bool_set_true:N \l__notesslides_frame_shrink_bool
7455   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7456   \bool_set_true:N \l__notesslides_frame_t_bool
7457   \keys_set:nn { notesslides / frame }{ #1 }
7458 }

```

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

```

7459 \renewenvironment{frame}[1][]{
7460   \__notesslides_frame_args:n{#1}
7461   \sffamily
7462   \stepcounter{slide}
7463   \def\@currentlabel{\theslide}
7464   \str_if_empty:NF \l__notesslides_frame_label_str {
7465     \label{\l__notesslides_frame_label_str}
7466   }

```

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

```

7467   \def\itemize@level{outer}
7468   \def\itemize@outer{outer}
7469   \def\itemize@inner{inner}
7470   \renewcommand\newpage{\addtocounter{framenum}{1}}
7471   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7472   \renewenvironment{itemize}{
7473     \ifx\itemize@level\itemize@outer
7474       \def\itemize@label{$\rhd$}
7475     \fi
7476     \ifx\itemize@level\itemize@inner
7477       \def\itemize@label{$\scriptstyle\rhd$}
7478     \fi
7479     \begin{list}
7480       {\itemize@label}
7481       {\setlength{\labelsep}{.3em}
7482        \setlength{\labelwidth}{.5em}
7483        \setlength{\leftmargin}{1.5em}
7484       }
7485     \edef\itemize@level{\itemize@inner}
7486   }{
7487     \end{list}
7488   }

```

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

```

7489   \stex_html_backend:TF {
7490     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7491     \mdf@patchamsthm
7492   }{
7493     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7494     }
7495   }{
7496     \stex_html_backend:TF {
7497       \miko@slidelabel\egroup\end{stex_annotate_env}
7498     }\medskip\miko@slidelabel\end{mdframed}}
7499   }

```

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

`\frametitle`

```

7500   \renewcommand{\frametitle}[1]{
7501     \stex_document_title:n { #1 }
7502     {\Large\bf\sf\color{blue}{#1}}\medskip
7503   }
7504 }

```

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

EdN:10

`\pause` 10

```

7505 \bool_if:NT \c__notesslides_notes_bool {
7506   \newcommand\pause{}
7507 }

```

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

`nparagraph (env.)`

```

7508 \bool_if:NTF \c__notesslides_notes_bool {
7509   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7510 }{
7511   \excludecomment{nparagraph}
7512 }

```

`nfragment (env.)`

```

7513 \bool_if:NTF \c__notesslides_notes_bool {
7514   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1][#2]}\end{sfragment}}
7515 }{
7516   \excludecomment{nfragment}
7517 }

```

`ndefinition (env.)`

```

7518 \bool_if:NTF \c__notesslides_notes_bool {
7519   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7520 }{
7521   \excludecomment{ndefinition}
7522 }

```

`nassertion (env.)`

```

7523 \bool_if:NTF \c__notesslides_notes_bool {
7524   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7525 }{
7526   \excludecomment{nassertion}
7527 }

```

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

`nsproof (env.)`

```
7528 \bool_if:NTF \c__notesslides_notes_bool {
7529   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7530 }{
7531   \excludecomment{nproof}
7532 }
```

`nexample (env.)`

```
7533 \bool_if:NTF \c__notesslides_notes_bool {
7534   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7535 }{
7536   \excludecomment{nexample}
7537 }
```

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

```
7538 \def\inputref@preskip{\smallskip}
7539 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
7540 \let\orig@inputref\inputref
7541 \def\inputref{@ifstar\ninputref\orig@inputref}
7542 \newcommand\ninputref[2] []{
7543   \bool_if:NT \c__notesslides_notes_bool {
7544     \orig@inputref[#1]{#2}
7545   }
7546 }
```

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

38.3 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>}`.

```
7547 \newlength{\slidelogoheight}
7548
7549 \RequirePackage{graphicx}
7550
7551 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7552 \providecommand\mhgraphics[2] []{
7553   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7554   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7555 }
7556
7557 \bool_if:NTF \c__notesslides_notes_bool {
7558   \setlength{\slidelogoheight}{.4cm}
7559 }{
7560   \setlength{\slidelogoheight}{.25cm}
7561 }
```

```

7562 \ifcsname slidelogo\endcsname\else
7563   \newsavebox{\slidelogo}
7564   \sbox{\slidelogo}{\sTeX}
7565 \fi
7566 \newrobustcmd{\setslidelogo}[2][]{
7567   \tl_if_empty:nTF{#1}{
7568     \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7569   }{
7570     \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7571   }
7572 }

```

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

\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.

```

7573 \bool_if:NT \c__notesslides_notes_bool {
7574   \def\author{\@dblarg\ns@author}
7575   \long\def\ns@author[#1]#2{%
7576     \def\c__notesslides_shortauthor{#1}%
7577     \def\@author{#2}
7578   }
7579 }

```

(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.

```

7580 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

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

\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.

```

7581 \def\copyrightnotice{%
7582   \footnotesize\copyright : \hspace{.3ex}%
7583   \ifcsname source\endcsname\source\else%
7584   \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7585   \PackageWarning{notesslides}{Author/Source~undefined~in~copyright~notice}%
7586   ?source/author?\fi%
7587 \fi}
7588 \newsavebox{\cclogo}
7589 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7590 \newif\ifcchref\cchreffalse
7591 \AtBeginDocument{
7592   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7593 }
7594 \def\licensing{
7595   \ifcchref
7596     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7597   \else
7598     {\usebox{\cclogo}}

```

```

7599 \fi
7600 }
7601 \newrobustcmd{\setlicensing}[2][]{
7602   \def\@url{#1}
7603   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7604   \ifx\@url\@empty
7605     \def\licensing{\usebox{\cclogo}}
7606   \else
7607     \def\licensing{
7608       \ifcchref
7609         \href{#1}{\usebox{\cclogo}}
7610       \else
7611         {\usebox{\cclogo}}
7612       \fi
7613     }
7614   \fi
7615 }

```

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

EdN:11

```

\slidelabel Now, we set up the slide label for the article mode.11
7616 \newrobustcmd\miko@slidelabel{
7617   \vbox to \slidelogoheight{
7618     \vss\hbox to \slidewidth
7619     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7620   }
7621 }

```

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

38.4 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.

```

7622 \def\Gin@mhrepos{}
7623 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7624 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7625 \newrobustcmd\frameimage[2][]{
7626   \stepcounter{slide}
7627   \bool_if:NT \c__notesslides_frameimages_bool {
7628     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7629     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7630     \begin{center}
7631       \bool_if:NTF \c__notesslides_fiboxed_bool {
7632         \fbox{
7633           \ifx\Gin@ewidth\@empty
7634             \ifx\Gin@mhrepos\@empty
7635               \mhgraphics[width=\slidewidth,#1]{#2}
7636             \else
7637               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7638             \fi
7639           \else% Gin@ewidth empty

```

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

```

7640         \ifx\Gin@mhrepos\@empty
7641         \mhgraphics[#1]{#2}
7642     \else
7643         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7644     \fi
7645 \fi% Gin@ewidth empty
7646 }
7647 }{
7648     \ifx\Gin@ewidth\@empty
7649     \ifx\Gin@mhrepos\@empty
7650         \mhgraphics[width=\slidewidth,#1]{#2}
7651     \else
7652         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7653     \fi
7654     \ifx\Gin@mhrepos\@empty
7655         \mhgraphics[#1]{#2}
7656     \else
7657         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7658     \fi
7659     \fi% Gin@ewidth empty
7660 }
7661 \end{center}
7662 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7663 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7664 }
7665 } % ifmks@sty@frameimages

```

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

38.5 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`.

```

7666 \stex_html_backend:F {
7667     \bool_if:NT \c__notesslides_sectocframes_bool {
7668         \str_if_eq:VnTF \__notesslidestopsect{part}{
7669             \newcounter{chapter}\counterwithin*{section}{chapter}
7670         }{
7671             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7672                 \newcounter{chapter}\counterwithin*{section}{chapter}
7673             }
7674         }
7675     }
7676 }

```

`\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
7677 \def\part@prefix{}
7678 \@ifpackageloaded{document-structure}{
7679     \str_case:VnF \__notesslidestopsect {

```

```

7680 {part}{
7681   \int_set:Nn \l_document_structure_section_level_int {0}
7682   \def\thesection{\arabic{chapter}.\arabic{section}}
7683   \def\part@prefix{\arabic{chapter}.}
7684 }
7685 {chapter}{
7686   \int_set:Nn \l_document_structure_section_level_int {1}
7687   \def\thesection{\arabic{chapter}.\arabic{section}}
7688   \def\part@prefix{\arabic{chapter}.}
7689 }
7690 }{
7691   \int_set:Nn \l_document_structure_section_level_int {2}
7692   \def\part@prefix{}
7693 }
7694 }
7695
7696 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the `sfragment` environment that chooses the L^AT_EX sectioning macros according to `\section@level`.

`sfragment (env.)`

```

7697 \renewenvironment{sfragment}[2][]{
7698   \__document_structure_sfragment_args:n { #1 }
7699   \int_incr:N \l_document_structure_section_level_int
7700   \bool_if:NT \c__notesslides_sectocframes_bool {
7701     \stepcounter{slide}
7702     \begin{frame}[noframenumbering]
7703     \vfill\Large\centering
7704     \red{
7705       \ifcase\l_document_structure_section_level_int\or
7706         \stepcounter{part}
7707         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7708         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7709         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7710         \def\currentsectionlevel{\omdoc@part@kw}
7711       \or
7712         \stepcounter{chapter}
7713         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7714         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7715         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7716         \def\currentsectionlevel{\omdoc@chapter@kw}
7717       \or
7718         \stepcounter{section}
7719         \def\__notesslideslabel{\part@prefix\arabic{section}}
7720         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7721         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7722         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7723         \def\currentsectionlevel{\omdoc@section@kw}
7724       \or
7725         \stepcounter{subsection}
7726         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7727         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```



```

7728         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7729         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7730         \def\currentsectionlevel{\omdoc@subsection@kw}
7731     \or
7732         \stepcounter{subsubsection}
7733         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7734         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7735         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7736         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7737         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7738     \or
7739         \stepcounter{paragraph}
7740         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7741         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7742         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7743         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7744         \def\currentsectionlevel{\omdoc@paragraph@kw}
7745     \else
7746         \def\__notesslideslabel{}
7747         \def\currentsectionlevel{\omdoc@paragraph@kw}
7748     \fi% end ifcase
7749     \__notesslideslabel\quad #2%
7750 }%
7751 \vfill%
7752 \end{frame}%
7753 }
7754 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7755     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7756 }
7757 }{}
7758 }

```

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

```

7759 \def\inserttheorembodyfont{\normalfont}
7760 %\bool_if:NF \c__notesslides_notes_bool {
7761 % \defbeamertemplate{theorem begin}{miko}
7762 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7763 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7764 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7765 % \defbeamertemplate{theorem end}{miko}{\fi}

```

and we set it as the default one.

```

7766 % \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.

```

7767 % \expandafter\def\csname Parent2\endcsname{}
7768 %}
7769
7770 \AddToHook{begindocument}{% this does not work for some reason
7771     \setbeamertemplate{theorems}[ams style]
7772 }
7773 \bool_if:NT \c__notesslides_notes_bool {
7774     \renewenvironment{columns}[1][\fi]{}{}

```

```

7775 \par\noindent%
7776 \begin{minipage}%
7777 \slidewidth\centering\leavevmode%
7778 }{%
7779 \end{minipage}\par\noindent%
7780 }%
7781 \newsavebox\columnbox%
7782 \renewenvironment<>{column}[2][]{%
7783 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7784 }{%
7785 \end{minipage}\end{lrbox}\usebox\columnbox%
7786 }%
7787 }

7788 \bool_if:NTF \c__notesslides_noproblems_bool {
7789 \newenvironment{problems}{}{}
7790 }{
7791 \excludacomment{problems}
7792 }

```

38.6 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.

```

7793 \gdef\printexcursions{}
7794 \newcommand\excursionref[2]{% label, text
7795 \bool_if:NT \c__notesslides_notes_bool {
7796 \begin{sparagraph}[title=Excursion]
7797 #2 \sref[fallback=the appendix]{#1}.
7798 \end{sparagraph}
7799 }
7800 }
7801 \newcommand\activate@excursion[2][{}{
7802 \gappto\printexcursions{\inputref{#1}{#2}}
7803 }
7804 \newcommand\excursion[4][{}{ repos, label, path, text
7805 \bool_if:NT \c__notesslides_notes_bool {
7806 \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7807 }
7808 }

```

(End definition for `\excursion`. This function is documented on page 62.)

\excursiongroup

```

7809 \keys_define:nn{notesslides / excursiongroup }{
7810 id .str_set_x:N = \l__notesslides_excursion_id_str,
7811 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
7812 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7813 }
7814 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7815 \tl_clear:N \l__notesslides_excursion_intro_tl
7816 \str_clear:N \l__notesslides_excursion_id_str

```

```

7817 \str_clear:N \l__notesslides_excursion_mhrepos_str
7818 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7819 }
7820 \newcommand\excursionsgroup[1][]{
7821   \__notesslides_excursion_args:n{ #1 }
7822   \ifdefempty\printexcursions{}% only if there are excursions
7823   {\begin{note}
7824     \begin{sfragment}[#1]{Excursions}%
7825     \ifdefempty\l__notesslides_excursion_intro_tl}{
7826       \inputref[\l__notesslides_excursion_mhrepos_str]{
7827         \l__notesslides_excursion_intro_tl
7828       }
7829     }
7830     \printexcursions%
7831     \end{sfragment}
7832   \end{note}}
7833 }
7834 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7835 \end{package}

```

(End definition for \excursionsgroup. This function is documented on page 62.)

Chapter 39

The Implementation

39.1 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.

```
7836 <*package>
7837 <@@=problems>
7838 \ProvidesExplPackage{problem}{2022/08/08}{3.2.0}{Semantic Markup for Problems}
7839 \RequirePackage{13keys2e}
7840 \RequirePackage{amssymb}% for \Box
7841
7842 \keys_define:nn { problem / pkg }{
7843   notes      .default:n    = { true },
7844   notes      .bool_set:N   = \c__problems_notes_bool,
7845   gnotes     .default:n    = { true },
7846   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7847   hints      .default:n    = { true },
7848   hints      .bool_set:N   = \c__problems_hints_bool,
7849   solutions  .default:n    = { true },
7850   solutions  .bool_set:N   = \c__problems_solutions_bool,
7851   pts        .default:n    = { true },
7852   pts        .bool_set:N   = \c__problems_pts_bool,
7853   min        .default:n    = { true },
7854   min        .bool_set:N   = \c__problems_min_bool,
7855   boxed      .default:n    = { true },
7856   boxed      .bool_set:N   = \c__problems_boxed_bool,
7857   unknown    .code:n       = {
7858     \PassOptionsToPackage{\CurrentOption}{stex}
7859   }
7860 }
7861 \newif\ifsolutions
7862
7863 \ProcessKeysOptions{ problem / pkg }
7864 \bool_if:NTF \c__problems_solutions_bool {
7865   \solutionstrue
7866 }{
7867   \solutionsfalse
```

```

7868 }
7869 \RequirePackage{stex}

```

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

```

7870 \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.

```

7871 \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.

```

7872 \def\prob@problem@kw{Problem}
7873 \def\prob@solution@kw{Solution}
7874 \def\prob@hint@kw{Hint}
7875 \def\prob@note@kw{Note}
7876 \def\prob@gnote@kw{Grading}
7877 \def\prob@pt@kw{pt}
7878 \def\prob@min@kw{min}
7879 \def\prob@correct@kw{Correct}
7880 \def\prob@wrong@kw{Wrong}

```

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

For the other languages, we set up triggers

```

7881 \AddToHook{begindocument}{
7882   \ltx@ifpackageloaded{babel}{
7883     \makeatletter
7884     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7885     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7886       \input{problem-ngerman.ldf}
7887     }
7888     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
7889       \input{problem-finnish.ldf}
7890     }
7891     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
7892       \input{problem-french.ldf}
7893     }
7894     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
7895       \input{problem-russian.ldf}
7896     }
7897     \makeatother
7898   }{}
7899 }

```

39.2 Problems and Solutions

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

```

7900 \keys_define:nn{ problem / problem }{
7901   id      .str_set_x:N = \l__problems_prob_id_str,
7902   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7903   min     .tl_set:N    = \l__problems_prob_min_tl,

```

```

7904 title .tl_set:N = \l__problems_prob_title_tl,
7905 type .tl_set:N = \l__problems_prob_type_tl,
7906 imports .tl_set:N = \l__problems_prob_imports_tl,
7907 name .str_set_x:N = \l__problems_prob_name_str,
7908 refnum .int_set:N = \l__problems_prob_refnum_int
7909 }
7910 \cs_new_protected:Nn \l__problems_prob_args:n {
7911 \str_clear:N \l__problems_prob_id_str
7912 \str_clear:N \l__problems_prob_name_str
7913 \tl_clear:N \l__problems_prob_pts_tl
7914 \tl_clear:N \l__problems_prob_min_tl
7915 \tl_clear:N \l__problems_prob_title_tl
7916 \tl_clear:N \l__problems_prob_type_tl
7917 \tl_clear:N \l__problems_prob_imports_tl
7918 \int_zero_new:N \l__problems_prob_refnum_int
7919 \keys_set:nn { problem / problem }{ #1 }
7920 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7921 \let\l__problems_prob_refnum_int\undefined
7922 }
7923 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7924 \newcounter{problem}[section]
7925 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
7926 \def\theplainsproblem{\arabic{problem}}
7927 \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.

```

7928 \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

```

7929 \newcommand\prob@number{
7930 \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7931 \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7932 }{
7933 \int_if_exist:NTF \l__problems_prob_refnum_int {
7934 \prob@label{\int_use:N \l__problems_prob_refnum_int }
7935 }{
7936 \prob@label\theplainsproblem
7937 }
7938 }
7939 }
7940 \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.

```

7941 \newcommand\prob@title[3]{%
7942   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7943     #2 \l__problems_inclprob_title_tl #3
7944   }{
7945     \tl_if_empty:NTF \l__problems_prob_title_tl {
7946       #1
7947     }{
7948       #2 \l__problems_prob_title_tl #3
7949     }
7950   }
7951 }

```

(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.

```

7952 \def\prob@heading{
7953   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7954   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
7955 }

```

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

With this in place, we can now define the **problem** environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whether the respective options are set.

sproblem (env.)

```

7956 \newenvironment{sproblem}[1][]{
7957   \__problems_prob_args:n{#1}%\sref@target%
7958   \@in@omtexttrue% we are in a statement (for inline definitions)
7959   \refstepcounter{sproblem}\record@problem
7960   \def\current@section@level{\prob@problem@kw}
7961
7962   \str_if_empty:NT \l__problems_prob_name_str {
7963     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7964     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7965     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7966   }
7967
7968   \stex_if_do_html:T{
7969     \tl_if_empty:NF \l__problems_prob_title_tl {
7970       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7971     }
7972   }
7973
7974   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7975
7976   \stex_reactivate_macro:N \STEXexport
7977   \stex_reactivate_macro:N \importmodule
7978   \stex_reactivate_macro:N \symdecl
7979   \stex_reactivate_macro:N \notation
7980   \stex_reactivate_macro:N \symdef

```

```

7981 \stex_if_do_html:T{
7982   \begin{stex_annotate_env} {problem} {
7983     \l_stex_module_ns_str ? \l_stex_module_name_str
7984   }
7985 }
7986
7987 \stex_annotate_invisible:nnn{header}{} {
7988   \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
7989   \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7990   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7991     \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7992   }
7993 }
7994 }
7995
7996 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7997
7998
7999 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
8000   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
8001 }{
8002   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
8003 }
8004 \str_if_exist:NTF \l__problems_inclprob_id_str {
8005   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
8006 }{
8007   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
8008 }
8009
8010
8011 \stex_if_smsmode:F {
8012   \clist_set:No \l_tmpa_clist \sproblemtype
8013   \tl_clear:N \l_tmpa_tl
8014   \clist_map_inline:Nn \l_tmpa_clist {
8015     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8016       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8017     }
8018   }
8019   \tl_if_empty:NTF \l_tmpa_tl {
8020     \__problems_sproblem_start:
8021   }{
8022     \l_tmpa_tl
8023   }
8024 }
8025 \stex_ref_new_doc_target:n \sproblemid
8026 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8027 }{
8028   \__stex_modules_end_module:
8029   \stex_if_smsmode:F{
8030     \clist_set:No \l_tmpa_clist \sproblemtype
8031     \tl_clear:N \l_tmpa_tl
8032     \clist_map_inline:Nn \l_tmpa_clist {
8033       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8034         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}

```



```

8035     }
8036   }
8037   \tl_if_empty:NTF \l_tmpa_tl {
8038     \__problems_sproblem_end:
8039   }{
8040     \l_tmpa_tl
8041   }
8042 }
8043 \stex_if_do_html:T{
8044   \end{stex_annotate_env}
8045 }
8046
8047 \smallskip
8048 }
8049
8050 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8051
8052
8053
8054 \cs_new_protected:Nn \__problems_sproblem_start: {
8055   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
8056 }
8057 \cs_new_protected:Nn \__problems_sproblem_end: { \par\smallskip}
8058
8059 \newcommand\stexpatchproblem[3][] {
8060   \str_set:Nx \l_tmpa_str{ #1 }
8061   \str_if_empty:NTF \l_tmpa_str {
8062     \tl_set:Nn \__problems_sproblem_start: { #2 }
8063     \tl_set:Nn \__problems_sproblem_end: { #3 }
8064   }{
8065     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8066     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8067   }
8068 }
8069
8070
8071 \bool_if:NT \c__problems_boxed_bool {
8072   \surroundwithhmdframed{problem}
8073 }

```

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

```

8074 \def\record@problem{
8075   \protected@write\@auxout{}
8076   {
8077     \string\@problem{\prob@number}
8078     {
8079       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8080         \l__problems_inclprob_pts_tl
8081       }{
8082         \l__problems_prob_pts_tl
8083       }
8084     }%
8085     {
8086       \tl_if_exist:NTF \l__problems_inclprob_min_tl {

```

```

8087         \l__problems_inclprob_min_tl
8088     }{
8089         \l__problems_prob_min_tl
8090     }
8091 }
8092 }
8093 }

```

(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).

```

8094 \def\@problem#1#2#3{

```

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

solution (env.) The solution environment is similar to the problem environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```

8095 \keys_define:nn { problem / solution }{
8096   id          .str_set_x:N = \l__problems_solution_id_str ,
8097   for         .str_set_x:N = \l__problems_solution_for_str ,
8098   type        .str_set_x:N = \l__problems_solution_type_str ,
8099   title       .tl_set:N     = \l__problems_solution_title_tl
8100 }
8101 \cs_new_protected:Nn \__problems_solution_args:n {
8102   \str_clear:N \l__problems_solution_id_str
8103   \str_clear:N \l__problems_solution_type_str
8104   \str_clear:N \l__problems_solution_for_str
8105   \tl_clear:N \l__problems_solution_title_tl
8106   \keys_set:nn { problem / solution }{ #1 }
8107 }

```

\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.

```

8108 \box_new:N \l__problems_solution_box
8109 \newenvironment{solution}[1][{}{
8110   \__problems_solution_args:n{#1}
8111   \stex_html_backend:TF{
8112     \stex_if_do_html:T{
8113       \begin{stex_annotate_env}{solution}{}
8114       \str_if_empty:NF \l__problems_solution_type_str {
8115         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
8116       }
8117       \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8118     }
8119   }{
8120     \setbox\l__problems_solution_box\vbox\bgroup
8121     \par\smallskip\hrule\smallskip
8122     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems_
8123   }
8124 }{
8125   \stex_html_backend:TF{
8126     \stex_if_do_html:T{
8127       \end{stex_annotate_env}

```

```

8128     }
8129   }{
8130     \smallskip\hrule
8131     \egroup
8132     \bool_if:NT \c__problems_solutions_bool {
8133       \box\l__problems_solution_box
8134     }
8135   }
8136 }
8137
8138 \newcommand\startsolutions{
8139   \bool_set_true:N \c__problems_solutions_bool
8140   \solutionstrue
8141   % \specialcomment{solution}{\@startsolution}{
8142   %   \bool_if:NF \c__problems_boxed_bool {
8143   %     \hrule\medskip
8144   %   }
8145   %   \end{small}%
8146   % }
8147   % \bool_if:NT \c__problems_boxed_bool {
8148   %   \surroundwithmdframed{solution}
8149   % }
8150 }

```

(End definition for \startsolutions. This function is documented on page 64.)

\stopsolutions

```

8151 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

(End definition for \stopsolutions. This function is documented on page 64.)

exnote (env.)

```

8152 \bool_if:NTF \c__problems_notes_bool {
8153   \newenvironment{exnote}[1][]{
8154     \par\smallskip\hrule\smallskip
8155     \noindent\textbf{\prob@note@kw :~ }\small
8156   }{
8157     \smallskip\hrule
8158   }
8159 }{
8160   \excludacomment{exnote}
8161 }

```

hint (env.)

```

8162 \bool_if:NTF \c__problems_notes_bool {
8163   \newenvironment{hint}[1][]{
8164     \par\smallskip\hrule\smallskip
8165     \noindent\textbf{\prob@hint@kw :~ }\small
8166   }{
8167     \smallskip\hrule
8168   }
8169 \newenvironment{exhint}[1][]{
8170   \par\smallskip\hrule\smallskip
8171   \noindent\textbf{\prob@hint@kw :~ }\small

```

```

8172 }{
8173   \smallskip\hrule
8174 }
8175 }{
8176   \excludecomment{hint}
8177   \excludecomment{exhint}
8178 }

```

gnote (env.)

```

8179 \bool_if:NTF \c__problems_notes_bool {
8180   \newenvironment{gnote}[1][]{
8181     \par\smallskip\hrule\smallskip
8182     \noindent\textbf{\prob@gnote@kw :~ } \small
8183   }{
8184     \smallskip\hrule
8185   }
8186 }{
8187   \excludecomment{gnote}
8188 }

```

39.3 Marup for Added Value Services

39.4 Multiple Choice Blocks

EdN:12

mcb (env.) ¹²

```

8189 \newenvironment{mcb}{
8190   \begin{enumerate}
8191 }{
8192   \end{enumerate}
8193 }

```

we define the keys for the mcb macro

```

8194 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8195   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8196     \bool_set_true:N #1
8197   }{
8198     \bool_set_false:N #1
8199   }
8200 }
8201 \keys_define:nn { problem / mcb }{
8202   id          .str_set_x:N = \l__problems_mcc_id_str ,
8203   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
8204   T           .default:n    = { false } ,
8205   T           .bool_set:N   = \l__problems_mcc_t_bool ,
8206   F           .default:n    = { false } ,
8207   F           .bool_set:N   = \l__problems_mcc_f_bool ,
8208   Ttext       .tl_set:N     = \l__problems_mcc_Ttext_tl ,
8209   Ftext       .tl_set:N     = \l__problems_mcc_Ftext_tl
8210 }
8211 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

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

```

8212 \str_clear:N \l__problems_mcc_id_str
8213 \tl_clear:N \l__problems_mcc_feedback_tl
8214 \bool_set_false:N \l__problems_mcc_t_bool
8215 \bool_set_false:N \l__problems_mcc_f_bool
8216 \tl_clear:N \l__problems_mcc_Ttext_tl
8217 \tl_clear:N \l__problems_mcc_Ftext_tl
8218 \str_clear:N \l__problems_mcc_id_str
8219 \keys_set:nn { problem / mcc }{ #1 }
8220 }

```

\mcc

```

8221 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8222 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8223 \newcommand\mcc[2][] {
8224   \l__problems_mcc_args:n{ #1 }
8225   \item[$\Box$] #2
8226   \bool_if:NT \c__problems_solutions_bool{
8227     \\\
8228     \bool_if:NT \l__problems_mcc_t_bool {
8229       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8230     }
8231     \bool_if:NT \l__problems_mcc_f_bool {
8232       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8233     }
8234     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8235       \emph{\l__problems_mcc_feedback_tl}
8236     }
8237   }
8238 } %solutions

```

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

39.5 Filling in Concrete Solutions

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

```

8239 \newcommand\fillinsol[1]{\quad%
8240   \ifsolutions\textcolor{red}{\#1!}\else%
8241   \fbox{\phantom{\huge{\#1}}}%
8242   \fi}

```

(End definition for \includeproblem. This function is documented on page 67.)

39.6 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.

```

8243
8244 \keys_define:nn{ problem / inclproblem }{
8245   id      .str_set_x:N = \l__problems_inclprob_id_str,
8246   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8247   min     .tl_set:N    = \l__problems_inclprob_min_tl,

```

```

8248 title .tl_set:N = \l__problems_inclprob_title_tl,
8249 refnum .int_set:N = \l__problems_inclprob_refnum_int,
8250 type .tl_set:N = \l__problems_inclprob_type_tl,
8251 mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8252 }
8253 \cs_new_protected:Nn \l__problems_inclprob_args:n {
8254 \str_clear:N \l__problems_prob_id_str
8255 \tl_clear:N \l__problems_inclprob_pts_tl
8256 \tl_clear:N \l__problems_inclprob_min_tl
8257 \tl_clear:N \l__problems_inclprob_title_tl
8258 \tl_clear:N \l__problems_inclprob_type_tl
8259 \int_zero_new:N \l__problems_inclprob_refnum_int
8260 \str_clear:N \l__problems_inclprob_mhrepos_str
8261 \keys_set:nn { problem / inclproblem }{ #1 }
8262 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8263 \let\l__problems_inclprob_pts_tl\undefined
8264 }
8265 \tl_if_empty:NT \l__problems_inclprob_min_tl {
8266 \let\l__problems_inclprob_min_tl\undefined
8267 }
8268 \tl_if_empty:NT \l__problems_inclprob_title_tl {
8269 \let\l__problems_inclprob_title_tl\undefined
8270 }
8271 \tl_if_empty:NT \l__problems_inclprob_type_tl {
8272 \let\l__problems_inclprob_type_tl\undefined
8273 }
8274 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8275 \let\l__problems_inclprob_refnum_int\undefined
8276 }
8277 }
8278
8279 \cs_new_protected:Nn \l__problems_inclprob_clear: {
8280 \let\l__problems_inclprob_id_str\undefined
8281 \let\l__problems_inclprob_pts_tl\undefined
8282 \let\l__problems_inclprob_min_tl\undefined
8283 \let\l__problems_inclprob_title_tl\undefined
8284 \let\l__problems_inclprob_type_tl\undefined
8285 \let\l__problems_inclprob_refnum_int\undefined
8286 \let\l__problems_inclprob_mhrepos_str\undefined
8287 }
8288 \l__problems_inclprob_clear:
8289
8290 \newcommand\includeproblem[2][]{
8291 \l__problems_inclprob_args:n{ #1 }
8292 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8293 \stex_html_backend:TF {
8294 \str_clear:N \l_tmpa_str
8295 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8296 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8297 }
8298 \stex_annotate_invisible:nnn{includeproblem}{
8299 \l_tmpa_str / #2
8300 }{}}
8301 }{

```

```

8302     \begingroup
8303     \inputreftrue
8304     \tl_if_empty:nTF{ ##1 }{
8305       \input{#2}
8306     }{
8307       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8308     }
8309   \endgroup
8310 }
8311 }
8312 \__problems_inclprob_clear:
8313 }

```

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

39.7 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.

```

8314 \AddToHook{enddocument}{
8315   \bool_if:NT \c__problems_pts_bool {
8316     \message{Total:~\arabic{pts}~points}
8317   }
8318   \bool_if:NT \c__problems_min_bool {
8319     \message{Total:~\arabic{min}~minutes}
8320   }
8321 }

```

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

```

8322 \def\pts#1{
8323   \bool_if:NT \c__problems_pts_bool {
8324     \marginpar{#1~\prob@pt@kw}
8325   }
8326 }
8327 \def\min#1{
8328   \bool_if:NT \c__problems_min_bool {
8329     \marginpar{#1~\prob@min@kw}
8330   }
8331 }

```

\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.

```

8332 \newcounter{pts}
8333 \def\show@pts{
8334   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8335     \bool_if:NT \c__problems_pts_bool {
8336       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8337       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8338     }
8339   }{
8340     \tl_if_exist:NT \l__problems_prob_pts_tl {
8341       \bool_if:NT \c__problems_pts_bool {

```

```

8342         \tl_if_empty:NT\l__problems_prob_pts_tl{
8343             \tl_set:Nn \l__problems_prob_pts_tl {0}
8344         }
8345         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8346         \addtocounter{pts}{\l__problems_prob_pts_tl}
8347     }
8348 }
8349 }
8350 }

```

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

\show@min

```

8351 \newcounter{min}
8352 \def\show@min{
8353     \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8354         \bool_if:NT \c__problems_min_bool {
8355             \marginpar{\l__problems_inclprob_pts_tl\ min}
8356             \addtocounter{min}{\l__problems_inclprob_min_tl}
8357         }
8358     }{
8359         \tl_if_exist:NT \l__problems_prob_min_tl {
8360             \bool_if:NT \c__problems_min_bool {
8361                 \tl_if_empty:NT\l__problems_prob_min_tl{
8362                     \tl_set:Nn \l__problems_prob_min_tl {0}
8363                 }
8364                 \marginpar{\l__problems_prob_min_tl\ min}
8365                 \addtocounter{min}{\l__problems_prob_min_tl}
8366             }
8367         }
8368     }
8369 }
8370 \end{package}

```

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

Chapter 40

Implementation: The hwexam Package

40.1 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.

```
8371 {*package}
8372 \ProvidesExplPackage{hwexam}{2022/08/08}{3.2.0}{homework assignments and exams}
8373 \RequirePackage{13keys2e}
8374
8375 \newif\iftest\testfalse
8376 \DeclareOption{test}{\testtrue}
8377 \newif\ifmultiple\multiplefalse
8378 \DeclareOption{multiple}{\multipletrue}
8379 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8380 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8381 \ProcessOptions
```

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

```
8382 \RequirePackage{keyval}[1997/11/10]
8383 \RequirePackage{problem}
```

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

```
8384 \newcommand\hwexam@assignment@kw{Assignment}
8385 \newcommand\hwexam@given@kw{Given}
8386 \newcommand\hwexam@due@kw{Due}
8387 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8388 \newcommand\hwexam@minutes@kw{minutes}
8389 \newcommand\correction@probs@kw{prob.}
8390 \newcommand\correction@pts@kw{total}
8391 \newcommand\correction@reached@kw{reached}
8392 \newcommand\correction@sum@kw{Sum}
8393 \newcommand\correction@grade@kw{grade}
8394 \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

```

8395 \AddToHook{begindocument}{
8396 \ltx@ifpackageloaded{babel}{
8397 \makeatletter
8398 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8399 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8400 \input{hwexam-ngerman.ldf}
8401 }
8402 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8403 \input{hwexam-finnish.ldf}
8404 }
8405 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8406 \input{hwexam-french.ldf}
8407 }
8408 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8409 \input{hwexam-russian.ldf}
8410 }
8411 \makeatother
8412 }{}
8413 }
8414

```

40.2 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.

```

8415 \newcounter{assignment}
8416 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8417 \keys_define:nn { hwexam / assignment } {
8418 id .str_set:N = \l_@@_assign_id_str,
8419 number .int_set:N = \l_@@_assign_number_int,
8420 title .tl_set:N = \l_@@_assign_title_tl,
8421 type .tl_set:N = \l_@@_assign_type_tl,
8422 given .tl_set:N = \l_@@_assign_given_tl,
8423 due .tl_set:N = \l_@@_assign_due_tl,
8424 loadmodules .code:n = {
8425 \bool_set_true:N \l_@@_assign_loadmodules_bool
8426 }
8427 }
8428 \cs_new_protected:Nn \_@@_assignment_args:n {
8429 \str_clear:N \l_@@_assign_id_str
8430 \int_set:Nn \l_@@_assign_number_int {-1}
8431 \tl_clear:N \l_@@_assign_title_tl
8432 \tl_clear:N \l_@@_assign_type_tl
8433 \tl_clear:N \l_@@_assign_given_tl
8434 \tl_clear:N \l_@@_assign_due_tl
8435 \bool_set_false:N \l_@@_assign_loadmodules_bool
8436 \keys_set:nn { hwexam / assignment }{ #1 }
8437 }

```

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.

```

8438 \newcommand\given@due[2]{
8439 \bool_lazy_all:nF {
8440 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8441 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8442 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8443 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8444 }{ #1 }
8445
8446 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8447 \tl_if_empty:NF \l_@@_assign_given_tl {
8448 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8449 }
8450 }{
8451 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8452 }
8453
8454 \bool_lazy_or:nnF {
8455 \bool_lazy_and_p:nn {
8456 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8457 }{
8458 \tl_if_empty_p:V \l_@@_assign_due_tl
8459 }
8460 }{
8461 \bool_lazy_and_p:nn {
8462 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8463 }{
8464 \tl_if_empty_p:V \l_@@_assign_due_tl
8465 }
8466 }{ ,~ }
8467
8468 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8469 \tl_if_empty:NF \l_@@_assign_due_tl {
8470 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8471 }
8472 }{
8473 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8474 }
8475
8476 \bool_lazy_all:nF {
8477 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8478 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8479 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8480 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8481 }{ #2 }
8482 }

```

`\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.

```

8483 \newcommand\assignment@title[3]{
8484 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8485 \tl_if_empty:NTF \l_@@_assign_title_tl {
8486 #1
8487 }{
8488 #2\l_@@_assign_title_tl#3
8489 }
8490 }{
8491 #2\l_@@_inclasssign_title_tl#3
8492 }
8493 }

```

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

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

```

8494 \newcommand\assignment@number{
8495 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8496 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8497 \arabic{assignment}
8498 } {
8499 \int_use:N \l_@@_assign_number_int
8500 }
8501 }{
8502 \int_use:N \l_@@_inclasssign_number_int
8503 }
8504 }

```

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

With them, we can define the central **assignment** environment. This has two forms (separated by \ifmultiple) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

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

```

8505 \newenvironment{assignment}[1][]{
8506 \_@@_assignment_args:n { #1 }
8507 %\sref@target
8508 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8509 \global\stepcounter{assignment}
8510 }{
8511 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8512 }
8513 \setcounter{sproblem}{0}
8514 \renewcommand\prob@label[1]{\assignment@number.##1}
8515 \def\current@section@level{\document@hwexamtype}
8516 %\sref@label{id{\document@hwexamtype \thesection}
8517 \begin{@assignment}
8518 }{
8519 \end{@assignment}
8520 }

```

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

```

8521 \def\ass@title{
8522 {\protect\document@hwexamtype}\arabic{assignment}
8523 \assignment@title{}\;\;{}{}\; -- \given@due{}\;}
8524 }
8525 \ifmultiple
8526 \newenvironment{@assignment}{
8527 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8528 \begin{sfragment}[loadmodules]{\ass@title}
8529 }{
8530 \begin{sfragment}{\ass@title}
8531 }
8532 }{
8533 \end{sfragment}
8534 }

```

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

```

8535 \else
8536 \newenvironment{@assignment}{
8537 \begin{center}\bf
8538 \Large@title\strut\
8539 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;\;{}{}\;
8540 \large\given@due{--\;\;{}{}\;}\;
8541 \end{center}
8542 }{}
8543 \fi% multiple

```

40.3 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.

```

8544 \keys_define:nn { hwexam / inclassignment } {
8545 %id .str_set_x:N = \l_@@_assign_id_str,
8546 number .int_set:N = \l_@@_inclassign_number_int,
8547 title .tl_set:N = \l_@@_inclassign_title_tl,
8548 type .tl_set:N = \l_@@_inclassign_type_tl,
8549 given .tl_set:N = \l_@@_inclassign_given_tl,
8550 due .tl_set:N = \l_@@_inclassign_due_tl,
8551 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8552 }
8553 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8554 \int_set:Nn \l_@@_inclassign_number_int {-1}
8555 \tl_clear:N \l_@@_inclassign_title_tl
8556 \tl_clear:N \l_@@_inclassign_type_tl
8557 \tl_clear:N \l_@@_inclassign_given_tl
8558 \tl_clear:N \l_@@_inclassign_due_tl
8559 \str_clear:N \l_@@_inclassign_mhrepos_str
8560 \keys_set:nn { hwexam / inclassignment }{ #1 }
8561 }
8562 \l_@@_inclassignment_args:n {}
8563
8564 \newcommand\inputassignment[2][]{

```

```

8565 \_@@_inclassignment_args:n { #1 }
8566 \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8567   \input{#2}
8568 }{
8569   \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8570     \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
8571   }
8572 }
8573 \_@@_inclassignment_args:n {}
8574 }
8575 \newcommand\includeassignment[2][]{
8576   \newpage
8577   \inputassignment[#1]{#2}
8578 }

```

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

40.4 Typesetting Exams

\quizheading

```

8579 \ExplSyntaxOff
8580 \newcommand\quizheading[1]{%
8581   \def\@tas{#1}%
8582   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8583   \ifx\@tas\@empty\else%
8584     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8585   \fi%
8586 }
8587 \ExplSyntaxOn

```

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

\testheading

```

8588
8589 \def\hwexamheader{\input{hwexam-default.header}}
8590
8591 \def\hwexamminutes{
8592   \tl_if_empty:NTF \testheading@duration {
8593     {\testheading@min}~\hwexam@minutes@kw
8594   }{
8595     \testheading@duration
8596   }
8597 }
8598
8599 \keys_define:nn { hwexam / testheading } {
8600   min .tl_set:N = \testheading@min,
8601   duration .tl_set:N = \testheading@duration,
8602   reqpts .tl_set:N = \testheading@reqpts,
8603   tools .tl_set:N = \testheading@tools
8604 }
8605 \cs_new_protected:Nn \_@@_testheading_args:n {
8606   \tl_clear:N \testheading@min
8607   \tl_clear:N \testheading@duration

```

```

8608 \tl_clear:N \testheading@reqpts
8609 \tl_clear:N \testheading@tools
8610 \keys_set:nn { hwexam / testheading }{ #1 }
8611 }
8612 \newenvironment{testheading}[1][]{
8613 \_@@_testheading_args:n{ #1 }
8614 \newcount\check@time\check@time=\testheading@min
8615 \advance\check@time by -\theassignment@totalmin
8616 \newif\if@bonuspoints
8617 \tl_if_empty:NTF \testheading@reqpts {
8618 \@bonuspointsfalse
8619 }{
8620 \newcount\bonus@pts
8621 \bonus@pts=\theassignment@totalpts
8622 \advance\bonus@pts by -\testheading@reqpts
8623 \edef\bonus@pts{\the\bonus@pts}
8624 \@bonuspointstrue
8625 }
8626 \edef\check@time{\the\check@time}
8627
8628 \makeatletter\hwexamheader\makeatother
8629 }{
8630 \newpage
8631 }

```

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

\testspace

```

8632 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8633 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8634 \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.

```

8635 <@=problems>
8636 \renewcommand\@problem[3]{
8637 \stepcounter{assignment@probs}
8638 \def\__problemspts{#2}
8639 \ifx\__problemspts\@empty\else
8640 \addtocounter{assignment@totalpts}{#2}
8641 \fi
8642 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8643 \xdef\correction@probs{\correction@probs & #1}%
8644 \xdef\correction@pts{\correction@pts & #2}
8645 \xdef\correction@reached{\correction@reached &}

```

```

8646 }
8647 <@@=hwexam>

```

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

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

```

8648 \newcounter{assignment@probs}
8649 \newcounter{assignment@totalpts}
8650 \newcounter{assignment@totalmin}
8651 \def\correction@probs{\correction@probs@kw}
8652 \def\correction@pts{\correction@pts@kw}
8653 \def\correction@reached{\correction@reached@kw}
8654 \stepcounter{assignment@probs}
8655 \newcommand\correction@table{
8656 \resizebox{\textwidth}{!}{%
8657 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8658 &\multicolumn{\theassignment@probs}{c|}|%|
8659 {\footnotesize\correction@forgrading@kw} &\\ \hline
8660 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8661 \correction@pts & \theassignment@totalpts & \\ \hline
8662 \correction@reached & & \[.7cm]\hline
8663 \end{tabular}}
8664 </package>

```

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

40.5 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 41

References

EdN:13

13

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T_EX distribution. The Comprehensive T_EX Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphicx.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

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).