

The sTeX3 Package Collection *

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

2022-08-07

Abstract

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

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

sTeX augments L^AT_EX with

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

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

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

*Version 3.1 (last revised 2022-08-07)

Contents

I	Manual	1
1	What is sTeX?	2
2	Quickstart	3
2.1	Setup	3
2.1.1	Minimal Setup for the PDF-only Workflow	3
2.1.2	GIT-based Setup for the sTeX Development Version	3
2.1.3	sTeX Archives (Manual Setup)	4
2.1.4	The sTeX IDE	4
2.1.5	Manual Setup for Active Documents and Knowledge Management Services	4
2.2	A First sTeX Document	5
2.2.1	OMDoc/xhtml Conversion	8
2.2.2	MMT/OMDoc Conversion	9
3	Creating sTeX Content	10
3.1	How Knowledge is Organized in sTeX	10
3.2	sTeX Archives	11
3.2.1	The Local MathHub-Directory	11
3.2.2	The Structure of sTeX Archives	12
3.2.3	MANIFEST.MF-Files	12
3.2.4	Using Files in sTeX Archives Directly	13
3.3	Module, Symbol and Notation Declarations	14
3.3.1	The smodule-Environment	14
3.3.2	Declaring New Symbols and Notations	16
3.3.3	Operator Notations	19
3.3.3	Argument Modes	20
3.3.3	Mode-b Arguments	20
3.3.3	Mode-a Arguments	21
3.3.3	Mode-B Arguments	22
3.3.4	Type and Definiens Components	23
3.3.5	Precedences and Automated Bracketing	24
3.3.6	Variables	26
3.3.7	Variable Sequences	27
3.4	Module Inheritance and Structures	28
3.4.1	Multilinguality and Translations	28
3.4.2	Simple Inheritance and Namespaces	29
3.4.3	The mathstructure Environment	31
3.4.4	The copymodule Environment	34
3.4.5	The interpretmodule Environment	35
3.5	Primitive Symbols (The sTeX Metatheory)	36
4	Using sTeX Symbols	37
4.1	\symref and its variants	37
4.2	Marking Up Text and On-the-Fly Notations	38

5	sTeX Statements	42
5.1	Definitions, Theorems, Examples, Paragraphs	42
5.2	Proofs	45
5.3	Highlighting and Presentation Customizations	50
6	Cross References	52
7	Additional Packages	54
7.1	Tikzinput: Treating TIKZ code as images	54
7.2	Modular Document Structuring	55
7.2.1	Introduction	55
7.2.2	Package Options	55
7.2.3	Document Fragments	55
7.2.4	Ending Documents Prematurely	57
7.2.5	Global Document Variables	57
7.3	Slides and Course Notes	57
7.3.1	Introduction	57
7.3.2	Package Options	58
7.3.3	Notes and Slides	58
7.3.4	Customizing Header and Footer Lines	59
7.3.5	Frame Images	60
7.3.6	Excursions	61
7.4	Representing Problems and Solutions	62
7.4.1	Introduction	62
7.4.2	Problems and Solutions	62
7.4.3	Markup for Added-Value Services	64
	Multiple Choice Blocks	64
	Filling-In Concrete Solutions	65
7.4.4	Including Problems	66
7.5	Homeworks, Quizzes and Exams	67
7.5.1	Introduction	67
7.5.2	Package Options	67
7.5.3	Assignments	67
7.5.4	Including Assignments	68
7.5.5	Typesetting Exams	68
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 $\text{\texttt{STeX}}$ concept relates to the MMT/OMDoc system, philosophy or language; see [MMT; Koh06] for introductions.

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

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

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

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

2.1.1 Minimal Setup for the PDF-only Workflow

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.1.5 Manual Setup for Active Documents and Knowledge Management Services

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

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

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

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

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

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

2.2 A First \TeX Document

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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

Remark 2.2.1:

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

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

```
smodule \begin{smodule}{GeometricSeries}
...
\end{smodule}
```

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

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

```
\importmodule \importmodule[snglom/calculus]{series}
\importmodule \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`
`\definiendum`

The `\define{geometricSeries} ...`

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

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

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

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

2.2.1 OMDoc/xhtml Conversion

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

TODO VSCode Plugin

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

```
<mrow resource="" property="stex:definiens">
  <mrow resource="...?series?infinitesum" property="stex:OMBIND">
    <munderover displaystyle="true">
      <mo resource="...?series?infinitesum" property="stex:comp">\Sigma</mo>
      <mrow>
        <mrow resource="1" property="stex:arg">
          <mi resource="var://n" property="stex:OMV">n</mi>
        </mrow>
        <mo resource="...?series?infinitesum" property="stex:comp">=</mo>
        <mi resource="2" property="stex:arg">1</mi>
      </mrow>
      <mi resource="...?series?infinitesum" property="stex:comp">\infty</mi>
    </munderover>
    <mrow resource="3" property="stex:arg">
      <mfrac resource="...?realarith?division#frac#" property="stex:OMA">
        <mi resource="1" property="stex:arg">1</mi>
        <mrow resource="2" property="stex:arg">
          <msup resource="...realarith?exponentiation" property="stex:OMA">
            <mi resource="1" property="stex:arg">2</mi>
            <mrow resource="2" property="stex:arg">
              <mi resource="var://n" property="stex:OMV">n</mi>
            </mrow>
          </msup>
        </mrow>
      </mfrac>
    </mrow>
  </mrow>
</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 $\text{\textbackslash symdecl}\{\text{symbolname}\}$, $\text{\textbackslash symdef}\{\text{symbolname}\}$ and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro* $\text{\textbackslash symbolname}$ generated by symbol declarations.
4. $\text{\S}\text{\TeX}$ **expressions** finally are built up from usages of semantic macros.

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

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

3.2.1 The Local MathHub-Directory

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

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

1. If the $\text{\S}\text{\TeX}$ package is loaded with the option `mathhub=/path/to/mathhub`, then $\text{\S}\text{\TeX}$ will consider `/path/to/mathhub` as the local MathHub-directory.
2. If the `mathhub` package option is *not* set, but the macro `\mathhub` exists when the $\text{\S}\text{\TeX}$ -package is loaded, then this macro is assumed to point to the local MathHub-directory; i.e. `\def\mathhub{/path/to/mathhub}\usepackage{stex}` will set the MathHub-directory as `path/to/mathhub`.
3. Otherwise, $\text{\S}\text{\TeX}$ will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local MathHub-directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.
4. Finally, if all else fails, $\text{\S}\text{\TeX}$ will look for a file `~/.stex/mathhub.path`. If this file exists, $\text{\S}\text{\TeX}$ will assume that it contains the path to the local MathHub-directory. This method is recommended on systems where it is difficult to set environment variables.

3.2.2 The Structure of \TeX Archives

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

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

<div style="border-bottom: 1px solid black; padding-bottom: 5px;">$\backslash\text{inputref}$</div>	$\backslash\text{inputref}$ [Some/Archive]{some/file} behaves like $\backslash\text{mhinput}$, but wraps the input in a <code>\begin{group} ... \end{group}</code> . When converting to <code>xhtml</code> , the file is not input at all, and instead an html-annotation is inserted that references the file, e.g. for lazy loading.
--	---

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

<div style="border-bottom: 1px solid black; padding-bottom: 5px;">$\backslash\text{ifinput}$</div>	Both $\backslash\text{mhinput}$ and $\backslash\text{inputref}$ set $\backslash\text{ifinput}$ to “true” during input. This allows for selectively including e.g. bibliographies only if the current file is not being currently included in a larger document.
---	---

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

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

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

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

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

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

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

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

Remark 3.2.1:

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

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

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

`smodule` A new module is declared using the basic syntax

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

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

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

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

`type` ($\langle string \rangle^*$) for use in customizations.
`deprecate` ($\langle module \rangle$) if set, will throw a warning when loaded, urging to use $\langle module \rangle$ instead.
`id` ($\langle string \rangle$) for cross-referencing.
`ns` ($\langle URI \rangle$) the namespace to use. *Should not be used, unless you know precisely what you're doing.* If not explicitly set, is computed using `\stex_modules_current_namespace:`.
`lang` ($\langle language \rangle$) if not set, computed from the current file name (e.g. `foo.en.tex`).
`sig` ($\langle language \rangle$) if the current file is a translation of a file with the same base name but a different language suffix, setting `sig=<lang>` will preload the module from that language file. This helps ensuring that the (formal) content of both modules is (almost) identical across languages and avoids duplication.
`creators` ($\langle string \rangle^*$) names of the creators.
`contributors` ($\langle string \rangle^*$) names of contributors.
`srccite` ($\langle string \rangle$) a source citation for the content of this module.

\hookrightarrow An $\text{\texttt{sTeX}}$ module corresponds to an MMT/OMDOC *theory*. As such it
 \rightarrow gets assigned a module URI (*universal resource identifier*) of the form
 \rightsquigarrow `T` \rightsquigarrow `<namespace>?<module-name>`.

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

Example 1

Input:

```

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

```

Output:

Hello World

\stexpatchmodule

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

For example:

Example 2

Input:

```

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

```

Output:

```

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

```

3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

Example 3

Input:

```

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

```

Output:

```

Given a , 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:

```

is a symbol taking two arguments.

```

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

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

`\notation`

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

Example 5

Input:

```

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

```

Output:

```

First: a; Second: b

```

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

`\comp`

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

We can introduce a new notation `highlight` for `\binarysymbol` that fixes this flaw, which we can subsequently use with `\binarysymbol[highlight]`:

Example 6

Input:

```

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

```

Output:

ab



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:

ab

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 \S T E X 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 \T E X ’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 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:

is also occasionally written

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

3.3.3 Argument Modes

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

Mode-b Arguments

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

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

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

Example 9

Input:

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

Output:

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

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

Mode-a Arguments

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

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

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

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

Example 10

Input:

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

Output:

Tadaa: $a_S b_S c_S d_S e$

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: $abcde$

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

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

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

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

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

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

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

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

Mode-B Arguments

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

Example 12

Input:

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

Output:

$xyzP$

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:

is an operation

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 operation is defined as $xx1$

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:

is an operation

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:

abcde

We all know that `\comp` binds stronger than `,`, so the output *abcde* 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:

ab(cde)

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:

$ab(cde)$

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

`\infprec`
`\neginfprec`

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



More precisely, each notation takes

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

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

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

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

In the example above:

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



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

3.3.6 Variables

All symbol and notation declarations require a module with which they are associated, hence the commands $\text{\texttt{\textbackslash symdecl}}$, $\text{\texttt{\textbackslash notation}}$, $\text{\texttt{\textbackslash symdef}}$ etc. are disabled outside of $\text{\texttt{s module}}$ -environments.

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

$\text{\texttt{\textbackslash svar}}$

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

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

$\text{\texttt{\textbackslash vardef}}$

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

Example 19

Input:

```

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


Output:

Given a function f :, by fn we mean the function $xf(xn)$

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

TODO: `bind=forall/exists`

3.3.7 Variable Sequences

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

`\varseq`

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

This is best shown by example:

Example 20

Input:

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

Output:

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

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$a_1 \dots a_n$

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

Example 22

Input:

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

Output:

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

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

Example 23

Input:

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

Output:

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

3.4 Module Inheritance and Structures

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

Before we get into theory morphisms in $\text{\texttt{gT\textsubscript{E}X}}$ we will see a very simple application of modules: managing multilinguality modularly.

3.4.1 Multilinguality and Translations

If we load the $\text{\texttt{gT\textsubscript{E}X}}$ document class or package with the option `lang=<lang>`, $\text{\texttt{gT\textsubscript{E}X}}$ will load the appropriate `babel` language for you – e.g. `lang=de` will load the `babel` language

ngerman. Additionally, it makes \TeX aware of the current document being set in (in this example) *german*. This matters for reasons other than mere `babel`-purposes, though:

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

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

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

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

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

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

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

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

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

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

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



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

Unfortunately, \TeX only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that \TeX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

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

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

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



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

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

- Similarly, in `\importmodule{some/path?Foo}` the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive's top-level namespace and `source`-folder, respectively.

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

- Similarly, `\importmodule[Some/Archive]{some/path?Foo}` is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.
- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared



in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document. Since this is less compatible with a modular development, using full URIs directly is strongly discouraged, unless the module is declared in the current file directly.

`\STEXexport`

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



For persistency reasons, everything in an `\STEXexport` is digested by T_EX in the L^AT_EX3-category code scheme. This means that the characters `_` and `:` are considered *letters* and valid parts of control sequence names, and space characters are ignored entirely. For spaces, use the character `~` instead, and keep in mind, that if you want to use subscripts, you should use `\c_math_subscript_token` instead of `_`! Also note, that `\newcommand` defines macros *globally* and throws an error if the macro already exists, potentially leading to low-level L^AT_EX errors if we put a `\newcommand` in an `\STEXexport` and the `<code>` is executed more than once in a document – which can happen easily. A safer alternative is to use macro definition principles, that are safe to use even if the macro being defined already exists, and ideally are local to the current T_EX group, such as `\def` or `\let`.

3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

```

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

Output:

A 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:

is a .

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:

```

, and  $ab$ .
Also:

```

`\instantiate`

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

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

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

- `mathstructure{<name>}` simply creates a nested theory with name `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)`
- $\text{M} \rightarrow$ – a *dependent record type with manifest fields*, the fields of which are generated from (and correspond to) the constants in `<name>-structure`.
- $\text{T} \rightarrow$ `\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 is a structure $M := U \circ e$ such that $\circ : UU \dots$

and

Example 28

Input:

```

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

```

Output:

Let $M_2 := \circ e$ be a on ...

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: $\rtimes a{\rm rplus} c{\rm rtimes} d{\rm }$
19 \end{smodule}

```

Output:

Test: $a(cde)$

TODO: explain donotclone

3.4.5 The interpretmodule Environment

TODO: explain

Example 31

Input:

```

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

```

Output:

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

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

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

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

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

Chapter 4

Using \TeX Symbols

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

4.1 `\symref` and its variants

`\symref`
`\symname`

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

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

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

Example 32

Input:

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

Output:

A 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:

```
are...
```



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

If `\symbolname` is a semantic macro, then $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ 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...`. $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ attempts to handle this case thusly:

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

```
n m is...
```

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

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

is...

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

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

Example 36

Input:

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

Output:

k yields...

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

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 nm , 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 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`),

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

- (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

STEX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

The *presentation* of these environments can be customized to use e.g. predefined theorem-environments, see ?? 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 $$$, $\multiplication{2,3}$ is $$.
8 \end{sexample}
```

Output:

Example 5.1.1 (An Example). 23 is 5, 23 is 6.

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

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

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

`\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 where : and such that

Axiom 5.1.2 (Associativity). *is associative*

Axiom 5.1.3 (Unit). *xx for all x*

An example for a is...

The main difference to before⁴ is that the two **sassertions** now have **name=** attributes. Thus the **mathstructure** 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 .

Proof: By contradiction

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

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 $2ca$

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 .

□

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

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

Proof: By contradiction

1. Assume $\sqrt{2}$ is

2. Then $(\frac{a^2}{b^2})2$ for some ab with a, b

3. Then a is even

4. Then b is also even

\Rightarrow Contradiction to a, b being .

□

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{spproof}[id=simple-proof,method=induction]
2   {We prove that  $\sum_{i=1}^n 2i-1 = n^2$  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 . Thus we have
 $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by . We can 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. □

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

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

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

\spfststep Regular proof steps are marked up with the **\spfststep** 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

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

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

\justarg

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

`\sproofend`

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

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

`\sProofEndSymbol`

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

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

5.3 Highlighting and Presentation Customizations

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

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

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

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

All of these commands take one optional and two proper arguments, i.e.

`\stexpatch* [<type>] {<begin-code>} {<end-code>}`.

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

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

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

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


```

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

```

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

```

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

```

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

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

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

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

For each of the four macros, there exists an additional macro that takes the full URI of the relevant symbol currently being highlighted as a second argument. That allows us to e.g. use pdf tooltips and links. For example, this document uses⁵

```

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

```

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

Chapter 6

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

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

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

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

\tikzinput
\ctikzinput

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

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

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

`\mhtikzinput`
`\cmhtikzinput`

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

`\libusetikzlibrary`

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

7.2 Modular Document Structuring

7.2.1 Introduction

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

7.2.3 Document Fragments

sfragment

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

```

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

```

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

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

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

```

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

```

Here we use two levels of `blindfragment`:

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

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

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

7.2.4 Ending Documents Prematurely

`\prematurestop`
`\afterprematurestop`

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

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

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the `TeX` preamble of the course notes file.

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.

`\setSGvar`
`\useSGvar`

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

`\ifSGvar`

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

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:

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

7.3.3 Notes and Slides

- `frame` Slides are represented with the `frame` environment just like in the `beamer` class, see [\[Tanb\]](#) for details.
- `note` The `notesslides` class adds the `note` environment for encapsulating the course note fragments.



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

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

```

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

```



```

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

```

`\ifnotes`

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

`nexample`, `nsproof`, `nassertion`

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

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

Test Slide

▷ Some text

©: MiKo
1
sTeX

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

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

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

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

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 \TeX notes.

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

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

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


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

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

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

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

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

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

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`
`\printexcursion`
`\excursionref`

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

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

`\excursiongroup`

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

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



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

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

<hr/> solutions	The problem package takes the options solutions (should solutions be output?), notes
notes	(should the problem notes be presented?), hints (do we give the hints?), gnotes (do we
hints	show grading notes?), pts (do we display the points awarded for solving the problem?),
gnotes	min (do we display the estimated minutes for problem soling). If theses are specified, then
pts	the corresponding auxiliary parts of the problems are output, otherwise, they remain
min	invisible.
boxed	The boxed option specifies that problems should be formatted in framed boxes so
test	that they are more visible in the text. Finally, the test option signifies that we are in
<hr/>	a test situation, so this option does not show the solutions (of course), but leaves space
	for the students to solve them.
 problem	 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 The `solution` environment can be used to specify a solution to a problem. If the package option `solutions` is set or `\solutionstrue` is set in the text, then the solution will be presented in the output. The `solution` environment takes an optional KeyVal argument with the keys `id` for an identifier that can be reference `for` to specify which problem this is a solution for, and `height` that allows to specify the amount of space to be left in test situations (i.e. if the `test` option is set in the `\usepackage` statement).

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

\startsolutions
\stopsolutions

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

\ifsolutions

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

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` 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=Nooooooooooo,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=Noooooooooooo,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 Elefants can you fit into a Volkswagen beetle? \fillinsol{4}
4 \end{problem}
```

Output:

Problem 7.4.4 (Fitting Elephants)

How many Elefants can you fit into a Volkswagen beetle?

and the actual solution in solutions mode:

Example 44

Input:

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

Output:

Problem 7.4.5 (Fitting Elephants)

How many Elefants can you fit into a Volkswagen beetle?

4!

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

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

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

screen after each run. This is useful in preparing exams, where we want to make sure that the students can indeed solve the problems in an allotted time period.

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

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

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

7.5.3 Assignments

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

7.5.4 Including Assignments

`\inputassignment`

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

7.5.5 Typesetting Exams

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

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

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

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

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

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-08-07

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

EdN:8

8

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

Part II
Documentation

Chapter 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 sTeX logo.
<code>\stex</code>	

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

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

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 $\langle content \rangle$ with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

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

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

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

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

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

<code>\stex_path_if_absolute_p:N</code> \star <code>\stex_path_if_absolute:N$\underline{T$</code> \star	Checks whether the path provided is <i>absolute</i> , i.e. starts with an empty segment
---	---

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

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

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

9.1.2 MathHub Archives

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

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

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

<code>\l_stex_current_repository_prop</code>	
--	--

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

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

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

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

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

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

9.1.3 Using Content in Archives

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

Chapter 10

STEX-References

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

10.1 Macros and Environments

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

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

\stex_get_document_uri:

Computes the current document uri from the current archive's **narr**-field and its location relative to the archive's **source**-directory. Reference targets are computed from this URI and the reference-id.

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

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

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

10.1.1 Setting Reference Targets

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

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

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

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

10.1.2 Using References

<code>\sref</code>	<code>\sref[<i><opt-args></i>]{<i><id></i>}</code>
--------------------	--

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

<code>\srefsym</code>	<code>\srefsym[<i><opt-args></i>]{<i><symbol></i>}</code>
-----------------------	---

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>`.

<code>\srefsymuri</code>	<code>\srefsymuri{<i><URI></i>}{<i><text></i>}</code>
--------------------------	---

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:

 $\backslash c_{stex_module_}\langle URI \rangle_prop$

A property list with the following fields:

name The *name* of the module,

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

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

lang the module's *language*,

sig the language of the signature module, if the current file is a translation from some other language,

deprecate if this module is deprecated, the module that replaces it,

meta the metatheory of the module.

 $\backslash c_{stex_module_}\langle URI \rangle_code$

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

 $\backslash c_{stex_module_}\langle URI \rangle_constants$

The names of all constants declared in the module

 $\backslash c_{stex_module_}\langle URI \rangle_constants$

The full URIs of all modules imported in this module

<hr/> <hr/> <code>\l_stex_current_module_str</code> <hr/> <hr/>	<code>\l_stex_current_module_str</code> always contains the URI of the current module (if existent).
<hr/> <hr/> <code>\l_stex_all_modules_seq</code> <hr/> <hr/>	Stores full URIs for all modules currently in scope.
<hr/> <hr/> <code>\stex_if_in_module_p: *</code> <code>\stex_if_in_module:TF *</code> <hr/> <hr/>	Conditional for whether we are currently in a module
<hr/> <hr/> <code>\stex_if_module_exists_p:n *</code> <code>\stex_if_module_exists:nTF *</code> <hr/> <hr/>	Conditional for whether a module with the provided URI is already known.
<hr/> <hr/> <code>\stex_add_to_current_module:n</code> <code>\STEXexport</code> <hr/> <hr/>	Adds the provided tokens to the <code>_code</code> control sequence of the current module. <code>\stex_add_to_current_module:n</code> is used internally, <code>\STEXexport</code> is intended for users and additionally executes the provided code immediately.
<hr/> <hr/> <code>\stex_add_constant_to_current_module:n</code> <hr/> <hr/>	Adds the declaration with the provided name to the <code>_constants</code> control sequence of the current module.
<hr/> <hr/> <code>\stex_add_import_to_current_module:n</code> <hr/> <hr/>	Adds the module with the provided full URI to the <code>_imports</code> control sequence of the current module.
<hr/> <hr/> <code>\stex_collect_imports:n</code> <hr/> <hr/>	Iterates over all imports of the provided (full URI of a) module and stores them as a topologically sorted list – including the provided module as the last element – in <code>\l_stex_collect_imports_seq</code>
<hr/> <hr/> <code>\stex_do_up_to_module:n</code> <hr/> <hr/>	Code that is <i>exported</i> from module (such as symbol declarations) should be local <i>to the current module</i> . For that reason, ideally all symbol declarations and similar commands should be called directly in the module environment, however, that is not always feasible, e.g. in structural features or <code>sparapraphs</code> . <code>\stex_do_up_to_module</code> therefore executes the provided code repeatedly in an <code>\aftergroup</code> up until the group level is equal to that of the innermost smodule environment.

\stex_modules_current_namespace:

Computes the current namespace as follows:

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

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

11.1.1 The smodule environment

module `\begin{module}[\langle options \rangle]{\langle name \rangle}`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\stex_invoke_module:n `\stex_invoke_module:n`

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

`\stex_activate_module:n`

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

Chapter 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: *`
`\stex_if_smsmode: TF *`

Tests whether SMS mode is currently active.

<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

`\stex_import_module_uri:nn`

`\stex_import_module_uri:nn {<archive-ID>} {<module-path>}`

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

If `<archive-ID>` is empty, it is automatically set to the ID of the current archive (if one exists).

1. If `<archive-ID>` is empty:

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

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

- (b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

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

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

- (b) If `<path>` is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive.

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

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

stores the result in these four variables.

`\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}`

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

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

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

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

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

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

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

Chapter 15

TeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` TODO

Chapter 16

sTeX-Statements

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

16.1 Macros and Environments

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

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

```

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

```

24.2 Preliminaries

```

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

```

\stex The sTeX logo:

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

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

`\stex_annotate_invisible:n`

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:NTF \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:N`. 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 \AddToHook{begindocument}{
917 \ltx@ifpackageloaded{graphicx}{
918   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
919   \providecommand\mhgraphics[2][]{%
920     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
921     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
922   \providecommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
923 }{
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         {\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}~\@cu
1102     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1103     \immediate\write\@auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_st
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_
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_
1449 %           }
1450 %         }
1451 %       }{
1452 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
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 {
1460   \__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
1470 \msg_new:nnn{stex}{error/unknownmodule}{
1471     No~module~#1~found
1472 }
1473 \msg_new:nnn{stex}{error/syntax}{
1474     Syntax~error:~#1
1475 }
1476 \msg_new:nnn{stex}{error/siglanguage}{
1477     Module~#1~declares~signature~#2,~but~does~not~
1478     declare~its~language
1479 }
1480 \msg_new:nnn{stex}{warning/deprecated}{
1481     #1~is~deprecated;~please~use~#2~instead!
1482 }
1483 \msg_new:nnn{stex}{error/conflictingmodules}{
1484     Conflicting~imports~for~module~#1
1485 }
1486
1487 \l_stex_current_module_str The current module:
1488 \str_new:N \l_stex_current_module_str
1489
1490 (End definition for \l_stex_current_module_str. This variable is documented on page 79.)
1491
1492 \l_stex_all_modules_seq Stores all available modules
1493 \seq_new:N \l_stex_all_modules_seq
1494
1495 (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`
`\STEXexport`

Only allowed within modules:

```

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 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:~#1\\~2:~#2\\~3:~#3\\~4:~#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:nnxx{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:NnTF \l_tmpa_str a {
2958       \__stex_notation_argument_assoc:nn{a}
2959     }{
2960       \str_if_eq:NnTF \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     {\_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 \_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 \_stex_notation_remaining_precs_seq \l_tmpa_str
3008   \tl_put_right:Nx \_stex_notation_dummyargs_tl { {
3009     \STEXInternalTermMathAssocArgiiii
3010     { \int_use:N \_stex_notation_currarg_int }
3011     { \l_tmpa_str }
3012     { ####\int_use:N \_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 \stex_annotate_invisible:nnn { precedence }
3063 { \l__stex_notation_prec_str }{}
3064
3065 \int_zero:N \l_tmpa_int
3066 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3067 \tl_clear:N \l_tmpa_tl
3068 \int_step_inline:nn { \l__stex_notation_arity_str }{
3069 \int_incr:N \l_tmpa_int
3070 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3071 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
3072 \str_if_eq:VnTF \l_tmpb_str a {
3073 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3074 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{},
3075 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{},
3076 } }
3077 }{
3078 \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}{
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   \prop_put:Nnn \l_tmpa_prop {argnames} {}
3530
3531   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3532   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3533
3534   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3535   \int_step_inline:nn \l__stex_variables_args_int {
3536     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3537   }
3538   \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3539   \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3540   \tl_if_empty:NF \l__stex_variables_mid_tl {
3541     \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3542     \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3543   }
3544   \exp_args:NNo \tl_set:No \l_tmpb_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_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3547   }
3548   \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3549   \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3550
3551

```

```

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

```

```

3606
3607   }}
3608
3609   \prop_set_eq:cN {l_stex_symdecl_varseq://\l__stex_variables_name_str _prop}\l_tmpa_prop
3610   \ignorespacesandpars
3611 }
3612
3613 </package>

```

Chapter 30

STEX -Terms Implementation

```
3614 <*package>
3615
3616 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3617
3618 <@@=stex_terms>
3619
3620 Warnings and error messages
3621 \msg_new:nnn{stex}{error/nonotation}{
3622   Symbol~#1~invoked,~but~has~no~notation#2!
3623 }
3624 \msg_new:nnn{stex}{error/notationarg}{
3625   Error~in~parsing~notation~#1
3626 }
3627 \msg_new:nnn{stex}{error/noop}{
3628   Symbol~#1~has~no~operator~notation~for~notation~#2
3629 }
3630 \msg_new:nnn{stex}{error/notallowed}{
3631   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3632 }
3633 \msg_new:nnn{stex}{error/doubleargument}{
3634   Argument~#1~of~symbol~#2~already~assigned
3635 }
3636 \msg_new:nnn{stex}{error/overarity}{
3637   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3638 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3638
3639
3640 \bool_new:N \l_stex_allow_semantic_bool
3641 \bool_set_true:N \l_stex_allow_semantic_bool
3642
```

```

3643 \cs_new_protected:Nn \stex_invoke_symbol:n {
3644   \ifvmode\indent\fi
3645   \bool_if:NTF \l_stex_allow_semantic_bool {
3646     \str_if_eq:eeF {
3647       \prop_item:cn {
3648         l_stex_symdecl_#1_prop
3649       }{ deprecate }
3650     }{}{
3651       \msg_warning:nxxx{stex}{warning/deprecated}{
3652         Symbol~#1
3653       }{
3654         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3655       }
3656     }
3657     \if_mode_math:
3658       \exp_after:wN \__stex_terms_invoke_math:n
3659     \else:
3660       \exp_after:wN \__stex_terms_invoke_text:n
3661     \fi: { #1 }
3662   }{
3663     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3664   }
3665 }
3666
3667 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3668   \peek_charcode_remove:NTF ! {
3669     \__stex_terms_invoke_op_custom:nn {#1}
3670   }{
3671     \__stex_terms_invoke_custom:nn {#1}
3672   }
3673 }
3674
3675 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3676   \peek_charcode_remove:NTF ! {
3677     % operator
3678     \peek_charcode_remove:NTF * {
3679       % custom op
3680       \__stex_terms_invoke_op_custom:nn {#1}
3681     }{
3682       % op notation
3683       \peek_charcode:NTF [ {
3684         \__stex_terms_invoke_op_notation:nw {#1}
3685       }{
3686         \__stex_terms_invoke_op_notation:nw {#1}[]
3687       }
3688     }
3689   }{
3690     \peek_charcode_remove:NTF * {
3691       \__stex_terms_invoke_custom:nn {#1}
3692       % custom
3693     }{
3694       % normal
3695       \peek_charcode:NTF [ {
3696         \__stex_terms_invoke_notation:nw {#1}

```

```

3697     }{
3698       \__stex_terms_invoke_notation:nw {#1}[]
3699     }
3700   }
3701 }
3702 }
3703
3704
3705 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3706   \exp_args:Nnx \use:nn {
3707     \def\comp{\_comp}
3708     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3709     \bool_set_false:N \l_stex_allow_semantic_bool
3710     \stex_mathml_intent:nn{#1}{
3711       \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3712         \comp{ #2 }
3713       }
3714     }
3715   }{
3716     \stex_reset:N \comp
3717     \stex_reset:N \STEXInternalCurrentSymbolStr
3718     \bool_set_true:N \l_stex_allow_semantic_bool
3719   }
3720 }
3721
3722 \keys_define:nn { stex / terms } {
3723   % lang      .tl_set_x:N = \l_stex_notation_lang_str ,
3724   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3725   unknown .code:n      = \str_set:Nx
3726     \l_stex_notation_variant_str \l_keys_key_str
3727 }
3728
3729 \cs_new_protected:Nn \__stex_terms_args:n {
3730   % \str_clear:N \l_stex_notation_lang_str
3731   \str_clear:N \l_stex_notation_variant_str
3732
3733   \keys_set:nn { stex / terms } { #1 }
3734 }
3735
3736 \cs_new_protected:Nn \stex_find_notation:nn {
3737   \__stex_terms_args:n { #2 }
3738   \seq_if_empty:cTF {
3739     l_stex_symdecl_ #1 _notations
3740   } {
3741     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3742   } {
3743     \str_if_empty:NTF \l_stex_notation_variant_str {
3744       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3745     }{
3746       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3747         \l_stex_notation_variant_str
3748       }{
3749         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3750       }{

```

```

3751         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3752         ~\l_stex_notation_variant_str
3753         }
3754     }
3755 }
3756 }
3757 }
3758
3759 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3760     \exp_args:Nnx \use:nn {
3761         \def\comp{\_comp}
3762         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3763         \stex_find_notation:nn { #1 }{ #2 }
3764         \bool_set_false:N \l_stex_allow_semantic_bool
3765         \cs_if_exist:cTF {
3766             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3767         }{
3768             \_stex_term_oms:nnn { #1 }{
3769                 #1 \c_hash_str \l_stex_notation_variant_str
3770             }{
3771                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3772             }
3773         }{
3774             \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3775                 \cs_if_exist:cTF {
3776                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3777                 }{
3778                     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3779                         \_stex_reset:N \comp
3780                         \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3781                         \_stex_reset:N \STEXInternalCurrentSymbolStr
3782                         \bool_set_true:N \l_stex_allow_semantic_bool
3783                     }
3784                     \def\comp{\_comp}
3785                     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3786                     \bool_set_false:N \l_stex_allow_semantic_bool
3787                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3788                 }{
3789                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3790                     ~\l_stex_notation_variant_str
3791                     }
3792                 }
3793             }{
3794                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3795             }
3796         }
3797     }{
3798         \_stex_reset:N \comp
3799         \_stex_reset:N \STEXInternalCurrentSymbolStr
3800         \bool_set_true:N \l_stex_allow_semantic_bool
3801     }
3802 }
3803
3804 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```



```

3805 \stex_find_notation:nn { #1 }{ #2 }
3806 \cs_if_exist:CTF {
3807   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3808 }{
3809   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3810     \_stex_reset:N \comp
3811     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3812     \_stex_reset:N \STEXInternalCurrentSymbolStr
3813     \bool_set_true:N \l_stex_allow_semantic_bool
3814   }
3815   \def\comp{\_comp}
3816   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3817   \bool_set_false:N \l_stex_allow_semantic_bool
3818   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3819 }{
3820   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3821     ~\l_stex_notation_variant_str
3822   }
3823 }
3824 }
3825
3826 \prop_new:N \l__stex_terms_custom_args_prop
3827 \clist_new:N \l_stex_argnames_seq
3828 \seq_new:N \l__stex_terms_tmp_seq
3829
3830 \cs_new_protected:Nn \__stex_terms_custom_comp:n{ \bool_set_false:N \l_stex_allow_semantic_bo
3831
3832 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3833   \exp_args:Nnx \use:nn {
3834     \def\comp{\__stex_terms_custom_comp:n}
3835     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3836     \prop_clear:N \l__stex_terms_custom_args_prop
3837     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3838     \prop_get:cnN {
3839       l_stex_symdecl_#1 _prop
3840     }{ args } \l_tmpa_str
3841     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
3842       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
3843     }
3844     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3845     \tl_set:Nn \arg { \__stex_terms_arg: }
3846     \str_if_empty:NTF \l_tmpa_str {
3847       \stex_mathml_intent:nn{#1}{
3848         \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3849       }
3850     }{
3851       \seq_clear:N \l__stex_terms_tmp_seq
3852       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
3853         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
3854         \bool_lazy_or:nnT{
3855           \str_if_eq_p:nn{a}{\str_item:Nn \l_tmpa_str{##1}}
3856         }{
3857           \str_if_eq_p:nn{B}{\str_item:Nn \l_tmpa_str{##1}}
3858         }{

```

```

3859         \tl_put_right:Nn \l__stex_terms_tmp_tl +
3860     }
3861     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
3862 }
3863 \stex_mathml_intent:nn{
3864     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}]{ args }(
3865     \seq_use:Nn \l__stex_terms_tmp_seq ,
3866 )
3867 ){
3868     \str_if_in:NnTF \l_tmpa_str b {
3869         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3870     }{
3871         \str_if_in:NnTF \l_tmpa_str B {
3872             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3873         }{
3874             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3875         }
3876     }
3877 }
3878 }
3879 % TODO check that all arguments exist
3880 ){
3881     \_stex_reset:N \l_stex_argnames_seq
3882     \_stex_reset:N \STEXInternalCurrentSymbolStr
3883     \_stex_reset:N \arg
3884     \_stex_reset:N \comp
3885     \_stex_reset:N \l__stex_terms_custom_args_prop
3886     %\bool_set_true:N \l_stex_allow_semantic_bool
3887 }
3888 }
3889
3890 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3891     \tl_if_empty:nTF {#2}{
3892         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3893         \bool_set_true:N \l_tmpa_bool
3894         \bool_do_while:Nn \l_tmpa_bool {
3895             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3896             \int_incr:N \l_tmpa_int
3897         }{
3898             \bool_set_false:N \l_tmpa_bool
3899         }
3900     }
3901     ){
3902         \int_set:Nn \l_tmpa_int { #2 }
3903     }
3904     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3905     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3906         \msg_error:nnxxx{stex}{error/overarity}
3907         {\int_use:N \l_tmpa_int}
3908         {\STEXInternalCurrentSymbolStr}
3909         {\str_count:N \l_tmpa_str}
3910     }
3911     \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3912     \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {

```

```

3913 \bool_lazy_any:nF {
3914   {\str_if_eq_p:Vn \l_tmpa_str {a}}
3915   {\str_if_eq_p:Vn \l_tmpa_str {B}}
3916 }{
3917   \msg_error:nnxx{stex}{error/doubleargument}
3918   {\int_use:N \l_tmpa_int}
3919   {\STEXInternalCurrentSymbolStr}
3920 }
3921 }
3922 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3923 \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
3924   \bool_set_true:N \l_stex_allow_semantic_bool
3925   \use:nn
3926 }
3927 {
3928 \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
3929   \IfBooleanTF#1{
3930     \stex_annotate_invisible:n { %TODO
3931       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3932     }
3933   }{ %TODO
3934     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3935   }
3936 }}
3937 {\bool_set_false:N \l_stex_allow_semantic_bool}
3938 }
3939
3940
3941 \cs_new_protected:Nn \_stex_term_arg:nn {
3942   \bool_set_true:N \l_stex_allow_semantic_bool
3943   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3944   \bool_set_false:N \l_stex_allow_semantic_bool
3945 }
3946
3947 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
3948   \exp_args:Nnx \use:nn
3949   { \int_set:Nn \l__stex_terms_downprec { #2 }
3950     \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
3951       \_stex_term_arg:nn { #1 }{ #3 }
3952     }
3953   }
3954   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3955 }

```

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

`\STEXInternalTermMathAssocArgiiii`

```

3956 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4#5 {
3957   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3958   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
3959   \tl_if_empty:nTF { #3 }{
3960     \STEXInternalTermMathArgiii{#5#1}{#2}{}
3961   }{
3962     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{

```

```

3963 \expandafter\if\expandafter\relax\noexpand#3
3964 \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
3965 \else
3966 \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
3967 \fi
3968 \l_tmpa_tl
3969 }{
3970 \_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
3971 }
3972 }
3973 }
3974
3975 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_maybe_sequence:Nnn {
3976 \str_set:Nx \l_tmpa_str {\cs_argument_spec:N #1 }
3977 \str_if_empty:NTF \l_tmpa_str {
3978 \exp_args:Nx \cs_if_eq:NNTF {
3979 \tl_head:N #1
3980 } \stex_invoke_sequence:n {
3981 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3982 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3983 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
3984 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3985 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3986 \exp_not:n{\exp_args:Nnx \use:nn} {
3987 \exp_not:n {
3988 \def\comp{\_varcomp}
3989 \str_set:Nn \STEXInternalCurrentSymbolStr
3990 } {varseq://\l_tmpa_str}
3991 \exp_not:n{ ##1 }
3992 }{
3993 \exp_not:n {
3994 \_stex_reset:N \comp
3995 \_stex_reset:N \STEXInternalCurrentSymbolStr
3996 }
3997 }
3998 }}}
3999 \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4000 \seq_reverse:N \l_tmpa_seq
4001 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4002 \seq_map_inline:Nn \l_tmpa_seq {
4003 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4004 \exp_args:Nno
4005 \l_tmpa_cs { ##1 } \l_tmpa_tl
4006 }
4007 }
4008 \tl_set:Nx \l_tmpa_tl {
4009 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4010 \exp_args:No \exp_not:n \l_tmpa_tl
4011 }
4012 }
4013 \exp_args:No\l_tmpb_tl\l_tmpa_tl
4014 }{
4015 \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4016 }

```

```

4017 } {
4018   \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4019 }
4020
4021 }
4022
4023 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nnn {
4024   \clist_set:Nn \l_tmpa_clist{ #2 }
4025   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4026     \tl_set:Nn \l_tmpa_tl {
4027       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4028         \_stex_term_arg:nn{A#3#1}{ #2 } }
4029     }
4030   }{
4031     \clist_reverse:N \l_tmpa_clist
4032     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4033     \tl_set:Nx \l_tmpa_tl {
4034       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4035         \_stex_term_arg:nn{A#3#1}{
4036           \exp_args:No \exp_not:n \l_tmpa_tl
4037         }
4038       }
4039     }
4040     \clist_map_inline:Nn \l_tmpa_clist {
4041       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
4042         \exp_args:Nno
4043         \l_tmpa_cs {
4044           \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4045             \_stex_term_arg:nn{A#3#1}{##1}
4046           }
4047         } \l_tmpa_tl
4048       }
4049     }
4050     \exp_args:No\l_tmpb_tl\l_tmpa_tl
4051   }

```

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

30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
4052 \tl_const:Nx \infprec {\int_use:N \c_max_int}
4053 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4054 \int_new:N \l__stex_terms_downprec
4055 \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
4056 \tl_set:Nn \l__stex_terms_left_bracket_str (
4057 \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

```

4058 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4059   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4060     \bool_set_false:N \l__stex_terms_brackets_done_bool
4061     #2
4062   } {
4063     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4064       \bool_if:NTF \l__stex_inarray_bool { #2 }{
4065         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4066         \dobrackets { #2 }
4067       }
4068     }{ #2 }
4069   }
4070 }

```

(End definition for _stex_terms_maybe_brackets:nn.)

\dobrackets

```

4071 \bool_new:N \l__stex_terms_brackets_done_bool
4072 %\RequirePackage{scalerel}
4073 \cs_new_protected:Npn \dobrackets #1 {
4074   %\ThisStyle{\if D\m@switch
4075   %   \exp_args:Nnx \use:nn
4076   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4077   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
4078   %   \else
4079   %   \exp_args:Nnx \use:nn
4080   %   {
4081   %     \bool_set_true:N \l__stex_terms_brackets_done_bool
4082   %     \int_set:Nn \l__stex_terms_downprec \infprec
4083   %     \l__stex_terms_left_bracket_str
4084   %     #1
4085   %   }
4086   %   {
4087   %     \bool_set_false:N \l__stex_terms_brackets_done_bool
4088   %     \l__stex_terms_right_bracket_str
4089   %     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4090   %   }
4091   %\fi}
4092 }

```

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

\withbrackets

```

4093 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4094   \exp_args:Nnx \use:nn
4095   {
4096     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4097     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4098     #3
4099   }
4100 }

```

```

4101 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4102 {\l__stex_terms_left_bracket_str}
4103 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4104 {\l__stex_terms_right_bracket_str}
4105 }
4106 }

```

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

\STEXinvisible

```

4107 \cs_new_protected:Npn \STEXinvisible #1 {
4108 \stex_annotate_invisible:n { #1 }
4109 }

```

(End definition for \STEXinvisible. This function is documented on page 88.)
OMDoc terms:

\STEXInternalTermMathOMSiiii

```

4110 \cs_new_protected:Nn \_stex_term_oms:nnn {
4111 \stex_annotate:nnn{ OMID }{ #2 }{
4112 #3
4113 }
4114 }
4115
4116 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4117 \__stex_terms_maybe_brackets:nn { #3 }{
4118 \stex_mathml_intent:nn{#1} {
4119 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4120 }
4121 }
4122 }

```

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

_stex_term_math_omv:nn

```

4123 \cs_new_protected:Nn \_stex_term_omv:nn {
4124 \stex_annotate:nnn{ OMV }{ #1 }{
4125 #2
4126 }
4127 }

```

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

\STEXInternalTermMathOMAiiai

```

4128 \cs_new_protected:Nn \_stex_term_oma:nnn {
4129 \stex_annotate:nnn{ OMA }{ #2 }{
4130 #3
4131 }
4132 }
4133
4134 \cs_new_protected:Npn \STEXInternalTermMathOMAiiai #1#2#3#4 {
4135 \exp_args:Nnx \use:nn {
4136 \seq_clear:N \l__stex_terms_tmp_seq
4137 \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4138 \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {

```

```

4139     \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4140   }
4141   \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4142     \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4143     \bool_lazy_or:nnT{
4144       \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4145     }{
4146       \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4147     }{
4148       \tl_put_right:Nn \l__stex_terms_tmp_tl +
4149     }
4150     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4151   }
4152 }
4153 \__stex_terms_maybe_brackets:nn { #3 }{
4154   \stex_mathml_intent:nn{
4155     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4156       \seq_use:Nn \l__stex_terms_tmp_seq ,
4157     )
4158   }{
4159     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4160   }
4161 }
4162 }{
4163   \_stex_reset:N \l_stex_argnames_seq
4164 }
4165 }

```

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

`\STEXInternalTermMathOMBiiii`

```

4166 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4167   \stex_annotate:nnn{ OMBIND }{ #2 }{
4168     #3
4169   }
4170 }
4171
4172 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
4173   \exp_args:Nnx \use:nn {
4174     \seq_clear:N \l__stex_terms_tmp_seq
4175     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4176       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4177         \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4178       }
4179       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4180         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4181         \bool_lazy_or:nnT{
4182           \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4183         }{
4184           \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4185         }{
4186           \tl_put_right:Nn \l__stex_terms_tmp_tl +
4187         }
4188         \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl

```



```

4189     }
4190   }
4191   \__stex_terms_maybe_brackets:nn { #3 }{
4192     \stex_mathml_intent:nn{
4193       #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4194         \seq_use:Nn \l__stex_terms_tmp_seq ,
4195       )
4196     }{
4197       \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4198     }
4199   }
4200   }{
4201     \stex_reset:N \l_stex_argnames_seq
4202   }
4203 }

```

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

`\symref`
`\symname`

```

4204 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4205
4206 \keys_define:nn { stex / symname } {
4207   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
4208   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
4209   root     .tl_set_x:N      = \l__stex_terms_root_tl
4210 }
4211
4212 \cs_new_protected:Nn \stex_symname_args:n {
4213   \tl_clear:N \l__stex_terms_post_tl
4214   \tl_clear:N \l__stex_terms_pre_tl
4215   \tl_clear:N \l__stex_terms_root_str
4216   \keys_set:nn { stex / symname } { #1 }
4217 }
4218
4219 \NewDocumentCommand \symref { m m }{
4220   \let\compemph_uri_prev:\compemph@uri
4221   \let\compemph@uri\symrefemph@uri
4222   \STEXsymbol{#1}!\{ #2 }
4223   \let\compemph@uri\compemph_uri_prev:
4224 }
4225
4226 \NewDocumentCommand \synonym { 0{} m m }{
4227   \stex_symname_args:n { #1 }
4228   \let\compemph_uri_prev:\compemph@uri
4229   \let\compemph@uri\symrefemph@uri
4230   % TODO
4231   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4232   \let\compemph@uri\compemph_uri_prev:
4233 }
4234
4235 \NewDocumentCommand \symname { 0{} m }{
4236   \stex_symname_args:n { #1 }
4237   \stex_get_symbol:n { #2 }
4238   \str_set:Nx \l_tmpa_str {

```

```

4239   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4240 }
4241 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4242
4243 \let\compemph_uri_prev:\compemph@uri
4244 \let\compemph@uri\symrefemph@uri
4245 \exp_args:Nnx \use:nn
4246 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4247   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4248 } }
4249 \let\compemph@uri\compemph_uri_prev:
4250 }
4251
4252 \NewDocumentCommand \Symname { 0{} m }{
4253   \stex_symname_args:n { #1 }
4254   \stex_get_symbol:n { #2 }
4255   \str_set:Nx \l_tmpa_str {
4256     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4257   }
4258   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4259   \let\compemph_uri_prev:\compemph@uri
4260   \let\compemph@uri\symrefemph@uri
4261   \exp_args:Nnx \use:nn
4262   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4263     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4264     \l__stex_terms_post_tl
4265   } }
4266   \let\compemph@uri\compemph_uri_prev:
4267 }

```

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

30.3 Notation Components

```

4268 <@@=stex_notationcomps>

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

4269 \cs_new_protected:Npn \_comp #1 {
4270   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4271     \stex_html_backend:TF {
4272       \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4273     }{
4274       \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4275     }
4276   }
4277 }
4278
4279 \cs_new_protected:Npn \_varcomp #1 {
4280   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4281     \stex_html_backend:TF {
4282       \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4283     }{
4284       \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
4285     }

```

```

4286 }
4287 }
4288
4289 \def\comp{\_comp}
4290
4291 \cs_new_protected:Npn \compemph@uri #1 #2 {
4292   \compemph{ #1 }
4293 }
4294
4295
4296 \cs_new_protected:Npn \compemph #1 {
4297   #1
4298 }
4299
4300 \cs_new_protected:Npn \defemph@uri #1 #2 {
4301   \defemph{#1}
4302 }
4303
4304 \cs_new_protected:Npn \defemph #1 {
4305   \textbf{#1}
4306 }
4307
4308 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4309   \symrefemph{#1}
4310 }
4311
4312 \cs_new_protected:Npn \symrefemph #1 {
4313   \emph{#1}
4314 }
4315
4316 \cs_new_protected:Npn \varemp@uri #1 #2 {
4317   \varemp{#1}
4318 }
4319
4320 \cs_new_protected:Npn \varemp #1 {
4321   #1
4322 }

```

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

\ellipses

```

4323 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 4324 \bool_new:N \l_stex_inarray_bool
\parrayline 4325 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 4326 \NewDocumentCommand \parray { m m } {
\parraycell 4327   \begingroup
4328   \bool_set_true:N \l_stex_inarray_bool
4329   \begin{array}{#1}
4330     #2
4331   \end{array}
4332   \endgroup

```

```

4333 }
4334
4335 \NewDocumentCommand \prmatrix { m } {
4336   \begingroup
4337   \bool_set_true:N \l_stex_inarray_bool
4338   \begin{matrix}
4339     #1
4340   \end{matrix}
4341   \endgroup
4342 }
4343
4344 \def \maybepline {
4345   \bool_if:NT \l_stex_inarray_bool {\hline}
4346 }
4347
4348 \def \parrayline #1 #2 {
4349   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
4350 }
4351
4352 \def \pmrow #1 { \parrayline{}{ #1 } }
4353
4354 \def \parraylineh #1 #2 {
4355   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
4356 }
4357
4358 \def \parraycell #1 {
4359   #1 \bool_if:NT \l_stex_inarray_bool {&}
4360 }

```

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

30.4 Variables

```

4361 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

4362 \cs_new_protected:Nn \stex_invoke_variable:n {
4363   \if_mode_math:
4364     \exp_after:wN \__stex_variables_invoke_math:n
4365   \else:
4366     \exp_after:wN \__stex_variables_invoke_text:n
4367   \fi: {#1}
4368 }
4369
4370 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4371   \peek_charcode_remove:NTF ! {
4372     \__stex_variables_invoke_op_custom:nn {#1}
4373   }{
4374     \__stex_variables_invoke_custom:nn {#1}
4375   }
4376 }
4377
4378
4379 \cs_new_protected:Nn \__stex_variables_invoke_math:n {

```

```

4380 \peek_charcode_remove:NTF ! {
4381   \peek_charcode_remove:NTF ! {
4382     \peek_charcode:NTF [ {
4383       % TODO throw error
4384     }{
4385       \__stex_variables_invoke_op_custom:nn
4386     }
4387   }{
4388     \__stex_variables_invoke_op:n { #1 }
4389   }
4390 }{
4391   \peek_charcode_remove:NTF * {
4392     \__stex_variables_invoke_custom:nn { #1 }
4393   }{
4394     \__stex_variables_invoke_math_ii:n { #1 }
4395   }
4396 }
4397 }
4398
4399 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4400   \exp_args:Nnx \use:nn {
4401     \def\comp{\_varcomp}
4402     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4403     \bool_set_false:N \l_stex_allow_semantic_bool
4404     \_stex_term_omv:nn {var://#1}{
4405       \comp{ #2 }
4406     }
4407   }{
4408     \_stex_reset:N \comp
4409     \_stex_reset:N \STEXInternalCurrentSymbolStr
4410     \bool_set_true:N \l_stex_allow_semantic_bool
4411   }
4412 }
4413
4414 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4415   \cs_if_exist:cTF {
4416     stex_var_op_notation_ #1 _cs
4417   }{
4418     \exp_args:Nnx \use:nn {
4419       \def\comp{\_varcomp}
4420       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4421       \_stex_term_omv:nn { var://#1 }{
4422         \use:c{stex_var_op_notation_ #1 _cs }
4423       }
4424     }{
4425       \_stex_reset:N \comp
4426       \_stex_reset:N \STEXInternalCurrentSymbolStr
4427     }
4428   }{
4429     \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4430       \__stex_variables_invoke_math_ii:n {#1}
4431     }{
4432       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
4433     }

```

```

4434 }
4435 }
4436
4437 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4438   \cs_if_exist:cTF {
4439     stex_var_notation_#1_cs
4440   }{
4441     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4442       \_stex_reset:N \comp
4443       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4444       \_stex_reset:N \STEXInternalCurrentSymbolStr
4445       \bool_set_true:N \l_stex_allow_semantic_bool
4446     }
4447     \def\comp{\_varcomp}
4448     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4449     \bool_set_false:N \l_stex_allow_semantic_bool
4450     \use:c{stex_var_notation_#1_cs}
4451   }{
4452     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
4453   }
4454 }
4455
4456 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4457   \exp_args:Nnx \use:nn {
4458     \def\comp{\_varcomp}
4459     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4460     \prop_clear:N \l__stex_terms_custom_args_prop
4461     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4462     \prop_get:cnN {
4463       l_stex_symdecl_var://#1 _prop
4464     }{ args } \l_tmpa_str
4465     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4466     \tl_set:Nn \arg { \_stex_terms_arg: }
4467     \str_if_empty:NTF \l_tmpa_str {
4468       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4469     }{
4470       \str_if_in:NnTF \l_tmpa_str b {
4471         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4472       }{
4473         \str_if_in:NnTF \l_tmpa_str B {
4474           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4475         }{
4476           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4477         }
4478       }
4479     }
4480     % TODO check that all arguments exist
4481   }{
4482     \_stex_reset:N \STEXInternalCurrentSymbolStr
4483     \_stex_reset:N \arg
4484     \_stex_reset:N \comp
4485     \_stex_reset:N \l__stex_terms_custom_args_prop
4486     %\bool_set_true:N \l_stex_allow_semantic_bool
4487   }

```

```
4488 }
```

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

30.5 Sequences

```
4489 <@@=stex_sequences>
4490
4491 \cs_new_protected:Nn \stex_invoke_sequence:n {
4492   \peek_charcode_remove:NTF ! {
4493     \stex_term_omv:nn {varseq://#1}{
4494       \exp_args:Nnx \use:nn {
4495         \def\comp{\_varcomp}
4496         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4497         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4498       }{
4499         \_stex_reset:N \comp
4500         \_stex_reset:N \STEXInternalCurrentSymbolStr
4501       }
4502     }
4503   }{
4504     \bool_set_false:N \l_stex_allow_semantic_bool
4505     \def\comp{\_varcomp}
4506     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4507     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4508       \_stex_reset:N \comp
4509       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4510       \_stex_reset:N \STEXInternalCurrentSymbolStr
4511       \bool_set_true:N \l_stex_allow_semantic_bool
4512     }
4513     \use:c { stex_varseq_#1_cs }
4514   }
4515 }
4516 </package>
```

Chapter 31

STEX -Structural Features Implementation

```
4517 ⟨*package⟩
4518
4519 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4520
4521 Warnings and error messages
4522 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4523   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4524 }
4525 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4526   Symbol~#1~not~assigned~in~interpretmodule~#2
4527 }
4528 \msg_new:nnn{stex}{error/unknownstructure}{
4529   No~structure~#1~found!
4530 }
4531
4532 \msg_new:nnn{stex}{error/unknownfield}{
4533   No~field~#1~in~instance~#2~found!\#3
4534 }
4535
4536 \msg_new:nnn{stex}{error/keyval}{
4537   Invalid~key=value~pair:#1
4538 }
4539 \msg_new:nnn{stex}{error/instantiate/missing}{
4540   Assignments~missing~in~instantiate:~#1
4541 }
4542 \msg_new:nnn{stex}{error/incompatible}{
4543   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4544 }
4545
```


31.1 Imports with modification

```

4546 <@@=stex_copymodule>
4547 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4548   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4549     \tl_set:Nn \l_tmpa_tl { #1 }
4550     \__stex_copymodule_get_symbol_from_cs:
4551   }{
4552     % argument is a string
4553     % is it a command name?
4554     \cs_if_exist:cTF { #1 }{
4555       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4556       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4557       \str_if_empty:NTF \l_tmpa_str {
4558         \exp_args:Nx \cs_if_eq:NNTF {
4559           \tl_head:N \l_tmpa_tl
4560         } \stex_invoke_symbol:n {
4561           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4562         }{
4563           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4564         }
4565       } {
4566         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4567       }
4568     }{
4569       % argument is not a command name
4570       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4571       % \l_stex_all_symbols_seq
4572     }
4573   }
4574 }
4575
4576 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4577   \str_set:Nn \l_tmpa_str { #1 }
4578   \bool_set_false:N \l_tmpa_bool
4579   \bool_if:NF \l_tmpa_bool {
4580     \tl_set:Nn \l_tmpa_tl {
4581       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4582     }
4583     \str_set:Nn \l_tmpa_str { #1 }
4584     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4585     \seq_map_inline:Nn #2 {
4586       \str_set:Nn \l_tmpb_str { ##1 }
4587       \str_if_eq:eeT { \l_tmpa_str } {
4588         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4589       } {
4590         \seq_map_break:n {
4591           \tl_set:Nn \l_tmpa_tl {
4592             \str_set:Nn \l_stex_get_symbol_uri_str {
4593               ##1
4594             }
4595           }
4596         }
4597       }

```

```

4598     }
4599     \l_tmpa_tl
4600   }
4601 }
4602
4603 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4604   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4605     { \tl_tail:N \l_tmpa_tl }
4606   \tl_if_single:NTF \l_tmpa_tl {
4607     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4608       \exp_after:wN \str_set:Nn \exp_after:wN
4609         \l_stex_get_symbol_uri_str \l_tmpa_tl
4610       \__stex_copymodule_get_symbol_check:n { #1 }
4611     }{
4612       % TODO
4613       % tail is not a single group
4614     }
4615   }{
4616     % TODO
4617     % tail is not a single group
4618   }
4619 }
4620
4621 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4622   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4623     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4624       :~\seq_use:Nn #1 {,~}
4625     }
4626   }
4627 }
4628
4629 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4630   % import module
4631   \stex_import_module_uri:nn { #1 } { #2 }
4632   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4633   \stex_import_require_module:nnnn
4634     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4635     { \l_stex_import_path_str } { \l_stex_import_name_str }
4636
4637   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4638   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4639
4640   % fields
4641   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4642   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4643     \seq_map_inline:cn {c_stex_module_##1_constants}{
4644       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4645         ##1 ? #####1
4646       }
4647     }
4648   }
4649
4650   % setup prop
4651   \seq_clear:N \l_tmpa_seq

```

```

4652 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4653   name      = \l_stex_current_copymodule_name_str ,
4654   module    = \l_stex_current_module_str ,
4655   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4656   includes  = \l_tmpa_seq %,
4657 % fields    = \l_tmpa_seq
4658 }
4659 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4660   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4661 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4662 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4663
4664 \stex_if_do_html:T {
4665   \begin{stex_annotate_env} {#4} {
4666     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4667   }
4668   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4669 }
4670 }
4671
4672 \cs_new_protected:Nn \stex_copymodule_end:n {
4673   % apply to every field
4674   \def \l_tmpa_cs ##1 ##2 {#1}
4675
4676   \tl_clear:N \__stex_copymodule_module_tl
4677   \tl_clear:N \__stex_copymodule_exec_tl
4678
4679   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4680   \seq_clear:N \__stex_copymodule_fields_seq
4681
4682   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4683     \seq_map_inline:cn {c_stex_module_##1_constants}{
4684
4685       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4686       \l_tmpa_cs{##1}{####1}
4687
4688       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4689         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4690         \stex_if_do_html:T {
4691           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4692             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4693           }
4694         }
4695       }{
4696         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4697       }
4698
4699       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4700       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4701       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4702
4703       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4704         \stex_if_do_html:T {
4705           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4706         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4707     }
4708 }
4709 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4710 }
4711
4712 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4713 \tl_put_right:Nx \__stex_copymodule_module_tl {
4714     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4715     \prop_set_from_keyval:cn {
4716         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4717     }{
4718         \prop_to_keyval:N \l_tmpa_prop
4719     }
4720 }
4721
4722 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4723     \stex_if_do_html:T {
4724         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4725             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4726         }
4727     }
4728     \tl_put_right:Nx \__stex_copymodule_module_tl {
4729         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4730             \stex_invoke_symbol:n {
4731                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4732             }
4733         }
4734     }
4735 }
4736
4737 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4738
4739 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4740     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4741 }
4742
4743 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4744     \stex_if_do_html:TF{
4745         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4746     }{
4747         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4748     }
4749 }
4750 }
4751 }
4752
4753
4754 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4755 \tl_put_left:Nx \__stex_copymodule_module_tl {
4756     \prop_set_from_keyval:cn {
4757         l_stex_copymodule_ \l_stex_current_module_str?l_stex_current_copymodule_name_str _pro
4758     }{
4759         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4760   }
4761 }
4762
4763 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4764   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4765 }
4766
4767 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4768 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4769 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4770
4771 \__stex_copymodule_exec_tl
4772 \stex_if_do_html:T {
4773   \end{stex_annotate_env}
4774 }
4775 }
4776
4777 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4778   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4779   \stex_deactivate_macro:Nn \symdecl {module~environments}
4780   \stex_deactivate_macro:Nn \symdef {module~environments}
4781   \stex_deactivate_macro:Nn \notation {module~environments}
4782   \stex_reactivate_macro:N \assign
4783   \stex_reactivate_macro:N \renamedekl
4784   \stex_reactivate_macro:N \donotcopy
4785   \stex_smsmode_do:
4786 }{
4787   \stex_copymodule_end:n {}
4788 }
4789
4790 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4791   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4792   \stex_deactivate_macro:Nn \symdecl {module~environments}
4793   \stex_deactivate_macro:Nn \symdef {module~environments}
4794   \stex_deactivate_macro:Nn \notation {module~environments}
4795   \stex_reactivate_macro:N \assign
4796   \stex_reactivate_macro:N \renamedekl
4797   \stex_reactivate_macro:N \donotcopy
4798   \stex_smsmode_do:
4799 }{
4800   \stex_copymodule_end:n {
4801     \tl_if_exist:cF {
4802       l__stex_copymodule_copymodule_##1?##2_def_tl
4803     }{
4804       \str_if_eq:eeF {
4805         \prop_item:cn{
4806           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4807         }{ true }{
4808           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4809             ##1?##2
4810           }{\l_stex_current_copymodule_name_str}
4811         }
4812       }
4813     }

```

```

4814 }
4815
4816 \iffalse \begin{stex_annotate_env} \fi
4817 \NewDocumentEnvironment {realization} { 0 } { m } {
4818   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4819   \stex_deactivate_macro:Nn \symdecl {module~environments}
4820   \stex_deactivate_macro:Nn \symdef {module~environments}
4821   \stex_deactivate_macro:Nn \notation {module~environments}
4822   \stex_reactivate_macro:N \donotcopy
4823   \stex_reactivate_macro:N \assign
4824   \stex_smsmode_do:
4825 } {
4826   \stex_import_module_uri:nn { #1 } { #2 }
4827   \tl_clear:N \__stex_copymodule_exec_tl
4828   \tl_set:Nx \__stex_copymodule_module_tl {
4829     \stex_import_require_module:nnnn
4830     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4831     { \l_stex_import_path_str } { \l_stex_import_name_str }
4832   }
4833
4834   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4835     \seq_map_inline:cn {c_stex_module_##1_constants}{
4836       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4837       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4838         \stex_if_do_html:T {
4839           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4840             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4841               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4842             }
4843           }
4844         }
4845         \tl_put_right:Nx \__stex_copymodule_module_tl {
4846           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4847         }
4848       }
4849     }
4850
4851     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4852
4853     \__stex_copymodule_exec_tl
4854     \stex_if_do_html:T {\end{stex_annotate_env}}
4855   }
4856
4857 \NewDocumentCommand \donotcopy { m } {
4858   \str_clear:N \l_stex_import_name_str
4859   \str_set:Nn \l_tmpa_str { #1 }
4860   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4861   \seq_map_inline:Nn \l_stex_all_modules_seq {
4862     \str_set:Nn \l_tmpb_str { ##1 }
4863     \str_if_eq:eeT { \l_tmpa_str } {
4864       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4865     } {
4866       \seq_map_break:n {
4867         \stex_if_do_html:T {

```

```

4868         \stex_if_smsmode:F {
4869             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4870                 \stex_annotate:nnn{domain}{##1}{}}
4871         }
4872     }
4873 }
4874 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4875 }
4876 }
4877 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4878     \str_set:Nn \l_tmpb_str { #####1 }
4879     \str_if_eq:eeT { \l_tmpa_str } {
4880         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4881     } {
4882         \seq_map_break:n {\seq_map_break:n {
4883             \stex_if_do_html:T {
4884                 \stex_if_smsmode:F {
4885                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4886                         \stex_annotate:nnn{domain}{
4887                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4888                         }{}
4889                     }
4890                 }
4891             }
4892             \str_set:Nx \l_stex_import_name_str {
4893                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4894             }
4895         }}
4896     }
4897 }
4898 }
4899 \str_if_empty:NTF \l_stex_import_name_str {
4900     % TODO throw error
4901 }{
4902     \stex_collect_imports:n {\l_stex_import_name_str }
4903     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4904         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4905         \seq_map_inline:cn {c_stex_module_###1_constants}{
4906             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4907             \bool_lazy_any:nT {
4908                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4909                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4910                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4911             }{
4912                 % TODO throw error
4913             }
4914         }
4915     }
4916     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4917     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4918     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4919 }
4920 \stex_smsmode_do:
4921 }

```

```

4922
4923 \NewDocumentCommand \assign { m m }{
4924   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4925   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4926   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4927   \stex_smsmode_do:
4928 }
4929
4930 \keys_define:nn { stex / renamedecl } {
4931   name          .str_set_x:N = \l_stex_renamedecl_name_str
4932 }
4933 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4934   \str_clear:N \l_stex_renamedecl_name_str
4935   \keys_set:nn { stex / renamedecl } { #1 }
4936 }
4937
4938 \NewDocumentCommand \renamedecl { O{} m m }{
4939   \__stex_copymodule_renamedecl_args:n { #1 }
4940   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4941   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4942   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4943   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4944     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4945       \l_stex_get_symbol_uri_str
4946     } }
4947   } {
4948     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4949     \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4950     \prop_set_eq:cc {l_stex_symdecl_
4951       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4952     }_prop
4953     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4954     \seq_set_eq:cc {l_stex_symdecl_
4955       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4956     }_notations
4957     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4958     \prop_put:cnx {l_stex_symdecl_
4959       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4960     }_prop
4961     }{ name }{ \l_stex_renamedecl_name_str }
4962     \prop_put:cnx {l_stex_symdecl_
4963       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4964     }_prop
4965     }{ module }{ \l_stex_current_module_str }
4966     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4967       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4968     }
4969     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4970       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4971     } }
4972   }
4973   \stex_smsmode_do:
4974 }
4975

```



```

4976 \stex_deactivate_macro:Nn \assign {copymodules}
4977 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4978 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4979
4980

```

31.2 The feature environment

structural@feature

```

4981 <@@=stex_features>
4982
4983 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4984   \stex_if_in_module:F {
4985     \msg_set:nnn{stex}{error/nomodule}{
4986       Structural~Feature~has~to~occur~in~a~module:\\
4987       Feature~#2~of~type~#1\\
4988       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4989     }
4990     \msg_error:nn{stex}{error/nomodule}
4991   }
4992
4993   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4994
4995   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4996
4997   \stex_if_do_html:T {
4998     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4999     \stex_annotate_invisible:nnn{header}{\{ #3 }
5000   }
5001 }{
5002   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5003   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5004   \stex_debug:nn{features}{
5005     Feature: \l_stex_last_feature_str
5006   }
5007   \stex_if_do_html:T {
5008     \end{stex_annotate_env}
5009   }
5010 }

```

31.3 Structure

structure

```

5011 <@@=stex_structures>
5012 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5013   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
5014     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
5015   }
5016   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
5017   {#1}{#2}
5018 }
5019

```

```

5020 \keys_define:nn { stex / features / structure } {
5021   name          .str_set_x:N = \l__stex_structures_name_str ,
5022 }
5023
5024 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5025   \str_clear:N \l__stex_structures_name_str
5026   \keys_set:nn { stex / features / structure } { #1 }
5027 }
5028
5029 \NewDocumentEnvironment{mathstructure}{m O{}}{
5030   \__stex_structures_structure_args:n { #2 }
5031   \str_if_empty:NT \l__stex_structures_name_str {
5032     \str_set:Nx \l__stex_structures_name_str { #1 }
5033   }
5034   \stex_suppress_html:n {
5035     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5036     \exp_args:Nx \stex_symdecl_do:nn {
5037       name = \l__stex_structures_name_str ,
5038       def = {\STEXsymbol{module-type}}{
5039         \STEXInternalTermMathOMSiiii {
5040           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5041             { ns } ?
5042           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5043             { name } / \l__stex_structures_name_str - structure
5044         }{}{0}{}
5045       }}
5046     }{ #1 }
5047   }
5048   \exp_args:Nnnx
5049   \begin{structural_feature_module}{ structure }
5050     { \l__stex_structures_name_str }{}
5051   \stex_smsmode_do:
5052 }{
5053   \end{structural_feature_module}
5054   \stex_reset_up_to_module:n \l_stex_last_feature_str
5055   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5056   \seq_clear:N \l_tmpa_seq
5057   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5058     \seq_map_inline:cn{c_stex_module_##1_constants}{
5059       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5060     }
5061   }
5062   \exp_args:Nnno
5063   \prop_gput:cn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5064   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5065   \stex_add_structure_to_current_module:nn
5066     \l__stex_structures_name_str
5067     \l_stex_last_feature_str
5068
5069   \stex_execute_in_module:x {
5070     \tl_set:cn { #1 }{
5071       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
5072     }
5073   }

```

```

5074 }
5075
5076 \cs_new:Nn \stex_invoke_structure:nn {
5077   \stex_invoke_symbol:n { #1?#2 }
5078 }
5079
5080 \cs_new_protected:Nn \stex_get_structure:n {
5081   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5082     \tl_set:Nn \l_tmpa_tl { #1 }
5083     \__stex_structures_get_from_cs:
5084   }{
5085     \cs_if_exist:cTF { #1 }{
5086       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5087       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5088       \str_if_empty:NNTF \l_tmpa_str {
5089         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
5090           \__stex_structures_get_from_cs:
5091         }{
5092           \__stex_structures_get_from_string:n { #1 }
5093         }
5094       }{
5095         \__stex_structures_get_from_string:n { #1 }
5096       }
5097     }{
5098       \__stex_structures_get_from_string:n { #1 }
5099     }
5100   }
5101 }
5102
5103 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5104   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5105     { \tl_tail:N \l_tmpa_tl }
5106   \str_set:Nx \l_tmpa_str {
5107     \exp_after:wN \use_i:nn \l_tmpa_tl
5108   }
5109   \str_set:Nx \l_tmpb_str {
5110     \exp_after:wN \use_ii:nn \l_tmpa_tl
5111   }
5112   \str_set:Nx \l_stex_get_structure_str {
5113     \l_tmpa_str ? \l_tmpb_str
5114   }
5115   \str_set:Nx \l_stex_get_structure_module_str {
5116     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5117   }
5118 }
5119
5120 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5121   \tl_set:Nn \l_tmpa_tl {
5122     \msg_error:nnn{stex}{error/unknownstructure}{#1}
5123   }
5124   \str_set:Nn \l_tmpa_str { #1 }
5125   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5126
5127   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

5128 \prop_if_exist:cT {c_stex_module_##1_structures} {
5129 \prop_map_inline:cn {c_stex_module_##1_structures} {
5130 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5131 \prop_map_break:n{\seq_map_break:n{
5132 \tl_set:Nn \l_tmpa_tl {
5133 \str_set:Nn \l_stex_get_structure_str {##1?####1}
5134 \str_set:Nn \l_stex_get_structure_module_str {####2}
5135 }
5136 }}
5137 }
5138 }
5139 }
5140 }
5141 \l_tmpa_tl
5142 }

```

\instantiate

```

5143
5144 \keys_define:nn { stex / instantiate } {
5145 name .str_set_x:N = \l__stex_structures_name_str
5146 }
5147 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5148 \str_clear:N \l__stex_structures_name_str
5149 \keys_set:nn { stex / instantiate } { #1 }
5150 }
5151
5152 \NewDocumentCommand \instantiate {m O{} m m O{}}{
5153 \beginingroup
5154 \stex_get_structure:n {#3}
5155 \__stex_structures_instantiate_args:n { #2 }
5156 \str_if_empty:NT \l__stex_structures_name_str {
5157 \str_set:Nn \l__stex_structures_name_str { #1 }
5158 }
5159 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5160 \seq_clear:N \l__stex_structures_fields_seq
5161 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5162 \seq_map_inline:Nn \l_stex_collect_imports_seq {
5163 \seq_map_inline:cn {c_stex_module_##1_constants}{
5164 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5165 }
5166 }
5167
5168 \tl_if_empty:nF{#5}{
5169 \seq_set_split:Nnn \l_tmpa_seq , {#5}
5170 \prop_clear:N \l_tmpa_prop
5171 \seq_map_inline:Nn \l_tmpa_seq {
5172 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5173 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5174 \msg_error:nnn{stex}{error/keyval}{##1}
5175 }
5176 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5177 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5178 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5179 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```

```

5180     \exp_args:Nxx \str_if_eq:nnF
5181     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5182     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5183     \msg_error:nnxxxx{stex}{error/incompatible}
5184     {\l__stex_structures_dom_str}
5185     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5186     {\l_stex_get_symbol_uri_str}
5187     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5188   }
5189   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
5190 }
5191 }
5192
5193 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5194   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5195   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5196
5197   \stex_add_constant_to_current_module:n {\l_tmpa_str}
5198   \stex_execute_in_module:x {
5199     \prop_set_from_keyval:cn { l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _p
5200     name   = \l_tmpa_str ,
5201     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5202     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5203     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5204     argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5205   }
5206   \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5207 }
5208
5209 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5210   \stex_find_notation:nn{##1}{}
5211   \stex_execute_in_module:x {
5212     \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
5213   }
5214
5215   \stex_copy_control_sequence_ii:ccN
5216   {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5217   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5218   \l_tmpa_tl
5219   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5220
5221
5222   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5223     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5224     \stex_execute_in_module:x {
5225       \tl_set:cn
5226       {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
5227       { \exp_args:No \exp_not:n \l_tmpa_cs}
5228     }
5229   }
5230
5231 }
5232
5233 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur

```

```

5234 }
5235
5236 \stex_execute_in_module:x {
5237   \prop_set_from_keyval:cn {l_stex_instance\_l_stex_current_module_str?l__stex_structur
5238     domain = \l_stex_get_structure_module_str ,
5239     \prop_to_keyval:N \l_tmpa_prop
5240   }
5241   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?l__stex_structur
5242 }
5243 \stex_debug:nn{instantiate}{
5244   Instance~\l_stex_current_module_str?l__stex_structures_name_str \
5245   \prop_to_keyval:N \l_tmpa_prop
5246 }
5247 \exp_args:Nxx \stex_symdecl_do:nn {
5248   type={\STEXsymbol{module-type}}{
5249     \STEXInternalTermMathOMSiiii {
5250       \l_stex_get_structure_module_str
5251     }{}{0}{}
5252   }}
5253 }{\l__stex_structures_name_str}
5254 % {
5255   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?l__stex_structures
5256   \tl_set:Nn \l_stex_notation_after_do_tl {\_stex_notation_final:}
5257   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
5258 % }
5259 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
5260 \endgroup
5261 \stex_smsmode_do:\ignorespacesandpars
5262 }
5263
5264 \cs_new_protected:Nn \stex_symbol_or_var:n {
5265   \cs_if_exist:cTF{#1}{
5266     \cs_set_eq:Nc \l_tmpa_tl { #1 }
5267     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
5268     \str_if_empty:NTF \l_tmpa_str {
5269       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
5270       \stex_invoke_variable:n {
5271         \bool_set_true:N \l_stex_symbol_or_var_bool
5272         \bool_set_false:N \l_stex_instance_or_symbol_bool
5273         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5274         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5275         \str_set:Nx \l_stex_get_symbol_uri_str {
5276           \exp_after:wN \use:n \l_tmpa_tl
5277         }
5278       }{ % TODO \stex_invoke_varinstance:n
5279         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
5280           \bool_set_true:N \l_stex_symbol_or_var_bool
5281           \bool_set_true:N \l_stex_instance_or_symbol_bool
5282           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5283           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5284           \str_set:Nx \l_stex_get_symbol_uri_str {
5285             \exp_after:wN \use:n \l_tmpa_tl
5286           }
5287         }{

```

```

5288         \bool_set_false:N \l_stex_symbol_or_var_bool
5289         \stex_get_symbol:n{#1}
5290     }
5291 }
5292 }{
5293     \__stex_structures_symbolorvar_from_string:n{ #1 }
5294 }
5295 }{
5296     \__stex_structures_symbolorvar_from_string:n{ #1 }
5297 }
5298 }
5299
5300 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
5301     \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5302         \bool_set_true:N \l_stex_symbol_or_var_bool
5303         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5304     }{
5305         \bool_set_false:N \l_stex_symbol_or_var_bool
5306         \stex_get_symbol:n{#1}
5307     }
5308 }
5309
5310 \keys_define:nn { stex / varinstantiate } {
5311     name .str_set_x:N = \l__stex_structures_name_str,
5312     bind .choices:nn =
5313         {forall,exists}
5314         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5315 }
5316 }
5317 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5318     \str_clear:N \l__stex_structures_name_str
5319     \str_clear:N \l__stex_structures_bind_str
5320     \keys_set:nn { stex / varinstantiate } { #1 }
5321 }
5322
5323 \NewDocumentCommand \varinstantiate {m O{}} m m O{}{}{
5324     \beginingroup
5325         \stex_get_structure:n {#3}
5326         \__stex_structures_varinstantiate_args:n { #2 }
5327         \str_if_empty:NT \l__stex_structures_name_str {
5328             \str_set:Nn \l__stex_structures_name_str { #1 }
5329         }
5330         \stex_if_do_html:TF{
5331             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5332         }{\use:n}
5333         {
5334             \stex_if_do_html:T{
5335                 \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
5336             }
5337             \seq_clear:N \l__stex_structures_fields_seq
5338             \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5339             \seq_map_inline:Nn \l_stex_collect_imports_seq {
5340                 \seq_map_inline:cn {c_stex_module_##1_constants}{
5341                     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }

```

```

5342     }
5343 }
5344 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5345 \prop_clear:N \l_tmpa_prop
5346 \tl_if_empty:nF {#5} {
5347     \seq_set_split:Nnn \l_tmpa_seq , {#5}
5348     \seq_map_inline:Nn \l_tmpa_seq {
5349         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5350         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5351             \msg_error:nnn{stex}{error/keyval}{##1}
5352         }
5353         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
5354         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5355         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq:nn
5356         \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5357         \stex_if_do_html:T{
5358             \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5359                 \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\l_stex_get_symbol_uri_str}{}}
5360         }
5361         \bool_if:NTF \l_stex_symbol_or_var_bool {
5362             \exp_args:Nxx \str_if_eq:nnF
5363                 {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5364                 {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}}{
5365                 \msg_error:nnxxx{stex}{error/incompatible}
5366                 {\l__stex_structures_dom_str}
5367                 {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5368                 {\l_stex_get_symbol_uri_str}
5369                 {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}
5370             }
5371             \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {}}
5372         }{
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_\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_\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_symbol:n {}}
5383         }
5384     }
5385 }
5386 \tl_gclear:N \g__stex_structures_aftergroup_tl
5387 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5388     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_~\l_tmpa_str :~##1}
5389     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5390     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5391         \stex_find_notation:nn{##1}{}
5392         \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
5393             {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5394         \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
5395         \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{

```



```

5396         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5397         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
5398         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5399     }
5400 }
5401
5402 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5403     \prop_set_from_keyval:cn {l_stex_symdecl_ var://\l_tmpa_str_prop}{
5404         name = \l_tmpa_str ,
5405         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5406         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5407         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5408         argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}} ,
5409     }
5410     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
5411     {g__stex_structures_tmpa_\l_tmpa_str_cs}
5412     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
5413     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5414 }
5415 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5416 }
5417 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5418     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
5419         domain = \l_stex_get_structure_module_str ,
5420         \prop_to_keyval:N \l_tmpa_prop
5421     }
5422     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5423     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
5424         \exp_args:Nnx \exp_not:N \use:nn {
5425             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5426             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5427                 \exp_not:n{
5428                     \_varcomp{#4}
5429                 }
5430             }
5431         }}{
5432             \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5433         }
5434     }
5435 }
5436 }
5437 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{g__stex_structures_a
5438 \aftergroup\g__stex_structures_aftergroup_tl
5439 \endgroup
5440 \stex_smsmode_do:\ignorespacesandpars
5441 }
5442
5443 \cs_new_protected:Nn \stex_invoke_instance:n {
5444     \peek_charcode_remove:NTF ! {
5445         \stex_invoke_symbol:n{#1}
5446     }{
5447         \_stex_invoke_instance:nn {#1}
5448     }
5449 }

```

```

5450
5451
5452 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5453   \peek_charcode_remove:NTF ! {
5454     \exp_args:Nnx \use:nn {
5455       \def\comp{\_varcomp}
5456       \use:c{l_stex_varinstance_#1_op_tl}
5457     }{
5458       \_stex_reset:N \comp
5459     }
5460   }{
5461     \_stex_invoke_varinstance:nn {#1}
5462   }
5463 }
5464
5465 \cs_new_protected:Nn \stex_invoke_instance:nn {
5466   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5467     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5468   }{
5469     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5470     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5471       \prop_to_keyval:N \l_tmpa_prop
5472     }
5473   }
5474 }
5475
5476 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
5477   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5478     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5479     \l_tmpa_tl
5480   }{
5481     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
5482     }
5483   }

```

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

\stex_invoke_structure:nnn

```

5484 % #1: URI of the instance
5485 % #2: URI of the instantiated module
5486 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5487   \tl_if_empty:nTF{ #3 }{
5488     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5489       c_stex_feature_ #2 _prop
5490     }
5491     \tl_clear:N \l_tmpa_tl
5492     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5493     \seq_map_inline:Nn \l_tmpa_seq {
5494       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5495       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5496       \cs_if_exist:cT {
5497         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5498       }{
5499         \tl_if_empty:NF \l_tmpa_tl {

```

```

5500         \tl_put_right:Nn \l_tmpa_tl {,}
5501     }
5502     \tl_put_right:Nx \l_tmpa_tl {
5503         \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5504     }
5505 }
5506 }
5507 \exp_args:No \mathstruct \l_tmpa_tl
5508 }{
5509     \stex_invoke_symbol:n{#1/#3}
5510 }
5511 }

```

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

```

5512 </package>

```

Chapter 32

STEX -Statements Implementation

```
5513 <*package>
5514
5515 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5516
5517 <@@=stex_statements>
    Warnings and error messages
5518
\titleemph
5519 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

32.1 Definitions

definiendum

```
5520 \keys_define:nn {stex / definiendum }{
5521   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5522   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5523   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5524   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5525 }
5526 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5527   \str_clear:N \l__stex_statements_definiendum_root_str
5528   \tl_clear:N \l__stex_statements_definiendum_post_tl
5529   \str_clear:N \l__stex_statements_definiendum_gfa_str
5530   \keys_set:nn { stex / definiendum }{ #1 }
5531 }
5532 \NewDocumentCommand \definiendum { O{} m m } {
5533   \__stex_statements_definiendum_args:n { #1 }
5534   \stex_get_symbol:n { #2 }
5535   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5536   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5537     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5538     \tl_set:Nn \l_tmpa_tl { #3 }
5539   } {
5540     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5541     \tl_set:Nn \l_tmpa_tl {
5542       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5543     }
5544   }
5545 } {
5546   \tl_set:Nn \l_tmpa_tl { #3 }
5547 }
5548
5549 % TODO root
5550 \stex_html_backend:TF {
5551   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5552 } {
5553   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5554 }
5555 }
5556 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 43.)

definame

```

5557
5558 \NewDocumentCommand \definame { 0{ } m } {
5559   \__stex_statements_definiendum_args:n { #1 }
5560   % TODO: root
5561   \stex_get_symbol:n { #2 }
5562   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5563   \str_set:Nx \l_tmpa_str {
5564     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5565   }
5566   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5567   \stex_html_backend:TF {
5568     \stex_if_do_html:T {
5569       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5570         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5571       }
5572     }
5573   } {
5574     \exp_args:Nnx \defemph@uri {
5575       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5576     } { \l_stex_get_symbol_uri_str }
5577   }
5578 }
5579 \stex_deactivate_macro:Nn \definame {definition~environments}
5580
5581 \NewDocumentCommand \Definame { 0{ } m } {
5582   \__stex_statements_definiendum_args:n { #1 }
5583   \stex_get_symbol:n { #2 }
5584   \str_set:Nx \l_tmpa_str {
5585     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5586   }
5587   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5588 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5589 \stex_html_backend:TF {
5590   \stex_if_do_html:T {
5591     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5592       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5593     }
5594   }
5595 } {
5596   \exp_args:Nnx \defemph@uri {
5597     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5598   } { \l_stex_get_symbol_uri_str }
5599 }
5600 }
5601 \stex_deactivate_macro:Nn \Definame {definition-environments}
5602
5603 \NewDocumentCommand \premise { m }{
5604   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5605 }
5606 \NewDocumentCommand \conclusion { m }{
5607   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5608 }
5609 \NewDocumentCommand \definiens { 0{} m }{
5610   \str_clear:N \l_stex_get_symbol_uri_str
5611   \tl_if_empty:nF {#1} {
5612     \stex_get_symbol:n { #1 }
5613   }
5614   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5615     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5616       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5617     }{
5618       % TODO throw error
5619     }
5620   }
5621   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5622   {\l_stex_current_module_str}{
5623     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5624   }{true}{
5625     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5626     \exp_args:Nx \stex_add_to_current_module:n {
5627       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5628     }
5629   }
5630 }
5631 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5632 }
5633
5634 \NewDocumentCommand \varbindforall {m}{
5635   \stex_symbol_or_var:n {#1}
5636   \bool_if:NTF\l_stex_symbol_or_var_bool{
5637     \stex_if_do_html:T {
5638       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5639     }
5640   }{
5641     % todo throw error

```

```

5642 }
5643 }
5644
5645 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5646 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5647 \stex_deactivate_macro:Nn \definiens {definition~environments}
5648 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5649

```

(End definition for definame. This function is documented on page 43.)

sdefinition

```

5650
5651 \keys_define:nn {stex / sdefinition }{
5652   type      .str_set_x:N = \sdefinitiontype,
5653   id        .str_set_x:N = \sdefinitionid,
5654   name      .str_set_x:N = \sdefinitionname,
5655   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5656   title     .tl_set:N     = \sdefinitiontitle
5657 }
5658 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5659   \str_clear:N \sdefinitiontype
5660   \str_clear:N \sdefinitionid
5661   \str_clear:N \sdefinitionname
5662   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5663   \tl_clear:N \sdefinitiontitle
5664   \keys_set:nn { stex / sdefinition }{ #1 }
5665 }
5666
5667 \NewDocumentEnvironment{sdefinition}{0{}}{
5668   \__stex_statements_sdefinition_args:n{ #1 }
5669   \stex_reactivate_macro:N \definiendum
5670   \stex_reactivate_macro:N \definame
5671   \stex_reactivate_macro:N \Definame
5672   \stex_reactivate_macro:N \premise
5673   \stex_reactivate_macro:N \definiens
5674   \stex_reactivate_macro:N \varbindforall
5675   \stex_if_smsmode:F{
5676     \seq_clear:N \l_tmpb_seq
5677     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5678       \tl_if_empty:nF{ ##1 }{
5679         \stex_get_symbol:n { ##1 }
5680         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5681           \l_stex_get_symbol_uri_str
5682         }
5683       }
5684     }
5685     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5686     \exp_args:Nnnx
5687     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5688     \str_if_empty:NF \sdefinitiontype {
5689       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5690     }
5691     \str_if_empty:NF \sdefinitionname {

```

```

5692     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5693   }
5694   \clist_set:No \l_tmpa_clist \sdefinitiontype
5695   \tl_clear:N \l_tmpa_tl
5696   \clist_map_inline:Nn \l_tmpa_clist {
5697     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5698       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5699     }
5700   }
5701   \tl_if_empty:NTF \l_tmpa_tl {
5702     \__stex_statements_sdefinition_start:
5703   }{
5704     \l_tmpa_tl
5705   }
5706 }
5707 \stex_ref_new_doc_target:n \sdefinitionid
5708 \stex_smsmode_do:
5709 ){
5710   \stex_suppress_html:n {
5711     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5712   }
5713   \stex_if_smsmode:F {
5714     \clist_set:No \l_tmpa_clist \sdefinitiontype
5715     \tl_clear:N \l_tmpa_tl
5716     \clist_map_inline:Nn \l_tmpa_clist {
5717       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5718         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5719       }
5720     }
5721     \tl_if_empty:NTF \l_tmpa_tl {
5722       \__stex_statements_sdefinition_end:
5723     }{
5724       \l_tmpa_tl
5725     }
5726     \end{stex_annotate_env}
5727   }
5728 }

```

\stexpatchdefinition

```

5729 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5730   \stex_par:\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5731     ~(\sdefinitiontitle)
5732   }~}
5733 }
5734 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5735
5736 \newcommand\stexpatchdefinition[3] [] {
5737   \str_set:Nx \l_tmpa_str{ #1 }
5738   \str_if_empty:NTF \l_tmpa_str {
5739     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5740     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5741   }{
5742     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5743     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```



```

5744     }
5745 }

```

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

`\inlinedef` inline:

```

5746 \keys_define:nn {stex / inlinedef }{
5747   type      .str_set_x:N = \sdefinitiontype,
5748   id        .str_set_x:N = \sdefinitionid,
5749   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5750   name      .str_set_x:N = \sdefinitionname
5751 }
5752 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5753   \str_clear:N \sdefinitiontype
5754   \str_clear:N \sdefinitionid
5755   \str_clear:N \sdefinitionname
5756   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5757   \keys_set:nn { stex / inlinedef }{ #1 }
5758 }
5759 \NewDocumentCommand \inlinedef { 0{} m } {
5760   \begingroup
5761   \__stex_statements_inlinedef_args:n{ #1 }
5762   \stex_reactivate_macro:N \definiendum
5763   \stex_reactivate_macro:N \definame
5764   \stex_reactivate_macro:N \Definame
5765   \stex_reactivate_macro:N \premise
5766   \stex_reactivate_macro:N \definiens
5767   \stex_reactivate_macro:N \varbindforall
5768   \stex_ref_new_doc_target:n \sdefinitionid
5769   \stex_if_smsmode:TF{\stex_suppress_html:n {
5770     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5771   }}{
5772     \seq_clear:N \l_tmpb_seq
5773     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5774       \tl_if_empty:nF{ ##1 }{
5775         \stex_get_symbol:n { ##1 }
5776         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5777           \l_stex_get_symbol_uri_str
5778         }
5779       }
5780     }
5781     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5782     \exp_args:Nnx
5783     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5784       \str_if_empty:NF \sdefinitiontype {
5785         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5786       }
5787       #2
5788       \str_if_empty:NF \sdefinitionname {
5789         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5790         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5791       }
5792     }
5793   }

```



```

5841 \clist_map_inline:Nn \l_tmpa_clist {
5842   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5843     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5844   }
5845 }
5846 \tl_if_empty:NTF \l_tmpa_tl {
5847   \__stex_statements_sassertion_start:
5848 }{
5849   \l_tmpa_tl
5850 }
5851 }
5852 \str_if_empty:NTF \sassertionid {
5853   \str_if_empty:NF \sassertionname {
5854     \stex_ref_new_doc_target:n {}
5855   }
5856 } {
5857   \stex_ref_new_doc_target:n \sassertionid
5858 }
5859 \stex_smsmode_do:
5860 ){
5861   \str_if_empty:NF \sassertionname {
5862     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5863     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5864   }
5865   \stex_if_smsmode:F {
5866     \clist_set:No \l_tmpa_clist \sassertiontype
5867     \tl_clear:N \l_tmpa_tl
5868     \clist_map_inline:Nn \l_tmpa_clist {
5869       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5870         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5871       }
5872     }
5873     \tl_if_empty:NTF \l_tmpa_tl {
5874       \__stex_statements_sassertion_end:
5875     }{
5876       \l_tmpa_tl
5877     }
5878     \end{stex_annotate_env}
5879   }
5880 }

```

\stexpatchassertion

```

5881
5882 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5883   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5884     (\sassertiontitle)
5885   }~}
5886 }
5887 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5888
5889 \newcommand\stexpatchassertion[3] [] {
5890   \str_set:Nx \l_tmpa_str{ #1 }
5891   \str_if_empty:NTF \l_tmpa_str {
5892     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

5893     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5894   }{
5895     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5896     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5897   }
5898 }

```

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

`\inlineass` inline:

```

5899 \keys_define:nn {stex / inlineass }{
5900   type      .str_set_x:N = \sassertiontype,
5901   id        .str_set_x:N = \sassertionid,
5902   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5903   name      .str_set_x:N = \sassertionname
5904 }
5905 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5906   \str_clear:N \sassertiontype
5907   \str_clear:N \sassertionid
5908   \str_clear:N \sassertionname
5909   \clist_clear:N \l__stex_statements_sassertion_for_clist
5910   \keys_set:nn { stex / inlineass }{ #1 }
5911 }
5912 \NewDocumentCommand \inlineass { 0{} m } {
5913   \beginngroup
5914   \stex_reactivate_macro:N \premise
5915   \stex_reactivate_macro:N \conclusion
5916   \stex_reactivate_macro:N \varbindforall
5917   \__stex_statements_inlineass_args:n{ #1 }
5918   \str_if_empty:NTF \sassertionid {
5919     \str_if_empty:NF \sassertionname {
5920       \stex_ref_new_doc_target:n {}
5921     }
5922   } {
5923     \stex_ref_new_doc_target:n \sassertionid
5924   }
5925
5926   \stex_if_smsmode:TF{
5927     \str_if_empty:NF \sassertionname {
5928       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
5929     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5930   }
5931 }{
5932   \seq_clear:N \l_tmpb_seq
5933   \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5934     \tl_if_empty:nF{ ##1 }{
5935       \stex_get_symbol:n { ##1 }
5936       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5937         \l_stex_get_symbol_uri_str
5938       }
5939     }
5940   }
5941   \exp_args:Nnx
5942   \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{

```

```

5943     \str_if_empty:NF \sassertiontype {
5944       \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{ }
5945     }
5946     #2
5947     \str_if_empty:NF \sassertionname {
5948       \stex_suppress_html:n{\stex_symdecl_do:nn{ }{\sassertionname}}
5949       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5950       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5951     }
5952   }
5953 }
5954 \endgroup
5955 \stex_smsmode_do:
5956 }

```

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

32.3 Examples

`sexample`

```

5957
5958 \keys_define:nn {stex / sexample }{
5959   type      .str_set_x:N = \exampletype,
5960   id        .str_set_x:N = \sexampleid,
5961   title     .tl_set:N    = \sexampletitle,
5962   name      .str_set_x:N = \sexamplename ,
5963   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5964 }
5965 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5966   \str_clear:N \sexampletype
5967   \str_clear:N \sexampleid
5968   \str_clear:N \sexamplename
5969   \tl_clear:N \sexampletitle
5970   \clist_clear:N \l__stex_statements_sexample_for_clist
5971   \keys_set:nn { stex / sexample }{ #1 }
5972 }
5973
5974 \NewDocumentEnvironment{sexample}{0{}}{
5975   \__stex_statements_sexample_args:n{ #1 }
5976   \stex_reactivate_macro:N \premise
5977   \stex_reactivate_macro:N \conclusion
5978   \stex_if_smsmode:F {
5979     \seq_clear:N \l_tmpb_seq
5980     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5981       \tl_if_empty:nF{ ##1 }{
5982         \stex_get_symbol:n { ##1 }
5983         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5984           \l_stex_get_symbol_uri_str
5985         }
5986       }
5987     }
5988     \exp_args:Nnnx
5989     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```

```

5990 \str_if_empty:NF \sexamplotype {
5991   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5992 }
5993 \str_if_empty:NF \sexamplename {
5994   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5995 }
5996 \clist_set:No \l_tmpa_clist \sexamplotype
5997 \tl_clear:N \l_tmpa_tl
5998 \clist_map_inline:Nn \l_tmpa_clist {
5999   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6000     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
6001   }
6002 }
6003 \tl_if_empty:NTF \l_tmpa_tl {
6004   \__stex_statements_sexample_start:
6005 }{
6006   \l_tmpa_tl
6007 }
6008 }
6009 \str_if_empty:NF \sexampleid {
6010   \stex_ref_new_doc_target:n \sexampleid
6011 }
6012 \stex_smsmode_do:
6013 }{
6014   \str_if_empty:NF \sexamplename {
6015     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6016   }
6017   \stex_if_smsmode:F {
6018     \clist_set:No \l_tmpa_clist \sexamplotype
6019     \tl_clear:N \l_tmpa_tl
6020     \clist_map_inline:Nn \l_tmpa_clist {
6021       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6022         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6023       }
6024     }
6025     \tl_if_empty:NTF \l_tmpa_tl {
6026       \__stex_statements_sexample_end:
6027     }{
6028       \l_tmpa_tl
6029     }
6030     \end{stex_annotate_env}
6031   }
6032 }

```

\stexpatchexample

```

6033
6034 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6035   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
6036     (\sexampltitle)
6037   }~}
6038 }
6039 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
6040
6041 \newcommand\stexpatchexample[3]{} {

```

```

6042 \str_set:Nx \l_tmpa_str{ #1 }
6043 \str_if_empty:NTF \l_tmpa_str {
6044   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6045   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6046 }{
6047   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6048   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6049 }
6050 }

```

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

`\inlineex` inline:

```

6051 \keys_define:nn {stex / inlineex }{
6052   type      .str_set_x:N = \sexamplotype,
6053   id        .str_set_x:N = \sexampleid,
6054   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
6055   name      .str_set_x:N = \sexamplename
6056 }
6057 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6058   \str_clear:N \sexamplotype
6059   \str_clear:N \sexampleid
6060   \str_clear:N \sexamplename
6061   \clist_clear:N \l__stex_statements_sexample_for_clist
6062   \keys_set:nn { stex / inlineex }{ #1 }
6063 }
6064 \NewDocumentCommand \inlineex { 0{} m } {
6065   \begingroup
6066   \stex_reactivate_macro:N \premise
6067   \stex_reactivate_macro:N \conclusion
6068   \__stex_statements_inlineex_args:n{ #1 }
6069   \str_if_empty:NF \sexampleid {
6070     \stex_ref_new_doc_target:n \sexampleid
6071   }
6072   \stex_if_smsmode:TF{
6073     \str_if_empty:NF \sexamplename {
6074       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
6075   }
6076 }{
6077   \seq_clear:N \l_tmpb_seq
6078   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6079     \tl_if_empty:nF{ ##1 }{
6080       \stex_get_symbol:n { ##1 }
6081       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6082         \l_stex_get_symbol_uri_str
6083       }
6084     }
6085   }
6086   \exp_args:Nnx
6087   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6088     \str_if_empty:NF \sexamplotype {
6089       \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
6090     }
6091     #2

```

```

6092     \str_if_empty:NF \sexamplename {
6093       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6094       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
6095     }
6096   }
6097 }
6098 \endgroup
6099 \stex_smsmode_do:
6100 }

```

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

32.4 Logical Paragraphs

`sparagraph`

```

6101 \keys_define:nn { stex / sparagraph } {
6102   id      .str_set:x:N = \sparagraphid ,
6103   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
6104   type    .str_set:x:N = \sparagraphtype ,
6105   for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
6106   from    .tl_set:N     = \sparagraphfrom ,
6107   to      .tl_set:N     = \sparagraphto ,
6108   start   .tl_set:N     = \l_stex_sparagraph_start_tl ,
6109   name    .str_set:N    = \sparagraphname ,
6110   imports .tl_set:N     = \l__stex_statements_sparagraph_imports_tl
6111 }
6112
6113 \cs_new_protected:Nn \stex_sparagraph_args:n {
6114   \tl_clear:N \l_stex_sparagraph_title_tl
6115   \tl_clear:N \sparagraphfrom
6116   \tl_clear:N \sparagraphto
6117   \tl_clear:N \l_stex_sparagraph_start_tl
6118   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6119   \str_clear:N \sparagraphid
6120   \str_clear:N \sparagraphtype
6121   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6122   \str_clear:N \sparagraphname
6123   \keys_set:nn { stex / sparagraph } { #1 }
6124 }
6125 \newif\if@in@omtext\@in@omtextfalse
6126
6127 \NewDocumentEnvironment {sparagraph} { 0{ } } {
6128   \stex_sparagraph_args:n { #1 }
6129   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6130     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6131   }{
6132     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6133   }
6134   \@in@omtexttrue
6135   \stex_if_smsmode:F {
6136     \seq_clear:N \l_tmpb_seq
6137     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6138       \tl_if_empty:nF{ ##1 }{

```



```

6139     \stex_get_symbol:n { ##1 }
6140     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6141         \l_stex_get_symbol_uri_str
6142     }
6143 }
6144 }
6145 \exp_args:Nnnx
6146 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6147 \str_if_empty:NF \sparagraphtype {
6148     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6149 }
6150 \str_if_empty:NF \sparagraphfrom {
6151     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6152 }
6153 \str_if_empty:NF \sparagraphto {
6154     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6155 }
6156 \str_if_empty:NF \sparagraphname {
6157     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6158 }
6159 \clist_set:No \l_tmpa_clist \sparagraphtype
6160 \tl_clear:N \l_tmpa_tl
6161 \clist_map_inline:Nn \sparagraphtype {
6162     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6163         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
6164     }
6165 }
6166 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6167 \tl_if_empty:NTF \l_tmpa_tl {
6168     \__stex_statements_sparagraph_start:
6169 }{
6170     \l_tmpa_tl
6171 }
6172 }
6173 \clist_set:No \l_tmpa_clist \sparagraphtype
6174 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6175     {
6176         \stex_reactivate_macro:N \definiendum
6177         \stex_reactivate_macro:N \definame
6178         \stex_reactivate_macro:N \Definame
6179         \stex_reactivate_macro:N \premise
6180         \stex_reactivate_macro:N \definiens
6181     }
6182 \str_if_empty:NTF \sparagraphid {
6183     \str_if_empty:NTF \sparagraphname {
6184         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6185             \stex_ref_new_doc_target:n {}
6186         }
6187     } {
6188         \stex_ref_new_doc_target:n {}
6189     }
6190 } {
6191     \stex_ref_new_doc_target:n \sparagraphid
6192 }

```

```

6193 \exp_args:NNx
6194 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6195   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6196     \tl_if_empty:nF{ ##1 }{
6197       \stex_get_symbol:n { ##1 }
6198       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6199     }
6200   }
6201 }
6202 \stex_smsmode_do:
6203 \ignorespacesandpars
6204 }{
6205   \str_if_empty:NF \sparagraphname {
6206     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6207     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6208   }
6209   \stex_if_smsmode:F {
6210     \clist_set:No \l_tmpa_clist \sparagraphtype
6211     \tl_clear:N \l_tmpa_tl
6212     \clist_map_inline:Nn \l_tmpa_clist {
6213       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
6214         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
6215       }
6216     }
6217     \tl_if_empty:NTF \l_tmpa_tl {
6218       \__stex_statements_sparagraph_end:
6219     }{
6220       \l_tmpa_tl
6221     }
6222     \end{stex_annotate_env}
6223   }
6224 }

```

\stexpatchparagraph

```

6225
6226 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6227   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6228     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6229       \titleemph{\l_stex_sparagraph_title_tl}:~
6230     }
6231   }{
6232     \titleemph{\l_stex_sparagraph_start_tl}~
6233   }
6234 }
6235 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6236
6237 \newcommand\stexpatchparagraph[3] [] {
6238   \str_set:Nx \l_tmpa_str{ #1 }
6239   \str_if_empty:NTF \l_tmpa_str {
6240     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6241     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
6242   }{
6243     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
6244     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

6245     }
6246 }
6247
6248 \keys_define:nn { stex / inlinepara } {
6249   id      .str_set:N = \sparagraphid ,
6250   type    .str_set:N = \sparagraphtype ,
6251   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6252   from    .tl_set:N   = \sparagraphfrom ,
6253   to      .tl_set:N   = \sparagraphto ,
6254   name    .str_set:N   = \sparagraphname
6255 }
6256 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6257   \tl_clear:N \sparagraphfrom
6258   \tl_clear:N \sparagraphto
6259   \str_clear:N \sparagraphid
6260   \str_clear:N \sparagraphtype
6261   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6262   \str_clear:N \sparagraphname
6263   \keys_set:nn { stex / inlinepara } { #1 }
6264 }
6265 \NewDocumentCommand \inlinepara { O{} m } {
6266   \begingroup
6267   \__stex_statements_inlinepara_args:n{ #1 }
6268   \clist_set:Nn \l_tmpa_clist \sparagraphtype
6269   \str_if_empty:NTF \sparagraphid {
6270     \str_if_empty:NTF \sparagraphname {
6271       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6272         \stex_ref_new_doc_target:n {}
6273       }
6274     } {
6275       \stex_ref_new_doc_target:n {}
6276     }
6277   } {
6278     \stex_ref_new_doc_target:n \sparagraphid
6279   }
6280   \stex_if_smsmode:TF{
6281     \str_if_empty:NF \sparagraphname {
6282       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6283     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6284   }
6285 }{
6286   \seq_clear:N \l_tmpb_seq
6287   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6288     \tl_if_empty:nF{ ##1 }{
6289       \stex_get_symbol:n { ##1 }
6290       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6291         \l_stex_get_symbol_uri_str
6292       }
6293     }
6294   }
6295   \exp_args:Nnx
6296   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6297     \str_if_empty:NF \sparagraphtype {
6298       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

6299     }
6300     \str_if_empty:NF \sparagraphfrom {
6301       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6302     }
6303     \str_if_empty:NF \sparagraphto {
6304       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6305     }
6306     \str_if_empty:NF \sparagraphname {
6307       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6308       \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6309       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6310     }
6311     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6312       \clist_map_inline:Nn \l_tmpb_seq {
6313         \stex_ref_new_sym_target:n {##1}
6314       }
6315     }
6316     #2
6317   }
6318 }
6319 \endgroup
6320 \stex_smsmode_do:
6321 }
6322

```

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

```

6323 </package>

```

Chapter 33

The Implementation

```
6324 <*package>
6325 <@@=stex_sproof>
6326
6327 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
6328
```

33.1 Proofs

We first define some keys for the proof environment.

```
6329 \keys_define:nn { stex / spf } {
6330   id          .str_set_x:N = \spfid,
6331   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
6332   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
6333   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
6334   type        .str_set_x:N = \spftype,
6335   title       .tl_set:N    = \spftitle,
6336   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
6337   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
6338   term        .tl_set:N    = \l__stex_sproof_spf_term_tl,
6339   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
6340   hide        .bool_set:N  = \l__stex_sproof_spf_hide_bool
6341 }
6342 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
6343   \str_clear:N \spfid
6344   \tl_clear:N \l__stex_sproof_spf_for_tl
6345   \tl_clear:N \l__stex_sproof_spf_from_tl
6346   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6347   \str_clear:N \spftype
6348   \tl_clear:N \spftitle
6349   \tl_clear:N \l__stex_sproof_spf_continues_tl
6350   \tl_clear:N \l__stex_sproof_spf_term_tl
6351   \tl_clear:N \l__stex_sproof_spf_functions_tl
6352   \tl_clear:N \l__stex_sproof_spf_method_tl
6353   \bool_set_false:N \l__stex_sproof_spf_hide_bool
6354   \keys_set:nn { stex / spf }{ #1 }
6355 }
6356 \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`

```
6357 \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.

```
6358 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6359 \cs_new_protected:Npn\sproofnumber {
6360   \int_set:Nn \l_tmpa_int {1}
6361   \bool_while_do:nn {
6362     \int_compare_p:nNn {
6363       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6364     } > 0
6365   }{
6366     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6367     \int_incr:N \l_tmpa_int
6368   }
6369 }
6370 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
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     \int_incr:N \l_tmpa_int
6378   }
6379   \int_compare:nNnF \l_tmpa_int = 1 {
6380     \int_decr:N \l_tmpa_int
6381   }
6382   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6383     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6384   }
6385 }
6386
6387 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6388   \int_set:Nn \l_tmpa_int {1}
6389   \bool_while_do:nn {
6390     \int_compare_p:nNn {
6391       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6392     } > 0
6393   }{
6394     \int_incr:N \l_tmpa_int
6395   }
6396   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6397 }
6398
```

```

6399 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6400   \int_set:Nn \l_tmpa_int {1}
6401   \bool_while_do:nn {
6402     \int_compare_p:nNn {
6403       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6404     } > 0
6405   }{
6406     \int_incr:N \l_tmpa_int
6407   }
6408   \int_decr:N \l_tmpa_int
6409   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6410 }

```

\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

```

6411 \def\sproof@box{
6412   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6413 }
6414 \def\sproofend{
6415   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6416     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6417   }
6418 }

```

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

spf@*kw

```

6419 \def\spf@proofsketch@kw{Proof~Sketch}
6420 \def\spf@proof@kw{Proof}
6421 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6422 \AddToHook{begindocument}{
6423   \ltx@ifpackageloaded{babel}{
6424     \makeatletter
6425     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6426     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6427       \input{sproof-ngerman.ldf}
6428     }
6429     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6430       \input{sproof-finnish.ldf}
6431     }
6432     \clist_if_in:NnT \l_tmpa_clist {french}{
6433       \input{sproof-french.ldf}
6434     }
6435     \clist_if_in:NnT \l_tmpa_clist {russian}{
6436       \input{sproof-russian.ldf}
6437     }
6438     \makeatother
6439   }{}
6440 }

```

spfsketch

```

6441 \newcommand\spfsketch[2] [] {
6442   \begin{group}
6443   \let \premise \stex_proof_premise:
6444   \_stex_sproof_spf_args:n{#1}
6445   \stex_if_smsmode:TF {
6446     \str_if_empty:NF \spfid {
6447       \stex_ref_new_doc_target:n \spfid
6448     }
6449   }{
6450     \seq_clear:N \l_tmpa_seq
6451     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6452       \tl_if_empty:nF{ ##1 }{
6453         \stex_get_symbol:n { ##1 }
6454         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6455           \l_stex_get_symbol_uri_str
6456         }
6457       }
6458     }
6459     \exp_args:Nnx
6460     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {},,}{
6461       \str_if_empty:NF \spftype {
6462         \stex_annotate_invisible:nnn{type}{\spftype}{
6463         }
6464       }
6465       \clist_set:Nn \l_tmpa_clist \spftype
6466       \tl_set:Nn \l_tmpa_tl {
6467         \titleemph{
6468           \tl_if_empty:NTF \spftitle {
6469             \spf@proofsketch@kw
6470           }{
6471             \spftitle
6472           }
6473         }
6474       }
6475       \clist_map_inline:Nn \l_tmpa_clist {
6476         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6477           \tl_clear:N \l_tmpa_tl
6478         }
6479       }
6480       \str_if_empty:NF \spfid {
6481         \stex_ref_new_doc_target:n \spfid
6482       }
6483       \l_tmpa_tl #2 \sproofend
6484     }
6485   }
6486   \endgroup
6487 }
6488

```

(End definition for *spfsketch*. This function is documented on page 49.)

```

\_stex_sproof_maybe_comment:
\_stex_sproof_maybe_comment_end:
\_stex_sproof_start_comment:
6489 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```



```

6490
6491 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6492   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6493     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6494   }
6495 }
6496 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6497   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6498 }
6499 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6500   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6501 }
6502

```

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

`\stexcommentfont`

```

6503 \cs_new_protected:Npn \stexcommentfont {
6504   \small\itshape
6505 }

```

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

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

```

6506 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6507   \seq_clear:N \l_tmpa_seq
6508   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6509     \tl_if_empty:NF{ ##1 }{
6510       \stex_get_symbol:n { ##1 }
6511       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6512         \l_stex_get_symbol_uri_str
6513       }
6514     }
6515   }
6516   \exp_args:Nnnx
6517   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6518   \str_if_empty:NF \spftype {
6519     \stex_annotate_invisible:nnn{type}{\spftype}{}
6520   }
6521   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6522   \str_if_empty:NF \spfid {
6523     \stex_ref_new_doc_target:n \spfid
6524   }
6525   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6526   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6527     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6528   }
6529   \begin{list}{}{
6530     \setlength\topsep{0pt}
6531     \setlength\parsep{0pt}
6532     \setlength\rightmargin{0pt}

```

```

6533 } \__stex_sproof_maybe_comment:
6534 }
6535 \cs_new_protected:Nn \__stex_sproof_end_env:n {
6536   \stex_if_smsmode:F{
6537     \__stex_sproof_maybe_comment_end:
6538     \end{list}
6539     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6540       \stex_html_backend:F{\egroup}
6541     }
6542     \clist_set:Nn \l_tmpa_clist \spftype
6543     #1
6544     \end{stex_annotate_env}
6545     \end{stex_annotate_env}
6546   }
6547 }
6548 }
6549 \NewDocumentEnvironment{sproof}{s O{} m}{
6550   \intarray_gzero:N \l__stex_sproof_counter_intarray
6551   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6552   \stex_reactivate_macro:N \yield
6553   \stex_reactivate_macro:N \eqstep
6554   \stex_reactivate_macro:N \assumption
6555   \stex_reactivate_macro:N \conclude
6556   \stex_reactivate_macro:N \spfstep
6557   \__stex_sproof_spf_args:n{#2}
6558   \stex_if_smsmode:TF {
6559     \str_if_empty:NF \spfid {
6560       \stex_ref_new_doc_target:n \spfid
6561     }
6562   }{
6563     \__stex_sproof_start_env:nnn{sproof}{#3}{
6564       \clist_set:Nn \l_tmpa_clist \spftype
6565       \tl_clear:N \l_tmpa_tl
6566       \clist_map_inline:Nn \l_tmpa_clist {
6567         \tl_if_exist:cT {\__stex_sproof_sproof_##1_start:}{
6568           \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_start:}}
6569         }
6570         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6571           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6572         }
6573       }
6574       \tl_if_empty:NTF \l_tmpa_tl {
6575         \__stex_sproof_sproof_start:
6576       }{
6577         \l_tmpa_tl
6578       }
6579     }
6580   }
6581   \stex_smsmode_do:
6582 }{\__stex_sproof_end_env:n{
6583   \tl_clear:N \l_tmpa_tl
6584   \clist_map_inline:Nn \l_tmpa_clist {
6585     \tl_if_exist:cT {\__stex_sproof_sproof_##1_end:}{
6586       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_end:}}

```

```

6587     }
6588   }
6589   \tl_if_empty:NTF \l_tmpa_tl {
6590     \__stex_sproof_sproof_end:
6591   }{
6592     \l_tmpa_tl
6593   }
6594 }
6595 \NewDocumentEnvironment{subproof}{s O{} m}{
6596   \__stex_sproof_spf_args:n{#2}
6597   \stex_if_smsmode:TF {
6598     \str_if_empty:NF \spfid {
6599       \stex_ref_new_doc_target:n \spfid
6600     }
6601   }{
6602     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6603   }
6604   \__stex_sproof_add_counter:
6605   \stex_smsmode_do:
6606 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6607   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6608     \__stex_sproof_inc_counter:
6609   }
6610   \aftergroup\__stex_sproof_maybe_comment:
6611 }
6612 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6613
6614 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6615   \par\noindent\titleemph{
6616     \tl_if_empty:NTF \spftype {
6617       \spf@proof@kw
6618     }{
6619       \spftype
6620     }
6621   }:
6622 }
6623 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6624
6625 \newcommand\stexpatchproof[3] [] {
6626   \str_set:Nx \l_tmpa_str{ #1 }
6627   \str_if_empty:NTF \l_tmpa_str {
6628     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6629     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6630   }{
6631     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6632     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6633   }
6634 }

```

\pstep
 \conclude
 \assumption
 \have
 \eqstep

```

6635
6636 \keys_define:nn { stex / spfsteps } {
6637   id          .str_set_x:N = \spfstepid,
6638   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,

```

```

6639 type .str_set_x:N = \spftype,
6640 title .tl_set:N = \spftitle,
6641 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6642 term .tl_set:N = \l__stex_sproof_spf_term_tl
6643 }
6644 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6645 \str_clear:N \spfstepid
6646 \clist_clear:N \l__stex_sproof_spf_for_clist
6647 \str_clear:N \spftype
6648 \tl_clear:N \l__stex_sproof_spf_method_tl
6649 \tl_clear:N \l__stex_sproof_spf_term_tl
6650 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6651 \keys_set:nn { stex / spfsteps }{ #1 }
6652 }
6653
6654 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6655 \NewDocumentCommand #1 {s O{} +m} {
6656 \__stex_sproof_maybe_comment_end:
6657
6658 \__stex_sproof_spfstep_args:n{##2}
6659 \stex_annotate:nnn{spfstep}{#2}{
6660 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6661 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6662 }
6663 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6664 #4
6665 }{
6666 \item[\IfBooleanTF ##1 {}{#3}]
6667 }
6668 \ignorespacesandpars ##3
6669 }
6670 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6671 \__stex_sproof_maybe_comment:
6672 }
6673 \stex_deactivate_macro:Nn #1 {sproof~environments}
6674 }
6675
6676 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6677 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6678 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6679
6680 \NewDocumentCommand \eqstep {s m}{
6681 \__stex_sproof_maybe_comment_end:
6682 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6683 $=$
6684 }{
6685 \item[$=$]
6686 }
6687 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6688 \__stex_sproof_maybe_comment:
6689 }
6690 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6691
6692 \NewDocumentCommand \yield {+m}{

```

```

6693 \stex_annotate:nnn{spfyield}{\}{ #1 }
6694 }
6695 \stex_deactivate_macro:Nn \yield {sproof~environments}
6696
6697 \NewDocumentEnvironment{spfblock}{\}{
6698   \item[]
6699   \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6700 }{
6701   \aftergroup\__stex_sproof_maybe_comment:
6702 }
6703 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6704

```

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

`\spfidea`

```

6705 \NewDocumentCommand\spfidea{0{} +m}{
6706   \__stex_sproof_spf_args:n{#1}
6707   \titleemph{
6708     \tl_if_empty:NTF \spftype {Proof~Idea}{
6709       \spftype
6710     }
6711   }~#2
6712   \sproofend
6713 }

```

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

```

6714 \newcommand\spfjust[1]{
6715   #1
6716 }
6717 \</package>

```

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

Chapter 34

STEX -Others Implementation

```
6718 <*package>
6719
6720 %%%%%%%%%% others.dtx %%%%%%%%%%
6721
6722 <@@=stex_others>
        Warnings and error messages
6723 % None

\MSC Math subject classifier

6724 \NewDocumentCommand \MSC {m} {
6725 % TODO
6726 }

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

6727 \@ifpackageloaded{tikzinput}{
6728 \RequirePackage{stex-tikzinput}
6729 }{}
6730
6731 \bool_if:NT \c_stex_persist_mode_bool {
6732 \let__stex_notation_restore_notation_old:nnnnn
6733 \__stex_notation_restore_notation:nnnnn
6734 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6735 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6736 \ExplSyntaxOn
6737 }
6738 \def__stex_notation_restore_notation:nnnnn{
6739 \ExplSyntaxOff
6740 \catcode'\sim10
6741 \__stex_notation_restore_notation_new:nnnnn
6742 }
6743 \input{\jobname.sms}
6744 \let__stex_notation_restore_notation:nnnnn
6745 \__stex_notation_restore_notation_old:nnnnn
6746 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```

6747 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6748 \l_tmpa_str
6749 \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6750 \c_stex_mathhub_main_manifest_prop
6751 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6752 }
6753 }
6754
6755 \stex_get_document_uri:
6756 </package>

```

Chapter 35

STEX -Metatheory Implementation

```
6757 <*package>
6758 <@@=stex_modules>
6759
6760 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6761
6762 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6763 \begingroup
6764 \stex_module_setup:nn{
6765   ns=\c_stex_metatheory_ns_str,
6766   meta=NONE
6767 }{Metatheory}
6768 \stex_reactivate_macro:N \symdecl
6769 \stex_reactivate_macro:N \notation
6770 \stex_reactivate_macro:N \symdef
6771 \ExplSyntaxOff
6772 \csname stex_suppress_html:n\endcsname{
6773   % is-a (a:A, a \in A, a is an A, etc.)
6774   \symdecl{isa}[args=ai]
6775   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6776   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6777   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6778
6779   % bind (\forall, \Pi, \lambda etc.)
6780   \symdecl{bind}[args=Bi,assoc=pre]
6781   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6782   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6783   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6784
6785   % implicit bind
6786   \symdecl{implicitbind}[args=Bi,assoc=pre]
6787   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6788     \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6789     \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6790
6791   % dummy variable
```



```

6792 \symdecl{dummyvar}
6793 \notation{dummyvar}[underscore]{\comp\_}
6794 \notation{dummyvar}[dot]{\comp\cdot}
6795 \notation{dummyvar}[dash]{\comp{\rm --}}
6796
6797 %fromto (function space, Hom-set, implication etc.)
6798 \symdecl{fromto}[args=ai]
6799 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6800 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6801
6802 % mapto (lambda etc.)
6803 \symdecl{mapto}[args=Bi]
6804 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6805 \notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6806 \notation{mapto}[lambdau]{\comp\lambda_#1 \comp.\; #2}{#1 \comp, #2}
6807
6808 % function/operator application
6809 \symdecl{apply}[args=ia]
6810 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6811 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6812
6813 % collection of propositions/booleans/truth values
6814 \symdecl{prop}[name=proposition]
6815 \notation{prop}[prop]{\comp{\rm prop}}
6816 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6817
6818 \symdecl{judgmentholds}[args=1]
6819 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6820
6821 % sequences
6822 \symdecl{seqtype}[args=1]
6823 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6824
6825 \symdecl{seqexpr}[args=a]
6826 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6827
6828 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
6829 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6830 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6831 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6832 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp
6833 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6834 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6835 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6836 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6837
6838 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
6839 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{#2}}
6840
6841 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6842 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6843 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#
6844
6845 % nat literals

```

```

6846 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6847
6848 % letin (''let'', local definitions, variable substitution)
6849 \symdecl{letin}[args=bii]
6850 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
6851 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6852 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6853
6854 % structures
6855 \symdecl*{module-type}[args=1]
6856 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6857 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6858 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\angle}{##1 \comp, ##2}
6859
6860 % objects
6861 \symdecl{object}
6862 \notation{object}{\comp{\mathtt{OBJECT}}}
6863
6864 }
6865
6866 % The following are abbreviations in the sTeX corpus that are left over from earlier
6867 % developments. They will eventually be phased out.
6868
6869 \ExplSyntaxOn
6870 \stex_add_to_current_module:n{
6871   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6872   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6873   \def\livar{\csname sequence-index\endcsname[li]}
6874   \def\uivar{\csname sequence-index\endcsname[ui]}
6875   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6876   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6877 }
6878 \__stex_modules_end_module:
6879 \endgroup
6880 \</package>

```

Chapter 36

Tikzinput Implementation

```
6881 <@@=tikzinput>
6882 <*package>
6883
6884 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6885
6886 \ProvidesExplPackage{tikzinput}{2022/05/24}{3.1.0}{tikzinput package}
6887 \RequirePackage{l3keys2e}
6888
6889 \keys_define:nn { tikzinput } {
6890   image .bool_set:N = \c_tikzinput_image_bool,
6891   image .default:n = false ,
6892   unknown .code:n = {}
6893 }
6894
6895 \ProcessKeysOptions { tikzinput }
6896
6897 \bool_if:NTF \c_tikzinput_image_bool {
6898   \RequirePackage{graphicx}
6899
6900   \providecommand\usetikzlibrary[]{}
6901   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6902 }{
6903   \RequirePackage{tikz}
6904   \RequirePackage{standalone}
6905
6906   \newcommand \tikzinput [2] [] {
6907     \setkeys{Gin}{#1}
6908     \ifx \Gin@ewidth \Gin@exclamation
6909       \ifx \Gin@eheight \Gin@exclamation
6910         \input { #2 }
6911       \else
6912         \resizebox{!}{ \Gin@eheight }{
6913           \input { #2 }
6914         }
6915       \fi
6916     \else
6917       \ifx \Gin@eheight \Gin@exclamation
6918         \resizebox{ \Gin@ewidth }{!}{
```

```

6919         \input { #2 }
6920     }
6921     \else
6922         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6923             \input { #2 }
6924         }
6925     \fi
6926 \fi
6927 }
6928 }
6929
6930 \newcommand \ctikzinput [2] [] {
6931     \begin{center}
6932         \tikzinput [#1] {#2}
6933     \end{center}
6934 }
6935
6936 \@ifpackageloaded{stex}{
6937     \RequirePackage{stex-tikzinput}
6938 }{}
6939
6940 </package>
6941 <*stex>
6942 \ProvidesExplPackage{stex-tikzinput}{2022/05/24}{3.1.0}{stex-tikzinput}
6943 \RequirePackage{stex}
6944 \RequirePackage{tikzinput}
6945
6946 \newcommand\mhtikzinput[2] [] {%
6947     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
6948     \stex_in_repository:nn\Gin@mhrepos{
6949         \tikzinput[#1]{\mhp\path{##1}{#2}}
6950     }
6951 }
6952 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6953
6954 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6955     \pgfkeys@spdef\pgf@temp{#1}
6956     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6957     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6958     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6959     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6960     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6961     \catcode'\@=11
6962     \catcode'\|=12
6963     \catcode'\$=3
6964     \pgfutil@InputIfFileExists{#2}{-}{-}
6965     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6966     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6967     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6968 }
6969
6970
6971 \newcommand\libusetikzlibrary[1]{

```

```

6972 \prop_if_exist:NF \l_stex_current_repository_prop {
6973   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6974 }
6975 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6976   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6977 }
6978 \seq_clear:N \l__tikzinput_libinput_files_seq
6979 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6980 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6981
6982 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6983   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6984   \IfFileExists{ \l_tmpa_str }{
6985     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6986   }{}
6987   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6988   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6989 }
6990
6991 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6992 \IfFileExists{ \l_tmpa_str }{
6993   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6994 }{}
6995
6996 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6997   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6998 }{
6999   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7000     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7001       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7002     }
7003   }{
7004     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7005   }
7006 }
7007 }
7008 </stex>

```

Chapter 37

document-structure.sty Implementation

```
7009 <*package>
7010 <@@=document_structure>
7011 \ProvidesExplPackage{document-structure}{2022/05/24}{3.1.0}{Modular Document Structure}
7012 \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).

```
7013
7014 \keys_define:nn{ document-structure }{
7015   class      .str_set_x:N = \c_document_structure_class_str,
7016   topsect    .str_set_x:N = \c_document_structure_topsect_str,
7017   unknown    .code:n      = {
7018     \PassOptionsToClass{\CurrentOption}{stex}
7019     \PassOptionsToClass{\CurrentOption}{tikzinput}
7020   }
7021   % showignores .bool_set:N = \c_document_structure_showignores_bool,
7022 }
7023 \ProcessKeysOptions{ document-structure }
7024 \str_if_empty:NT \c_document_structure_class_str {
7025   \str_set:Nn \c_document_structure_class_str {article}
7026 }
7027 \str_if_empty:NT \c_document_structure_topsect_str {
7028   \str_set:Nn \c_document_structure_topsect_str {section}
7029 }
```

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

```
7030 \RequirePackage{xspace}
7031 \RequirePackage{comment}
7032 \RequirePackage{stex}
7033 \AddToHook{begindocument}{
```

```

7034 \ltx@ifpackageloaded{babel}{
7035     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7036     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7037         \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7038     }
7039 }{}
7040 }

```

`\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.

```

7041 \int_new:N \l_document_structure_section_level_int
7042 \str_case:NnF \c_document_structure_topsect_str {
7043     {part}}{
7044         \int_set:Nn \l_document_structure_section_level_int {0}
7045     }
7046     {chapter}{
7047         \int_set:Nn \l_document_structure_section_level_int {1}
7048     }
7049 }{
7050     \str_case:NnF \c_document_structure_class_str {
7051         {book}{
7052             \int_set:Nn \l_document_structure_section_level_int {0}
7053         }
7054         {report}{
7055             \int_set:Nn \l_document_structure_section_level_int {0}
7056         }
7057     }{
7058         \int_set:Nn \l_document_structure_section_level_int {2}
7059     }
7060 }

```

37.2 Document Structure

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

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.⁹

EdN:9

```

7061 \def\current@section@level{document}%
7062 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
7063 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

7064 \cs_new_protected:Npn \skipfragment {

```

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

```

7065 \ifcase\l_document_structure_section_level_int
7066 \or\stepcounter{part}
7067 \or\stepcounter{chapter}
7068 \or\stepcounter{section}
7069 \or\stepcounter{subsection}
7070 \or\stepcounter{subsubsection}
7071 \or\stepcounter{paragraph}
7072 \or\stepcounter{subparagraph}
7073 \fi
7074 }

```

(End definition for `\skipfragment`. This function is documented on page 56.)

blindfragment

```

7075 \newcommand\at@begin@blindsfragment[1]{
7076 \newenvironment{blindfragment}
7077 {
7078 \int_incr:N\l_document_structure_section_level_int
7079 \at@begin@blindsfragment\l_document_structure_section_level_int
7080 }{}

```

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

```

7081 \newcommand\sfragment@nonum[2]{
7082 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7083 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7084 }

```

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

\sfragment@num convenience macro: `\sfragment@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `sfragment` environment and – if it is use it. But how to do that depends on whether the `rdfmata` package has been loaded. In the end we call `\sref@label@id` to enable crossreferencing.

```

7085 \newcommand\sfragment@num[2]{
7086 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7087 \@nameuse{#1}{#2}
7088 }{
7089 \cs_if_exist:NTF\rdfmata@sectioning{
7090 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7091 }{
7092 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7093 }
7094 }
7095 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7096 }

```

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

sfragment

```

7097 \keys_define:nn { document-structure / sfragment }{
7098 id .str_set_x:N = \l__document_structure_sfragment_id_str,
7099 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```



```

7100 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7101 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7102 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
7103 type          .tl_set:N     = \l__document_structure_sfragment_type_tl,
7104 short         .tl_set:N     = \l__document_structure_sfragment_short_tl,
7105 intro         .tl_set:N     = \l__document_structure_sfragment_intro_tl,
7106 imports       .tl_set:N     = \l__document_structure_sfragment_imports_tl,
7107 loadmodules   .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
7108 }
7109 \cs_new_protected:Nn \__document_structure_sfragment_args:n {
7110   \str_clear:N \l__document_structure_sfragment_id_str
7111   \str_clear:N \l__document_structure_sfragment_date_str
7112   \clist_clear:N \l__document_structure_sfragment_creators_clist
7113   \clist_clear:N \l__document_structure_sfragment_contributors_clist
7114   \tl_clear:N \l__document_structure_sfragment_srccite_tl
7115   \tl_clear:N \l__document_structure_sfragment_type_tl
7116   \tl_clear:N \l__document_structure_sfragment_short_tl
7117   \tl_clear:N \l__document_structure_sfragment_imports_tl
7118   \tl_clear:N \l__document_structure_sfragment_intro_tl
7119   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7120   \keys_set:nn { document-structure / sfragment } { #1 }
7121 }

```

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.

```

7122 \newif\if@mainmatter\@mainmattertrue
7123 \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.

```

7124 \keys_define:nn { document-structure / sectioning }{
7125   name      .str_set_x:N = \l__document_structure_sect_name_str  ,
7126   ref       .str_set_x:N = \l__document_structure_sect_ref_str   ,
7127   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
7128   clear     .default:n   = {true}                                ,
7129   num       .bool_set:N  = \l__document_structure_sect_num_bool   ,
7130   num       .default:n   = {true}
7131 }
7132 \cs_new_protected:Nn \__document_structure_sect_args:n {
7133   \str_clear:N \l__document_structure_sect_name_str
7134   \str_clear:N \l__document_structure_sect_ref_str
7135   \bool_set_false:N \l__document_structure_sect_clear_bool
7136   \bool_set_false:N \l__document_structure_sect_num_bool
7137   \keys_set:nn { document-structure / sectioning } { #1 }
7138 }
7139 \newcommand\omdoc@sectioning[3][]{
7140   \__document_structure_sect_args:n {#1 }
7141   \let\omdoc@sect@name\l__document_structure_sect_name_str
7142   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7143   \if@mainmatter% numbering not overridden by frontmatter, etc.
7144     \bool_if:NTF \l__document_structure_sect_num_bool {
7145       \sfragment@num{#2}{#3}
7146     }{

```

```

7147     \sfragment@nonum{#2}{#3}
7148   }
7149   \def\current@section@level{\omdoc@sect@name}
7150   \else
7151     \sfragment@nonum{#2}{#3}
7152   \fi
7153 }% 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.

```

7154 \newcommand\sfragment@redefine@addtocontents[1]{%
7155 %\edef\__document_structureimport{#1}%
7156 %\@for\@I:=\__document_structureimport\do{%
7157 %\edef\@path{\csname module@\@I @path\endcsname}%
7158 %\@ifundefined{tf@toc}\relax%
7159 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
7160 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7161 %\def\addcontentsline##1##2##3{%
7162 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7163 %\else% hyperref.sty not loaded
7164 %\def\addcontentsline##1##2##3{%
7165 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7166 %\fi
7167 }% 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.

```

7168 \newenvironment{sfragment}[2][ ]% keys, title
7169 {
7170   \__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.

```

7171   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7172
7173   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7174     \sfragment@redefine@addtocontents{
7175       %\@ifundefined{module@id}\used@modules%
7176       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7177     }
7178   }

```

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

```

7179
7180   \stex_document_title:n { #2 }
7181
7182   \int_incr:N\l__document_structure_section_level_int
7183   \ifcase\l__document_structure_section_level_int
7184     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7185     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7186     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
7187     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

7188 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7189 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
7190 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
7191 \fi
7192 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
7193 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7194   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7195 }
7196 }% for customization
7197 {}

```

and finally, we localize the sections

```

7198 \newcommand\omdoc@part@kw{Part}
7199 \newcommand\omdoc@chapter@kw{Chapter}
7200 \newcommand\omdoc@section@kw{Section}
7201 \newcommand\omdoc@subsection@kw{Subsection}
7202 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7203 \newcommand\omdoc@paragraph@kw{paragraph}
7204 \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`

```

7205 \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).

```

7206 \cs_if_exist:NTF\frontmatter{
7207   \let\__document_structure_orig_frontmatter\frontmatter
7208   \let\frontmatter\relax
7209 }{
7210   \tl_set:Nn\__document_structure_orig_frontmatter{
7211     \clearpage
7212     \@mainmatterfalse
7213     \pagenumbering{roman}
7214   }
7215 }
7216 \cs_if_exist:NTF\backmatter{
7217   \let\__document_structure_orig_backmatter\backmatter
7218   \let\backmatter\relax
7219 }{
7220   \tl_set:Nn\__document_structure_orig_backmatter{
7221     \clearpage
7222     \@mainmatterfalse
7223     \pagenumbering{roman}
7224   }

```

7225 }

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

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

```
7226 \newenvironment{frontmatter}{
7227   \_document_structure_orig_frontmatter
7228 }{
7229   \cs_if_exist:NTF\mainmatter{
7230     \mainmatter
7231   }{
7232     \clearpage
7233     \@mainmattertrue
7234     \pagenumbering{arabic}
7235   }
7236 }
```

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

```
7237 \newenvironment{backmatter}{
7238   \_document_structure_orig_backmatter
7239 }{
7240   \cs_if_exist:NTF\mainmatter{
7241     \mainmatter
7242   }{
7243     \clearpage
7244     \@mainmattertrue
7245     \pagenumbering{arabic}
7246   }
7247 }
```

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

```
7248 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7249 \def \c__document_structure_document_str{document}
7250 \newcommand\afterprematurestop{}
7251 \def\prematurestop@endsfragment{
7252   \unless\ifx\@currenvir\c__document_structure_document_str
7253     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7254       \expandafter\prematurestop@endsfragment
7255     }
7256   }
7257 \providecommand\prematurestop{
7258   \message{Stopping~sTeX~processing~prematurely}
7259   \prematurestop@endsfragment
7260   \afterprematurestop
7261   \end{document}
7262 }
```

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

37.4 Global Variables

\setSGvar set a global variable

```
7263 \RequirePackage{etoolbox}
7264 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
7265 \newrobustcmd\useSGvar[1]{%
7266   \@ifundefined{sTeX@Gvar@#1}
7267   {\PackageError{document-structure}
7268     {The sTeX Global variable #1 is undefined}
7269     {set it with \protect\setSGvar}}
7270   \@nameuse{sTeX@Gvar@#1}}
```

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

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

```
7271 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7272   \@ifundefined{sTeX@Gvar@#1}
7273   {\PackageError{document-structure}
7274     {The sTeX Global variable #1 is undefined}
7275     {set it with \protect\setSGvar}}
7276   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

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.

```
7277 \*cls)
7278 \@@=notesslides)
7279 \ProvidesExplClass{notesslides}{2022/05/24}{3.1.0}{notesslides Class}
7280 \RequirePackage{13keys2e}
7281
7282 \keys_define:nn{notesslides / cls}{
7283   class .str_set_x:N = \c__notesslides_class_str,
7284   notes .bool_set:N = \c__notesslides_notes_bool ,
7285   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7286   docopt .str_set_x:N = \c__notesslides_docopt_str,
7287   unknown .code:n = {
7288     \PassOptionsToPackage{\CurrentOption}{document-structure}
7289     \PassOptionsToClass{\CurrentOption}{beamer}
7290     \PassOptionsToPackage{\CurrentOption}{notesslides}
7291     \PassOptionsToPackage{\CurrentOption}{stex}
7292   }
7293 }
7294 \ProcessKeysOptions{ notesslides / cls }
7295
7296 \str_if_empty:NF \c__notesslides_class_str {
7297   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7298 }
7299
7300 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7301   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7302 }
7303 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7304   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7305 }
7306
7307 \RequirePackage{stex}
```

```

7308 \stex_html_backend:T {
7309   \bool_set_true:N\c__notesslides_notes_bool
7310 }
7311
7312 \bool_if:NTF \c__notesslides_notes_bool {
7313   \PassOptionsToPackage{notes=true}{notesslides}
7314   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7315 }{
7316   \PassOptionsToPackage{notes=false}{notesslides}
7317   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7318 }
7319 </cls>

```

now we do the same for the notesslides package.

```

7320 <*package>
7321 \ProvidesExplPackage{notesslides}{2022/05/24}{3.1.0}{notesslides Package}
7322 \RequirePackage{l3keys2e}
7323
7324 \keys_define:nn{notesslides / pkg}{
7325   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7326   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7327   notes            .bool_set:N = \c__notesslides_notes_bool ,
7328   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7329   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7330   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7331   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7332   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
7333   unknown          .code:n      = {
7334     \PassOptionsToClass{\CurrentOption}{stex}
7335     \PassOptionsToClass{\CurrentOption}{tikzinput}
7336   }
7337 }
7338 \ProcessKeysOptions{ notesslides / pkg }
7339
7340 \RequirePackage{stex}
7341 \stex_html_backend:T {
7342   \bool_set_true:N\c__notesslides_notes_bool
7343 }
7344
7345 \newif\ifnotes
7346 \bool_if:NTF \c__notesslides_notes_bool {
7347   \notesttrue
7348 }{
7349   \notesfalse
7350 }
7351

```

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

```

7352 \str_if_empty:NTF \c__notesslides_topsect_str {
7353   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7354 }{
7355   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7356 }
7357 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
7358 </package>
```

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

```
7359 <*cls>
7360 \bool_if:NTF \c__notesslides_notes_bool {
7361   \str_if_empty:NT \c__notesslides_class_str {
7362     \str_set:Nn \c__notesslides_class_str {article}
7363   }
7364   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
7365     {\c__notesslides_class_str}
7366 }{
7367   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7368   \newcounter{Item}
7369   \newcounter{paragraph}
7370   \newcounter{subparagraph}
7371   \newcounter{Hfootnote}
7372 }
7373 \RequirePackage{document-structure}
```

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

```
7374 \RequirePackage{notesslides}
7375 </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.

```
7376 <*package>
7377 \bool_if:NT \c__notesslides_notes_bool {
7378   \RequirePackage{a4wide}
7379   \RequirePackage{marginnote}
7380   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7381   \RequirePackage{mdframed}
7382   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7383   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7384 }
7385 \RequirePackage{stex-tikzinput}
7386 \RequirePackage{comment}
7387 \RequirePackage{url}
7388 \RequirePackage{graphicx}
7389 \RequirePackage{pgf}
7390 \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.

```
7391 \bool_if:NT \c__notesslides_notes_bool {
7392   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7393 }
```



```

7394 \NewDocumentCommand \libusetheme {0{} m} {
7395   \libusepackage[#1]{beamertheme#2}
7396 }
7397

```

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

```

7398 \newcounter{slide}
7399 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7400 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

note The `note` environment is used to leave out text in the `slides` mode. It does not have a counterpart in OMDoc. So for course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

```

7401 \bool_if:NTF \c__notesslides_notes_bool {
7402   \renewenvironment{note}{\ignorespaces}{}
7403 }{
7404   \excludecomment{note}
7405 }

```

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.

```

7406 \bool_if:NT \c__notesslides_notes_bool {
7407   \newlength{\slideframewidth}
7408   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

7409 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7410   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7411     \bool_set_true:N #1
7412   }{
7413     \bool_set_false:N #1
7414   }
7415 }
7416 \keys_define:nn{notesslides / frame}{
7417   label .str_set_x:N = \l__notesslides_frame_label_str,
7418   allowframebreaks .code:n = {
7419     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7420   },
7421   allowdisplaybreaks .code:n = {
7422     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7423   },
7424   fragile .code:n = {
7425     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7426   },
7427   shrink .code:n = {
7428     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7429   },
7430   squeeze .code:n = {
7431     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7432   },
7433   t .code:n = {

```

```

7434     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7435   },
7436   unknown    .code:n      = {}
7437 }
7438 \cs_new_protected:Nn \__notesslides_frame_args:n {
7439   \str_clear:N \l__notesslides_frame_label_str
7440   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7441   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7442   \bool_set_true:N \l__notesslides_frame_fragile_bool
7443   \bool_set_true:N \l__notesslides_frame_shrink_bool
7444   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7445   \bool_set_true:N \l__notesslides_frame_t_bool
7446   \keys_set:nn { notesslides / frame }{ #1 }
7447 }

```

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

```

7448 \renewenvironment{frame}[1][]{
7449   \__notesslides_frame_args:n{#1}
7450   \sffamily
7451   \stepcounter{slide}
7452   \def\@currentlabel{\theslide}
7453   \str_if_empty:NF \l__notesslides_frame_label_str {
7454     \label{\l__notesslides_frame_label_str}
7455   }

```

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

```

7456   \def\itemize@level{outer}
7457   \def\itemize@outer{outer}
7458   \def\itemize@inner{inner}
7459   \renewcommand\newpage{\addtocounter{framenum}{1}}
7460   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7461   \renewenvironment{itemize}{
7462     \ifx\itemize@level\itemize@outer
7463       \def\itemize@label{$\rhd$}
7464     \fi
7465     \ifx\itemize@level\itemize@inner
7466       \def\itemize@label{$\scriptstyle\rhd$}
7467     \fi
7468     \begin{list}
7469       {\itemize@label}
7470       {\setlength{\labelsep}{.3em}
7471        \setlength{\labelwidth}{.5em}
7472        \setlength{\leftmargin}{1.5em}
7473       }
7474     \edef\itemize@level{\itemize@inner}
7475   }{
7476     \end{list}
7477   }

```

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

```

7478   \stex_html_backend:TF {
7479     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7480     \mdf@patchamsthm
7481   }{
7482     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7483     }
7484   }{
7485     \stex_html_backend:TF {
7486       \miko@slidelabel\egroup\end{stex_annotate_env}
7487     }\medskip\miko@slidelabel\end{mdframed}}
7488   }

```

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

\frametitle

```

7489   \renewcommand{\frametitle}[1]{
7490     \stex_document_title:n { #1 }
7491     {\Large\bf\sf\color{blue}{#1}}\medskip
7492   }
7493 }

```

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

EdN:10

\pause 10

```

7494 \bool_if:NT \c__notesslides_notes_bool {
7495   \newcommand\pause{}
7496 }

```

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

nparagraph

```

7497 \bool_if:NTF \c__notesslides_notes_bool {
7498   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7499 }{
7500   \excludecomment{nparagraph}
7501 }

```

nfragment

```

7502 \bool_if:NTF \c__notesslides_notes_bool {
7503   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
7504 }{
7505   \excludecomment{nfragment}
7506 }

```

ndefinition

```

7507 \bool_if:NTF \c__notesslides_notes_bool {
7508   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7509 }{
7510   \excludecomment{ndefinition}
7511 }

```

nassertion

```

7512 \bool_if:NTF \c__notesslides_notes_bool {
7513   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7514 }{
7515   \excludecomment{nassertion}
7516 }

```

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

nsproof

```
7517 \bool_if:NTF \c__notesslides_notes_bool {
7518   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7519 }{
7520   \excludecomment{nproof}
7521 }
```

nexample

```
7522 \bool_if:NTF \c__notesslides_notes_bool {
7523   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7524 }{
7525   \excludecomment{nexample}
7526 }
```

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

```
7527 \def\inputref@preskip{\smallskip}
7528 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
7529 \let\orig@inputref\inputref
7530 \def\inputref{\@ifstar\ninputref\orig@inputref}
7531 \newcommand\ninputref[2] []{
7532   \bool_if:NT \c__notesslides_notes_bool {
7533     \orig@inputref[#1]{#2}
7534   }
7535 }
```

(End definition for \inputref*. This function is documented on page 59.)

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>}.

```
7536 \newlength{\slidelogoheight}
7537
7538 \RequirePackage{graphicx}
7539
7540 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7541 \providecommand\mhgraphics[2] []{
7542   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7543   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7544 }
7545
7546 \bool_if:NTF \c__notesslides_notes_bool {
7547   \setlength{\slidelogoheight}{.4cm}
7548 }{
7549   \setlength{\slidelogoheight}{.25cm}
7550 }
```

```

7551 \ifcsname slidelogo\endcsname\else
7552 \newsavebox{\slidelogo}
7553 \sbox{\slidelogo}{\sTeX}
7554 \fi
7555 \newrobustcmd{\setslidelogo}[2][]{
7556 \tl_if_empty:nTF{#1}{
7557 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7558 }{
7559 \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7560 }
7561 }

```

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

\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.

```

7562 \bool_if:NT \c__notesslides_notes_bool {
7563 \def\author{\@dblarg\@ns@author}
7564 \long\def\@ns@author[#1]#2{%
7565 \def\c__notesslides_shortauthor{#1}%
7566 \def\@author{#2}
7567 }
7568 }

```

(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.

```

7569 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

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

\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.

```

7570 \def\copyrightnotice{%
7571 \footnotesize\copyright : \hspace{.3ex}%
7572 \ifcsname source\endcsname\source\else%
7573 \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7574 \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7575 ?source/author?\fi%
7576 \fi}
7577 \newsavebox{\cclogo}
7578 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7579 \newif\ifcchref\cchreffalse
7580 \AtBeginDocument{
7581 \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7582 }
7583 \def\licensing{
7584 \ifcchref
7585 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7586 \else
7587 {\usebox{\cclogo}}

```

```

7588 \fi
7589 }
7590 \newrobustcmd{\setlicensing}[2][]{
7591   \def\@url{#1}
7592   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7593   \ifx\@url\@empty
7594     \def\licensing{\usebox{\cclogo}}
7595   \else
7596     \def\licensing{
7597       \ifcchref
7598         \href{#1}{\usebox{\cclogo}}
7599       \else
7600         {\usebox{\cclogo}}
7601       \fi
7602     }
7603   \fi
7604 }

```

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

EdN:11

```

\slidelabel Now, we set up the slide label for the article mode.11
7605 \newrobustcmd\miko@slidelabel{
7606   \vbox to \slidelogoheight{
7607     \vss\hbox to \slidewidth
7608       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7609   }
7610 }

```

(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.

```

7611 \def\Gin@mhrepos{}
7612 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7613 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7614 \newrobustcmd\frameimage[2][]{
7615   \stepcounter{slide}
7616   \bool_if:NT \c__notesslides_frameimages_bool {
7617     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7618     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7619     \begin{center}
7620       \bool_if:NTF \c__notesslides_fiboxed_bool {
7621         \fbox{
7622           \ifx\Gin@ewidth\@empty
7623             \ifx\Gin@mhrepos\@empty
7624               \mhgraphics[width=\slidewidth,#1]{#2}
7625             \else
7626               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7627             \fi
7628           \else% Gin@ewidth empty

```

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

```

7629         \ifx\Gin@mhrepos\@empty
7630         \mhgraphics[#1]{#2}
7631     \else
7632         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7633     \fi
7634 \fi% Gin@ewidth empty
7635 }
7636 }{
7637     \ifx\Gin@ewidth\@empty
7638     \ifx\Gin@mhrepos\@empty
7639         \mhgraphics[width=\slidewidth,#1]{#2}
7640     \else
7641         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7642     \fi
7643     \ifx\Gin@mhrepos\@empty
7644         \mhgraphics[#1]{#2}
7645     \else
7646         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7647     \fi
7648     \fi% Gin@ewidth empty
7649 }
7650 \end{center}
7651 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7652 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7653 }
7654 } % ifmks@sty@frameimages

```

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

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`.

```

7655 \stex_html_backend:F {
7656     \bool_if:NT \c__notesslides_sectocframes_bool {
7657         \str_if_eq:VnTF \__notesslidestopsect{part}{
7658             \newcounter{chapter}\counterwithin*{section}{chapter}
7659         }{
7660             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7661                 \newcounter{chapter}\counterwithin*{section}{chapter}
7662             }
7663         }
7664     }
7665 }

```

`\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
7666 \def\part@prefix{}
7667 \@ifpackageloaded{document-structure}{
7668     \str_case:VnF \__notesslidestopsect {

```

```

7669 {part}{
7670   \int_set:Nn \l_document_structure_section_level_int {0}
7671   \def\thesection{\arabic{chapter}.\arabic{section}}
7672   \def\part@prefix{\arabic{chapter}.}
7673 }
7674 {chapter}{
7675   \int_set:Nn \l_document_structure_section_level_int {1}
7676   \def\thesection{\arabic{chapter}.\arabic{section}}
7677   \def\part@prefix{\arabic{chapter}.}
7678 }
7679 }{
7680   \int_set:Nn \l_document_structure_section_level_int {2}
7681   \def\part@prefix{}
7682 }
7683 }
7684
7685 \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`

```

7686 \renewenvironment{sfragment}[2][]{
7687   \__document_structure_sfragment_args:n { #1 }
7688   \int_incr:N \l_document_structure_section_level_int
7689   \bool_if:NT \c__notesslides_sectocframes_bool {
7690     \stepcounter{slide}
7691     \begin{frame}[noframenumbering]
7692     \vfill\Large\centering
7693     \red{
7694       \ifcase\l_document_structure_section_level_int\or
7695         \stepcounter{part}
7696         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7697         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7698         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7699         \def\currentsectionlevel{\omdoc@part@kw}
7700       \or
7701         \stepcounter{chapter}
7702         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7703         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7704         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7705         \def\currentsectionlevel{\omdoc@chapter@kw}
7706       \or
7707         \stepcounter{section}
7708         \def\__notesslideslabel{\part@prefix\arabic{section}}
7709         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7710         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7711         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7712         \def\currentsectionlevel{\omdoc@section@kw}
7713       \or
7714         \stepcounter{subsection}
7715         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7716         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```



```

7717         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7718         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7719         \def\currentsectionlevel{\omdoc@subsection@kw}
7720     \or
7721         \stepcounter{subsubsection}
7722         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7723         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7724         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7725         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7726         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7727     \or
7728         \stepcounter{paragraph}
7729         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7730         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7731         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7732         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7733         \def\currentsectionlevel{\omdoc@paragraph@kw}
7734     \else
7735         \def\__notesslideslabel{}
7736         \def\currentsectionlevel{\omdoc@paragraph@kw}
7737     \fi% end ifcase
7738     \__notesslideslabel\quad #2%
7739 }%
7740 \vfill%
7741 \end{frame}%
7742 }
7743 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7744     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7745 }
7746 }{}
7747 }

```

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

```

7748 \def\inserttheorembodyfont{\normalfont}
7749 %\bool_if:NF \c__notesslides_notes_bool {
7750 % \defbeamertemplate{theorem begin}{miko}
7751 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7752 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7753 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7754 % \defbeamertemplate{theorem end}{miko}{\fi}

```

and we set it as the default one.

```

7755 % \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.

```

7756 % \expandafter\def\csname Parent2\endcsname{}
7757 %}
7758
7759 \AddToHook{begindocument}{% this does not work for some reason
7760     \setbeamertemplate{theorems}[ams style]
7761 }
7762 \bool_if:NT \c__notesslides_notes_bool {
7763     \renewenvironment{columns}[1][\fi]{}{}

```

```

7764     \par\noindent%
7765     \begin{minipage}%
7766     \slidewidth\centering\leavevmode%
7767   }{%
7768     \end{minipage}\par\noindent%
7769   }%
7770   \newsavebox\columnbox%
7771   \renewenvironment<>{column}[2][]{%
7772     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7773   }{%
7774     \end{minipage}\end{lrbox}\usebox\columnbox%
7775   }%
7776 }

7777 \bool_if:NTF \c__notesslides_noproblems_bool {
7778   \newenvironment{problems}{}{}
7779 }{
7780   \excludacomment{problems}
7781 }

```

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.

```

7782 \gdef\printexcursions{}
7783 \newcommand\excursionref[2]{% label, text
7784   \bool_if:NT \c__notesslides_notes_bool {
7785     \begin{sparagraph}[title=Excursion]
7786       #2 \sref[fallback=the appendix]{#1}.
7787     \end{sparagraph}
7788   }
7789 }
7790 \newcommand\activate@excursion[2][{}{
7791   \gappto\printexcursions{\inputref{#1}{#2}}
7792 }
7793 \newcommand\excursion[4][{}{ repos, label, path, text
7794   \bool_if:NT \c__notesslides_notes_bool {
7795     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7796   }
7797 }

```

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

\excursiongroup

```

7798 \keys_define:nn{notesslides / excursiongroup }{
7799   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7800   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7801   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7802 }
7803 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7804   \tl_clear:N \l__notesslides_excursion_intro_tl
7805   \str_clear:N \l__notesslides_excursion_id_str

```

```

7806 \str_clear:N \l__notesslides_excursion_mhrepos_str
7807 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7808 }
7809 \newcommand\excursionsgroup[1][ ]{
7810 \__notesslides_excursion_args:n{ #1 }
7811 \ifdefempty\printexcursions{ }% only if there are excursions
7812 {\begin{note}
7813 \begin{sfragment}[#1]{Excursions}%
7814 \ifdefempty\l__notesslides_excursion_intro_tl}{
7815 \inputref[\l__notesslides_excursion_mhrepos_str]{
7816 \l__notesslides_excursion_intro_tl
7817 }
7818 }
7819 \printexcursions%
7820 \end{sfragment}
7821 \end{note}}
7822 }
7823 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7824 \end{package}

```

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

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.

```
7825 <*package>
7826 <@@=problems>
7827 \ProvidesExplPackage{problem}{2022/05/24}{3.1.0}{Semantic Markup for Problems}
7828 \RequirePackage{13keys2e}
7829 \RequirePackage{amssymb}% for \Box
7830
7831 \keys_define:nn { problem / pkg }{
7832   notes      .default:n    = { true },
7833   notes      .bool_set:N   = \c__problems_notes_bool,
7834   gnotes     .default:n    = { true },
7835   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7836   hints      .default:n    = { true },
7837   hints      .bool_set:N   = \c__problems_hints_bool,
7838   solutions  .default:n    = { true },
7839   solutions  .bool_set:N   = \c__problems_solutions_bool,
7840   pts        .default:n    = { true },
7841   pts        .bool_set:N   = \c__problems_pts_bool,
7842   min        .default:n    = { true },
7843   min        .bool_set:N   = \c__problems_min_bool,
7844   boxed      .default:n    = { true },
7845   boxed      .bool_set:N   = \c__problems_boxed_bool,
7846   unknown    .code:n       = {
7847     \PassOptionsToPackage{\CurrentOption}{stex}
7848   }
7849 }
7850 \newif\ifsolutions
7851
7852 \ProcessKeysOptions{ problem / pkg }
7853 \bool_if:NTF \c__problems_solutions_bool {
7854   \solutionstrue
7855 }{
7856   \solutionsfalse
```

```

7857 }
7858 \RequirePackage{stex}

```

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

```

7859 \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.

```

7860 \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.

```

7861 \def\prob@problem@kw{Problem}
7862 \def\prob@solution@kw{Solution}
7863 \def\prob@hint@kw{Hint}
7864 \def\prob@note@kw{Note}
7865 \def\prob@gnote@kw{Grading}
7866 \def\prob@pt@kw{pt}
7867 \def\prob@min@kw{min}
7868 \def\prob@correct@kw{Correct}
7869 \def\prob@wrong@kw{Wrong}

```

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

For the other languages, we set up triggers

```

7870 \AddToHook{begindocument}{
7871   \ltx@ifpackageloaded{babel}{
7872     \makeatletter
7873     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7874     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7875       \input{problem-ngerman.ldf}
7876     }
7877     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
7878       \input{problem-finnish.ldf}
7879     }
7880     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
7881       \input{problem-french.ldf}
7882     }
7883     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
7884       \input{problem-russian.ldf}
7885     }
7886     \makeatother
7887   }{}
7888 }

```

39.2 Problems and Solutions

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

```

7889 \keys_define:nn{ problem / problem }{
7890   id      .str_set_x:N = \l__problems_prob_id_str,
7891   pts     .tl_set:N   = \l__problems_prob_pts_tl,
7892   min     .tl_set:N   = \l__problems_prob_min_tl,

```

```

7893 title .tl_set:N = \l__problems_prob_title_tl,
7894 type .tl_set:N = \l__problems_prob_type_tl,
7895 imports .tl_set:N = \l__problems_prob_imports_tl,
7896 name .str_set_x:N = \l__problems_prob_name_str,
7897 refnum .int_set:N = \l__problems_prob_refnum_int
7898 }
7899 \cs_new_protected:Nn \l__problems_prob_args:n {
7900 \str_clear:N \l__problems_prob_id_str
7901 \str_clear:N \l__problems_prob_name_str
7902 \tl_clear:N \l__problems_prob_pts_tl
7903 \tl_clear:N \l__problems_prob_min_tl
7904 \tl_clear:N \l__problems_prob_title_tl
7905 \tl_clear:N \l__problems_prob_type_tl
7906 \tl_clear:N \l__problems_prob_imports_tl
7907 \int_zero_new:N \l__problems_prob_refnum_int
7908 \keys_set:nn { problem / problem }{ #1 }
7909 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7910 \let\l__problems_prob_refnum_int\undefined
7911 }
7912 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7913 \newcounter{problem}[section]
7914 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
7915 \def\theplainsproblem{\arabic{problem}}
7916 \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.

```

7917 \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

```

7918 \newcommand\prob@number{
7919 \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7920 \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7921 }{
7922 \int_if_exist:NTF \l__problems_prob_refnum_int {
7923 \prob@label{\int_use:N \l__problems_prob_refnum_int }
7924 }{
7925 \prob@label\theplainsproblem
7926 }
7927 }
7928 }
7929 \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.

```

7930 \newcommand\prob@title[3]{%
7931   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7932     #2 \l__problems_inclprob_title_tl #3
7933   }{
7934     \tl_if_empty:NTF \l__problems_prob_title_tl {
7935       #1
7936     }{
7937       #2 \l__problems_prob_title_tl #3
7938     }
7939   }
7940 }

```

(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.

```

7941 \def\prob@heading{
7942   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7943   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
7944 }

```

(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

```

7945 \newenvironment{sproblem}[1][]{
7946   \__problems_prob_args:n{#1}%\sref@target%
7947   \@in@omtexttrue% we are in a statement (for inline definitions)
7948   \refstepcounter{sproblem}\record@problem
7949   \def\current@section@level{\prob@problem@kw}
7950
7951   \str_if_empty:NT \l__problems_prob_name_str {
7952     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7953     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7954     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7955   }
7956
7957   \stex_if_do_html:T{
7958     \tl_if_empty:NF \l__problems_prob_title_tl {
7959       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7960     }
7961   }
7962
7963   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7964
7965   \stex_reactivate_macro:N \STEXexport
7966   \stex_reactivate_macro:N \importmodule
7967   \stex_reactivate_macro:N \symdecl
7968   \stex_reactivate_macro:N \notation
7969   \stex_reactivate_macro:N \symdef

```

```

7970
7971 \stex_if_do_html:T{
7972   \begin{stex_annotate_env} {problem} {
7973     \l_stex_module_ns_str ? \l_stex_module_name_str
7974   }
7975
7976   \stex_annotate_invisible:nnn{header}{} {
7977     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
7978     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7979     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7980       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7981     }
7982   }
7983 }
7984
7985 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7986
7987
7988 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7989   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7990 }{
7991   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7992 }
7993 \str_if_exist:NTF \l__problems_inclprob_id_str {
7994   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7995 }{
7996   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7997 }
7998
7999
8000 \stex_if_smsmode:F {
8001   \clist_set:No \l_tmpa_clist \sproblemtype
8002   \tl_clear:N \l_tmpa_tl
8003   \clist_map_inline:Nn \l_tmpa_clist {
8004     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8005       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8006     }
8007   }
8008   \tl_if_empty:NTF \l_tmpa_tl {
8009     \__problems_sproblem_start:
8010   }{
8011     \l_tmpa_tl
8012   }
8013 }
8014 \stex_ref_new_doc_target:n \sproblemid
8015 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8016 }{
8017   \__stex_modules_end_module:
8018   \stex_if_smsmode:F{
8019     \clist_set:No \l_tmpa_clist \sproblemtype
8020     \tl_clear:N \l_tmpa_tl
8021     \clist_map_inline:Nn \l_tmpa_clist {
8022       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8023         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}

```



```

8024     }
8025   }
8026   \tl_if_empty:NTF \l_tmpa_tl {
8027     \__problems_sproblem_end:
8028   }{
8029     \l_tmpa_tl
8030   }
8031 }
8032 \stex_if_do_html:T{
8033   \end{stex_annotate_env}
8034 }
8035
8036 \smallskip
8037 }
8038
8039 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8040
8041
8042
8043 \cs_new_protected:Nn \__problems_sproblem_start: {
8044   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
8045 }
8046 \cs_new_protected:Nn \__problems_sproblem_end: { \par\smallskip}
8047
8048 \newcommand\stexpatchproblem[3][] {
8049   \str_set:Nx \l_tmpa_str{ #1 }
8050   \str_if_empty:NTF \l_tmpa_str {
8051     \tl_set:Nn \__problems_sproblem_start: { #2 }
8052     \tl_set:Nn \__problems_sproblem_end: { #3 }
8053   }{
8054     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8055     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8056   }
8057 }
8058
8059
8060 \bool_if:NT \c__problems_boxed_bool {
8061   \surroundwithhmdframed{problem}
8062 }

```

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

```

8063 \def\record@problem{
8064   \protected@write\auxout{}
8065   {
8066     \string\@problem{\prob@number}
8067   {
8068     \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8069       \l__problems_inclprob_pts_tl
8070     }{
8071       \l__problems_prob_pts_tl
8072     }
8073   }%
8074   {
8075     \tl_if_exist:NTF \l__problems_inclprob_min_tl {

```

```

8076         \l__problems_inclprob_min_tl
8077     }{
8078         \l__problems_prob_min_tl
8079     }
8080 }
8081 }
8082 }

```

(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).

```

8083 \def\@problem#1#2#3{

```

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

solution The solution environment is similar to the problem environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```

8084 \keys_define:nn { problem / solution }{
8085   id          .str_set_x:N = \l__problems_solution_id_str ,
8086   for         .str_set_x:N = \l__problems_solution_for_str ,
8087   type       .str_set_x:N = \l__problems_solution_type_str ,
8088   title      .tl_set:N     = \l__problems_solution_title_tl
8089 }
8090 \cs_new_protected:Nn \__problems_solution_args:n {
8091   \str_clear:N \l__problems_solution_id_str
8092   \str_clear:N \l__problems_solution_type_str
8093   \str_clear:N \l__problems_solution_for_str
8094   \tl_clear:N \l__problems_solution_title_tl
8095   \keys_set:nn { problem / solution }{ #1 }
8096 }

```

\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.

```

8097 \box_new:N \l__problems_solution_box
8098 \newenvironment{solution}[1][{}{
8099   \__problems_solution_args:n{#1}
8100   \stex_html_backend:TF{
8101     \stex_if_do_html:T{
8102       \begin{stex_annotate_env}{solution}{}
8103       \str_if_empty:NF \l__problems_solution_type_str {
8104         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
8105       }
8106       \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8107     }
8108   }{
8109     \setbox\l__problems_solution_box\vbox\bgroup
8110     \par\smallskip\hrule\smallskip
8111     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
8112   }
8113 }{
8114   \stex_html_backend:TF{
8115     \stex_if_do_html:T{
8116       \end{stex_annotate_env}

```

```

8117     }
8118   }{
8119     \smallskip\hrule
8120     \egroup
8121     \bool_if:NT \c__problems_solutions_bool {
8122       \box\l__problems_solution_box
8123     }
8124   }
8125 }
8126
8127 \newcommand\startsolutions{
8128   \bool_set_true:N \c__problems_solutions_bool
8129   \solutionstrue
8130   % \specialcomment{solution}{\@startsolution}{
8131   %   \bool_if:NF \c__problems_boxed_bool {
8132   %     \hrule\medskip
8133   %   }
8134   %   \end{small}%
8135   % }
8136   % \bool_if:NT \c__problems_boxed_bool {
8137   %   \surroundwithmdframed{solution}
8138   % }
8139 }

```

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

\stopsolutions

```

8140 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

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

exnote

```

8141 \bool_if:NTF \c__problems_notes_bool {
8142   \newenvironment{exnote}[1][]{
8143     \par\smallskip\hrule\smallskip
8144     \noindent\textbf{\prob@note@kw :~ }\small
8145   }{
8146     \smallskip\hrule
8147   }
8148 }{
8149   \excludacomment{exnote}
8150 }

```

hint

```

8151 \bool_if:NTF \c__problems_notes_bool {
8152   \newenvironment{hint}[1][]{
8153     \par\smallskip\hrule\smallskip
8154     \noindent\textbf{\prob@hint@kw :~ }\small
8155   }{
8156     \smallskip\hrule
8157   }
8158 }{
8159   \newenvironment{exhint}[1][]{
8160     \par\smallskip\hrule\smallskip
8161     \noindent\textbf{\prob@hint@kw :~ }\small

```

```

8161 }{
8162     \smallskip\hrule
8163 }
8164 }{
8165     \excludecomment{hint}
8166     \excludecomment{exhint}
8167 }

```

gnote

```

8168 \bool_if:NTF \c__problems_notes_bool {
8169     \newenvironment{gnote}[1][]{
8170         \par\smallskip\hrule\smallskip
8171         \noindent\textbf{\prob@gnote@kw :~ }\small
8172     }{
8173         \smallskip\hrule
8174     }
8175 }{
8176     \excludecomment{gnote}
8177 }

```

39.3 Marup for Added Value Services

39.4 Multiple Choice Blocks

EdN:12

mcb

12

```

8178 \newenvironment{mcb}{
8179     \begin{enumerate}
8180 }{
8181     \end{enumerate}
8182 }

```

we define the keys for the mcb macro

```

8183 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8184     \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8185         \bool_set_true:N #1
8186     }{
8187         \bool_set_false:N #1
8188     }
8189 }
8190 \keys_define:nn { problem / mcb }{
8191     id .str_set:x:N = \l__problems_mcc_id_str ,
8192     feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
8193     T .default:n = { false } ,
8194     T .bool_set:N = \l__problems_mcc_t_bool ,
8195     F .default:n = { false } ,
8196     F .bool_set:N = \l__problems_mcc_f_bool ,
8197     Ttext .tl_set:N = \l__problems_mcc_Ttext_tl ,
8198     Ftext .tl_set:N = \l__problems_mcc_Ftext_tl
8199 }
8200 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

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

```

8201 \str_clear:N \l__problems_mcc_id_str
8202 \tl_clear:N \l__problems_mcc_feedback_tl
8203 \bool_set_false:N \l__problems_mcc_t_bool
8204 \bool_set_false:N \l__problems_mcc_f_bool
8205 \tl_clear:N \l__problems_mcc_Ttext_tl
8206 \tl_clear:N \l__problems_mcc_Ftext_tl
8207 \str_clear:N \l__problems_mcc_id_str
8208 \keys_set:nn { problem / mcc }{ #1 }
8209 }

```

\mcc

```

8210 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8211 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8212 \newcommand\mcc[2][] {
8213   \l__problems_mcc_args:n{ #1 }
8214   \item[$\Box$] #2
8215   \bool_if:NT \c__problems_solutions_bool{
8216     \\\
8217     \bool_if:NT \l__problems_mcc_t_bool {
8218       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8219     }
8220     \bool_if:NT \l__problems_mcc_f_bool {
8221       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8222     }
8223     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8224       \emph{\l__problems_mcc_feedback_tl}
8225     }
8226   }
8227 } %solutions

```

(End definition for \mcc. This function is documented on page 64.)

39.5 Filling in Concrete Solutions

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

```

8228 \newcommand\fillinsol[1]{\quad%
8229   \ifsolutions\textcolor{red}{\#!}\else%
8230   \fbox{\phantom{\huge{\#!}}}%
8231   \fi}

```

(End definition for \includeproblem. This function is documented on page 66.)

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.

```

8232
8233 \keys_define:nn{ problem / inclproblem }{
8234   id      .str_set_x:N = \l__problems_inclprob_id_str,
8235   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8236   min     .tl_set:N    = \l__problems_inclprob_min_tl,

```

```

8237 title .tl_set:N = \l__problems_inclprob_title_tl,
8238 refnum .int_set:N = \l__problems_inclprob_refnum_int,
8239 type .tl_set:N = \l__problems_inclprob_type_tl,
8240 mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8241 }
8242 \cs_new_protected:Nn \l__problems_inclprob_args:n {
8243 \str_clear:N \l__problems_prob_id_str
8244 \tl_clear:N \l__problems_inclprob_pts_tl
8245 \tl_clear:N \l__problems_inclprob_min_tl
8246 \tl_clear:N \l__problems_inclprob_title_tl
8247 \tl_clear:N \l__problems_inclprob_type_tl
8248 \int_zero_new:N \l__problems_inclprob_refnum_int
8249 \str_clear:N \l__problems_inclprob_mhrepos_str
8250 \keys_set:nn { problem / inclproblem }{ #1 }
8251 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8252 \let\l__problems_inclprob_pts_tl\undefined
8253 }
8254 \tl_if_empty:NT \l__problems_inclprob_min_tl {
8255 \let\l__problems_inclprob_min_tl\undefined
8256 }
8257 \tl_if_empty:NT \l__problems_inclprob_title_tl {
8258 \let\l__problems_inclprob_title_tl\undefined
8259 }
8260 \tl_if_empty:NT \l__problems_inclprob_type_tl {
8261 \let\l__problems_inclprob_type_tl\undefined
8262 }
8263 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8264 \let\l__problems_inclprob_refnum_int\undefined
8265 }
8266 }
8267
8268 \cs_new_protected:Nn \l__problems_inclprob_clear: {
8269 \let\l__problems_inclprob_id_str\undefined
8270 \let\l__problems_inclprob_pts_tl\undefined
8271 \let\l__problems_inclprob_min_tl\undefined
8272 \let\l__problems_inclprob_title_tl\undefined
8273 \let\l__problems_inclprob_type_tl\undefined
8274 \let\l__problems_inclprob_refnum_int\undefined
8275 \let\l__problems_inclprob_mhrepos_str\undefined
8276 }
8277 \l__problems_inclprob_clear:
8278
8279 \newcommand\includeproblem[2][]{
8280 \l__problems_inclprob_args:n{ #1 }
8281 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8282 \stex_html_backend:TF {
8283 \str_clear:N \l_tmpa_str
8284 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8285 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8286 }
8287 \stex_annotate_invisible:nnn{includeproblem}{
8288 \l_tmpa_str / #2
8289 }{}}
8290 }{

```

```

8291     \begingroup
8292     \inputreftrue
8293     \tl_if_empty:nTF{ ##1 }{
8294       \input{#2}
8295     }{
8296       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8297     }
8298   \endgroup
8299 }
8300 }
8301 \__problems_inclprob_clear:
8302 }

```

(End definition for `\includeproblem`. This function is documented on page 66.)

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.

```

8303 \AddToHook{enddocument}{
8304   \bool_if:NT \c__problems_pts_bool {
8305     \message{Total:~\arabic{pts}~points}
8306   }
8307   \bool_if:NT \c__problems_min_bool {
8308     \message{Total:~\arabic{min}~minutes}
8309   }
8310 }

```

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

```

8311 \def\pts#1{
8312   \bool_if:NT \c__problems_pts_bool {
8313     \marginpar{#1~\prob@pt@kw}
8314   }
8315 }
8316 \def\min#1{
8317   \bool_if:NT \c__problems_min_bool {
8318     \marginpar{#1~\prob@min@kw}
8319   }
8320 }

```

`\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.

```

8321 \newcounter{pts}
8322 \def\show@pts{
8323   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8324     \bool_if:NT \c__problems_pts_bool {
8325       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8326       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8327     }
8328   }{
8329     \tl_if_exist:NT \l__problems_prob_pts_tl {
8330       \bool_if:NT \c__problems_pts_bool {

```

```

8331         \tl_if_empty:NT\l__problems_prob_pts_tl{
8332             \tl_set:Nn \l__problems_prob_pts_tl {0}
8333         }
8334         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8335         \addtocounter{pts}{\l__problems_prob_pts_tl}
8336     }
8337 }
8338 }
8339 }

```

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

\show@min

```

8340 \newcounter{min}
8341 \def\show@min{
8342     \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8343         \bool_if:NT \c__problems_min_bool {
8344             \marginpar{\l__problems_inclprob_pts_tl\ min}
8345             \addtocounter{min}{\l__problems_inclprob_min_tl}
8346         }
8347     }{
8348         \tl_if_exist:NT \l__problems_prob_min_tl {
8349             \bool_if:NT \c__problems_min_bool {
8350                 \tl_if_empty:NT\l__problems_prob_min_tl{
8351                     \tl_set:Nn \l__problems_prob_min_tl {0}
8352                 }
8353                 \marginpar{\l__problems_prob_min_tl\ min}
8354                 \addtocounter{min}{\l__problems_prob_min_tl}
8355             }
8356         }
8357     }
8358 }
8359 \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.

```
8360 {*package}
8361 \ProvidesExplPackage{hwexam}{2022/05/24}{3.1.0}{homework assignments and exams}
8362 \RequirePackage{13keys2e}
8363
8364 \newif\iftest\testfalse
8365 \DeclareOption{test}{\testtrue}
8366 \newif\ifmultiple\multiplefalse
8367 \DeclareOption{multiple}{\multipletrue}
8368 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8369 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8370 \ProcessOptions
```

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

```
8371 \RequirePackage{keyval}[1997/11/10]
8372 \RequirePackage{problem}
```

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

```
8373 \newcommand\hwexam@assignment@kw{Assignment}
8374 \newcommand\hwexam@given@kw{Given}
8375 \newcommand\hwexam@due@kw{Due}
8376 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8377 \newcommand\hwexam@minutes@kw{minutes}
8378 \newcommand\correction@probs@kw{prob.}
8379 \newcommand\correction@pts@kw{total}
8380 \newcommand\correction@reached@kw{reached}
8381 \newcommand\correction@sum@kw{Sum}
8382 \newcommand\correction@grade@kw{grade}
8383 \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

```

8384 \AddToHook{begindocument}{
8385 \ltx@ifpackageloaded{babel}{
8386 \makeatletter
8387 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8388 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8389 \input{hwexam-ngerman.ldf}
8390 }
8391 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8392 \input{hwexam-finnish.ldf}
8393 }
8394 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8395 \input{hwexam-french.ldf}
8396 }
8397 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8398 \input{hwexam-russian.ldf}
8399 }
8400 \makeatother
8401 }{}
8402 }
8403

```

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.

```

8404 \newcounter{assignment}
8405 %\numberproblemsin{assignment}

```

We will prepare the keyval support for the `assignment` environment.

```

8406 \keys_define:nn { hwexam / assignment } {
8407 id .str_set:N = \l_@@_assign_id_str,
8408 number .int_set:N = \l_@@_assign_number_int,
8409 title .tl_set:N = \l_@@_assign_title_tl,
8410 type .tl_set:N = \l_@@_assign_type_tl,
8411 given .tl_set:N = \l_@@_assign_given_tl,
8412 due .tl_set:N = \l_@@_assign_due_tl,
8413 loadmodules .code:n = {
8414 \bool_set_true:N \l_@@_assign_loadmodules_bool
8415 }
8416 }
8417 \cs_new_protected:Nn \_@@_assignment_args:n {
8418 \str_clear:N \l_@@_assign_id_str
8419 \int_set:Nn \l_@@_assign_number_int {-1}
8420 \tl_clear:N \l_@@_assign_title_tl
8421 \tl_clear:N \l_@@_assign_type_tl
8422 \tl_clear:N \l_@@_assign_given_tl
8423 \tl_clear:N \l_@@_assign_due_tl
8424 \bool_set_false:N \l_@@_assign_loadmodules_bool
8425 \keys_set:nn { hwexam / assignment }{ #1 }
8426 }

```

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.

```

8427 \newcommand\given@due[2]{
8428 \bool_lazy_all:nF {
8429 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8430 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8431 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8432 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8433 }{ #1 }
8434
8435 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8436 \tl_if_empty:NF \l_@@_assign_given_tl {
8437 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8438 }
8439 }{
8440 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8441 }
8442
8443 \bool_lazy_or:nnF {
8444 \bool_lazy_and_p:nn {
8445 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8446 }{
8447 \tl_if_empty_p:V \l_@@_assign_due_tl
8448 }
8449 }{
8450 \bool_lazy_and_p:nn {
8451 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8452 }{
8453 \tl_if_empty_p:V \l_@@_assign_due_tl
8454 }
8455 }{ ,~ }
8456
8457 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8458 \tl_if_empty:NF \l_@@_assign_due_tl {
8459 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8460 }
8461 }{
8462 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8463 }
8464
8465 \bool_lazy_all:nF {
8466 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8467 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8468 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8469 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8470 }{ #2 }
8471 }

```

`\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.

```

8472 \newcommand\assignment@title[3]{
8473 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8474 \tl_if_empty:NTF \l_@@_assign_title_tl {
8475 #1
8476 }{
8477 #2\l_@@_assign_title_tl#3
8478 }
8479 }{
8480 #2\l_@@_inclasssign_title_tl#3
8481 }
8482 }

```

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

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

```

8483 \newcommand\assignment@number{
8484 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8485 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8486 \arabic{assignment}
8487 } {
8488 \int_use:N \l_@@_assign_number_int
8489 }
8490 }{
8491 \int_use:N \l_@@_inclasssign_number_int
8492 }
8493 }

```

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

With them, we can define the central **assignment** environment. This has two forms (separated by \ifmultiple) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

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

```

8494 \newenvironment{assignment}[1][]{
8495 \_@@_assignment_args:n { #1 }
8496 %\sref@target
8497 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8498 \global\stepcounter{assignment}
8499 }{
8500 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8501 }
8502 \setcounter{sproblem}{0}
8503 \renewcommand\prob@label[1]{\assignment@number.##1}
8504 \def\current@section@level{\document@hwexamtype}
8505 %\sref@label{id}{\document@hwexamtype \thesection}
8506 \begin{@assignment}
8507 }{
8508 \end{@assignment}
8509 }

```

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

```

8510 \def\ass@title{
8511 {\protect\document@hwexamtype}\arabic{assignment}
8512 \assignment@title{}\;{}{}\;} -- \given@due{}\}
8513 }
8514 \ifmultiple
8515 \newenvironment{@assignment}{
8516 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8517 \begin{sfragment}[loadmodules]{\ass@title}
8518 }{
8519 \begin{sfragment}{\ass@title}
8520 }
8521 }{
8522 \end{sfragment}
8523 }

```

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

```

8524 \else
8525 \newenvironment{@assignment}{
8526 \begin{center}\bf
8527 \Large@title\strut\
8528 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;{}{}\;{}\\}
8529 \large\given@due{--\;}\;{};--}
8530 \end{center}
8531 }{}
8532 \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.

```

8533 \keys_define:nn { hwexam / inclassignment } {
8534 %id .str_set_x:N = \l_@@_assign_id_str,
8535 number .int_set:N = \l_@@_inclassign_number_int,
8536 title .tl_set:N = \l_@@_inclassign_title_tl,
8537 type .tl_set:N = \l_@@_inclassign_type_tl,
8538 given .tl_set:N = \l_@@_inclassign_given_tl,
8539 due .tl_set:N = \l_@@_inclassign_due_tl,
8540 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8541 }
8542 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8543 \int_set:Nn \l_@@_inclassign_number_int {-1}
8544 \tl_clear:N \l_@@_inclassign_title_tl
8545 \tl_clear:N \l_@@_inclassign_type_tl
8546 \tl_clear:N \l_@@_inclassign_given_tl
8547 \tl_clear:N \l_@@_inclassign_due_tl
8548 \str_clear:N \l_@@_inclassign_mhrepos_str
8549 \keys_set:nn { hwexam / inclassignment }{ #1 }
8550 }
8551 \l_@@_inclassignment_args:n {}
8552
8553 \newcommand\inputassignment[2][]{

```

```

8554 \_@@_inclassassignment_args:n { #1 }
8555 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8556 \input{#2}
8557 }{
8558 \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8559 \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8560 }
8561 }
8562 \_@@_inclassassignment_args:n {}
8563 }
8564 \newcommand\includeassignment[2][]{
8565 \newpage
8566 \inputassignment[#1]{#2}
8567 }

```

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

40.4 Typesetting Exams

\quizheading

```

8568 \ExplSyntaxOff
8569 \newcommand\quizheading[1]{%
8570 \def\@tas{#1}%
8571 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8572 \ifx\@tas\@empty\else%
8573 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8574 \fi%
8575 }
8576 \ExplSyntaxOn

```

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

\testheading

```

8577
8578 \def\hwexamheader{\input{hwexam-default.header}}
8579
8580 \def\hwexamminutes{
8581 \tl_if_empty:NTF \testheading@duration {
8582 {\testheading@min}~\hwexam@minutes@kw
8583 }{
8584 \testheading@duration
8585 }
8586 }
8587
8588 \keys_define:nn { hwexam / testheading } {
8589 min .tl_set:N = \testheading@min,
8590 duration .tl_set:N = \testheading@duration,
8591 reqpts .tl_set:N = \testheading@reqpts,
8592 tools .tl_set:N = \testheading@tools
8593 }
8594 \cs_new_protected:Nn \_@@_testheading_args:n {
8595 \tl_clear:N \testheading@min
8596 \tl_clear:N \testheading@duration

```

```

8597 \tl_clear:N \testheading@reqpts
8598 \tl_clear:N \testheading@tools
8599 \keys_set:nn { hwexam / testheading }{ #1 }
8600 }
8601 \newenvironment{testheading}[1][]{
8602 \_@@_testheading_args:n{ #1 }
8603 \newcount\check@time\check@time=\testheading@min
8604 \advance\check@time by -\theassignment@totalmin
8605 \newif\if@bonuspoints
8606 \tl_if_empty:NTF \testheading@reqpts {
8607 \@bonuspointsfalse
8608 }{
8609 \newcount\bonus@pts
8610 \bonus@pts=\theassignment@totalpts
8611 \advance\bonus@pts by -\testheading@reqpts
8612 \edef\bonus@pts{\the\bonus@pts}
8613 \@bonuspointstrue
8614 }
8615 \edef\check@time{\the\check@time}
8616
8617 \makeatletter\hwexamheader\makeatother
8618 }{
8619 \newpage
8620 }

```

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

\testspace

```

8621 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8622 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8623 \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.

```

8624 <@=problems>
8625 \renewcommand\@problem[3]{
8626 \stepcounter{assignment@probs}
8627 \def\__problemspts{#2}
8628 \ifx\__problemspts\@empty\else
8629 \addtocounter{assignment@totalpts}{#2}
8630 \fi
8631 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8632 \xdef\correction@probs{\correction@probs & #1}%
8633 \xdef\correction@pts{\correction@pts & #2}
8634 \xdef\correction@reached{\correction@reached &}

```

```

8635 }
8636 <@@=hwexam>

```

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

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

```

8637 \newcounter{assignment@probs}
8638 \newcounter{assignment@totalpts}
8639 \newcounter{assignment@totalmin}
8640 \def\correction@probs{\correction@probs@kw}
8641 \def\correction@pts{\correction@pts@kw}
8642 \def\correction@reached{\correction@reached@kw}
8643 \stepcounter{assignment@probs}
8644 \newcommand\correction@table{
8645 \resizebox{\textwidth}{!}{%
8646 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8647 &\multicolumn{\theassignment@probs}{c|}||%|
8648 {\footnotesize\correction@forgrading@kw} &\\ \hline
8649 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8650 \correction@pts & \theassignment@totalpts & \\ \hline
8651 \correction@reached & & \[.7cm]\hline
8652 \end{tabular}}
8653 </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

- [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).