

The sTeX3 Package Collection *

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

2022-07-21

Abstract

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

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

sTeX augments L^AT_EX with

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

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

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

*Version 3.1 (last revised 2022-07-21)

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
	Operator Notations	19
3.3.3	Argument Modes	20
	Mode-b Arguments	20
	Mode-a Arguments	21
	Mode-B Arguments	22
3.3.4	Type and Definiens Components	23
3.3.5	Precedences and Automated Bracketing	24
3.3.6	Variables	26
3.3.7	Variable Sequences	27
3.4	Module Inheritance and Structures	28
3.4.1	Multilinguality and Translations	28
3.4.2	Simple Inheritance and Namespaces	29
3.4.3	The mathstructure Environment	31
3.4.4	The copymodule Environment	34
3.4.5	The interpretmodule Environment	35
3.5	Primitive Symbols (The sTeX Metatheory)	36
4	Using sTeX Symbols	37
4.1	\symref and its variants	37
4.2	Marking Up Text and On-the-Fly Notations	38
4.3	Referencing Symbols and Statements	40

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

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

23	STeX-Basics Implementation	97
23.1	The STeXDocument Class	97
23.2	Preliminaries	98
23.3	Messages and logging	98
23.4	HTML Annotations	99
23.5	Babel Languages	101
23.6	Persistence	102
23.7	Auxiliary Methods	103
24	STeX-MathHub Implementation	106
24.1	Generic Path Handling	106
24.2	PWD and kpsewhich	108
24.3	File Hooks and Tracking	109
24.4	MathHub Repositories	110
24.5	Using Content in Archives	115
25	STeX-References Implementation	119
25.1	Document URIs and URLs	119
25.2	Setting Reference Targets	121
25.3	Using References	123
26	STeX-Modules Implementation	126
26.1	The smodule environment	130
26.2	Invoking modules	136
27	STeX-Module Inheritance Implementation	138
27.1	SMS Mode	138
27.2	Inheritance	142
28	STeX-Symbols Implementation	148
28.1	Symbol Declarations	148
28.2	Notations	156
28.3	Variables	164
29	STeX-Terms Implementation	172
29.1	Symbol Invocations	172
29.2	Terms	179
29.3	Notation Components	183
29.4	Variables	185
29.5	Sequences	188
30	STeX-Structural Features Implementation	189
30.1	Imports with modification	190
30.2	The feature environment	198
30.3	Structure	198
31	STeX-Statements Implementation	209
31.1	Definitions	209
31.2	Assertions	215
31.3	Examples	218
31.4	Logical Paragraphs	221

32 The Implementation	226
32.1 Proofs	226
33 \TeX-Others Implementation	235
34 \TeX-Metatheory Implementation	237
35 Tikzinput Implementation	240
36 document-structure.sty Implementation	243
36.1 Package Options	243
36.2 Document Structure	244
36.3 Front and Backmatter	248
36.4 Global Variables	250
37 NotesSlides – Implementation	251
37.1 Class and Package Options	251
37.2 Notes and Slides	253
37.3 Header and Footer Lines	257
37.4 Frame Images	259
37.5 Sectioning	260
37.6 Excursions	263
38 The Implementation	265
38.1 Package Options	265
38.2 Problems and Solutions	266
38.3 Markup for Added Value Services	273
38.4 Multiple Choice Blocks	273
38.5 Filling in Concrete Solutions	274
38.6 Including Problems	274
38.7 Reporting Metadata	276
39 Implementation: The hwexam Package	278
39.1 Package Options	278
39.2 Assignments	279
39.3 Including Assignments	282
39.4 Typesetting Exams	283
39.5 Leftovers	285
40 References	286

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 \LaTeX Document

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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

Remark 2.2.1:

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

Let's investigate this document in detail to understand the respective parts of the \LaTeX 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[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. `\infinitesum`, `\realdive`, `\realpower`) in the imported module available to the current module `GeometricSeries`. The module `GeometricSeries` “exports” all of these symbols to all modules imports it via an `\importmodule{GeometricSeries}` instruction. Additionally it exports the local symbol `\geometricSeries`.

```
\usemodule
```

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

```
\symdef
```

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

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

```
\comp
```

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

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

What follows are two \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>
```

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

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

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

```

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

```

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

Remark 2.2.2:

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

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

2.2.2 Mmt/OMDoc Conversion

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

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

Chapter 3

Creating sTeX Content

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

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

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

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

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

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

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

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

3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

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

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

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

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

3.2.1 The Local MathHub-Directory

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

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

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

3.2.2 The Structure of \TeX Archives

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

Each such archive needs two subdirectories:

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

An additional `lib`-directory is optional, and is where \TeX will look for files included via `\libinput`.

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

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

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

3.2.3 MANIFEST.MF-Files

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

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

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

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

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

source-base or

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

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

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

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

3.2.4 Using Files in \TeX Archives Directly

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

<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.

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

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

Example 1

Input:

```

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

Output:

Hello World

\stexpatchmodule

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

For example:

Example 2

Input:

```

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

```

Output:

```

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

```

3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

Example 3

Input:

```

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

```

Output:

```

Given a foo, we can...

```

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

Example 4

Input:

```

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

```

Output:

```

this is a symbol taking two arguments.

```

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

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

`\notation`

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

Example 5

Input:

```

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

```

Output:

```

First: a; Second: b

```

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

`\comp`

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

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

Example 6

Input:

```

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

```

Output:

First: *a*; Second: *b*



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



Note that it is required that

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

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

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

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

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

`\symdef`

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

Example 7

Input:

```

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

```

Output:

```

1.: a; 2.: b

```

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

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

\setnotation

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

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

\textsymdecl

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

Operator Notations

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

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

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

Example 8

Input:

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

Output:

`newbinarysymbol` is also occasionally written `a: · ; b: ·`

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

3.3.3 Argument Modes

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

Mode-b Arguments

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

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

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

Example 9

Input:

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

Output:

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

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

Mode-a Arguments

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

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

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

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

Example 10

Input:

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

Output:

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

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

Example 11

Input:

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

Output:

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

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

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

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

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

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

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

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

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

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

Mode-B Arguments

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

Example 12

Input:

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

Output:

$\forall x, y, z. P$

3.3.4 Type and Definiens Components

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

The `type` and `def` keys correspond to the `type` and `definiens` components of

- \hookrightarrow OMDoc/MMT constants.
- \hookrightarrow Correspondingly, the name “type” should be taken with a grain of salt, since
- \hookrightarrow OMDoc/MMT— being foundation-independent — does not a priori implement a fixed typing system.

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

Example 13

Input:

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

Output:

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

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

Example 14

Input:

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

Output:

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

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

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

Output:

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

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

Example 16

Input:

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

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

We will investigate the precise meaning of `<opprec>` and the `<argprec>`s shortly – in the vast majority of cases, it is perfectly sufficient to think of `prec=` taking a single number and having that be *the* precedence of the notation, where lower precedences (somewhat

counterintuitively) bind stronger than higher precedences. So fixing our notations for `\addition` and `\multiplication`, we get:

Example 18

Input:

```

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

```

Output:

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

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

`\infprec`
`\neginfprec`

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



More precisely, each notation takes

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

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

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

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

In the example above:

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



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

3.3.6 Variables

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

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

\svar

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

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

\vardef

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

Example 19

Input:

```

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

```


Output:

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

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

TODO: `bind=forall/exists`

3.3.7 Variable Sequences

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

`\varseq`

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

This is best shown by example:

Example 20

Input:

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

Output:

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

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

TODO: *more notations for invoking sequences.*

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

Example 21

Input:

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

Output:

$a_1 + \dots + a_n$

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

Example 22

Input:

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

Output:

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

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

Example 23

Input:

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

Output:

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

3.4 Module Inheritance and Structures

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

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

3.4.1 Multilinguality and Translations

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

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

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

Technically, each `smodule`-environment induces *two* OMDoc/MMT theories:
 $\begin{array}{ll} \text{\textcolor{blue}{M}} \rightarrow & \text{\code{\begin{smodule}[lang=<lang>]{Foo}}} \text{ generates a theory } \text{some/namespace?Foo} \\ \text{\textcolor{green}{M}} \rightarrow & \text{that only contains the “formal” part of the module – i.e. exactly the content} \\ \text{\textcolor{red}{T}} \rightarrow & \text{that is exported when using } \text{\code{\importmodule}}. \\ & \text{Additionally, MMT generates a } \textit{language theory} \text{ some/namespace/Foo?<lang> that} \\ & \text{includes } \text{some/namespace?Foo} \text{ and contains all the other document content – vari-} \\ & \text{able declarations, includes for each } \text{\code{\usemodule}}, \text{ etc.} \end{array}$

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

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

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

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

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

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

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

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

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



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

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

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

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

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



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

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

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

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

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



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

`\STEXexport`

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



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

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

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

3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

```

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

Output:

A **monoid** is...

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

Example 25

Input:

```

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

Output:

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

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

Example 26

Input:

```

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

```

Output:

```

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

```

\instantiate

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

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

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

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

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

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

\varinstantiate

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

This allows us to do things like:

Example 27

Input:

```

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

```

Output:

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

.

and

Example 28

Input:

```

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

```

Output:

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

.

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

3.4.4 The copymodule Environment

TODO: explain

Given modules:

Example 29

Input:

```

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

```

Output:

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

Example 30

Input:

```

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

```

Output:

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

TODO: explain donotclone

3.4.5 The interpretmodule Environment

TODO: explain

Example 31

Input:

```

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

```

Output:

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

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

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

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

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

Chapter 4

Using \TeX Symbols

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

4.1 `\symref` and its variants

`\symref`
`\symname`

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

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

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

Example 32

Input:

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

Output:

A natural number is...

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

`\Symname`

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

Example 33

Input:

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

Output:

Natural numbers are...



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

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

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

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

4.2 Marking Up Text and On-the-Fly Notations

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

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

Example 34

Input:

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

Output:

The sum of n and m is...

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

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



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

\arg

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

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

Example 35

Input:

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

Output:

```
Addition is...
```

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

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

Example 36

Input:

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

Output:

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

adding k yields...

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

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

Example 37

Input:

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

Output:

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

4.3 Referencing Symbols and Statements

TODO: references documentation

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

Chapter 5

sTEX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

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

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

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

Example 38

Input:

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

Output:

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

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

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

\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 $\langle U, \circ, e \rangle$ where $\circ : U \rightarrow U$ and $e \in U$ such that

Axiom 5.1.2 (Associativity). \circ is associative

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

An example for a **monoid** is...

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

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

```

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

```

⁴EdNOTE: MK: reference

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

5.2 Proofs

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

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

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

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

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

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

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

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

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

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

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

```

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

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

```

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

```

which will produce:

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

Proof: By contradiction

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

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

sulting components be a and b

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

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

3. Then a is even

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

3.2. Hence a^2 is even

\Rightarrow Hence a is even as well

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

4. Then b is also even

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

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

$$= 4a^2$$

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

4.4. Hence b^2 is even

\Rightarrow Hence b is even

By the same argument as above

\Rightarrow Contradiction to a, b being coprime.

□

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

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

Proof: By contradiction

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

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

3. Then a is even

4. Then b is also even

\Rightarrow Contradiction to a, b being coprime.

□

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

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

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

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

```

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

```

This yields the following result:

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

For the induction we have to consider three cases:

1. $n = 1$

then we compute $1 = 1^2$

2. $n = 2$

This case is not really necessary, but we do it for the fun of it (and to get more intuition).

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

3. $n > 1$

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

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

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

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

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

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

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

\yield See above

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

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

\justarg

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

`\sproofend`

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

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

`\sProofEndSymbol`

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

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

5.3 Highlighting and Presentation Customizations

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

The `stexthm` package defines some default customizations that can be used, but of course many existing 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

Additional Packages

6.1 Tikzinput: Treating TIKZ code as images

image

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

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

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

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

\tikzinput
\ctikzinput

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

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

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

`\mhtikzinput`
`\cmhtikzinput`

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

`\libusetikzlibrary`

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

6.2 Modular Document Structuring

6.2.1 Introduction

The `document-structure` package supplies an infrastructure for writing OMDOC documents in \LaTeX . This includes a simple structure sharing mechanism for \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.

6.2.2 Package Options

The `document-structure` package accepts the following options:

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

6.2.3 Document Fragments

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

```

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

```

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

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

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

```

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

```

Here we use two levels of `blindfragment`:

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

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

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

6.2.4 Ending Documents Prematurely

`\prematurestop`
`\afterprematurestop`

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

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

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

6.2.5 Global Document Variables

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

`\setSGvar`
`\useSGvar`

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

`\ifSGvar`

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

6.3 Slides and Course Notes

6.3.1 Introduction

The `notesslides` document class is derived from `beamer.cls` [Tana], it adds a “notes version” for course notes that is more suited to printing than the one supplied by `beamer.cls`.

The `notesslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the `TeX` and `OMDOC`. To

support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `notesslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

In practice we want to generate two documents from the same source: the slides for presentation in the lecture and the course notes as a narrative document for home study. To achieve this, the `notesslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

6.3.2 Package Options

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

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

6.3.3 Notes and Slides

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



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

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

```

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

```

```

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

```

`\ifnotes`

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

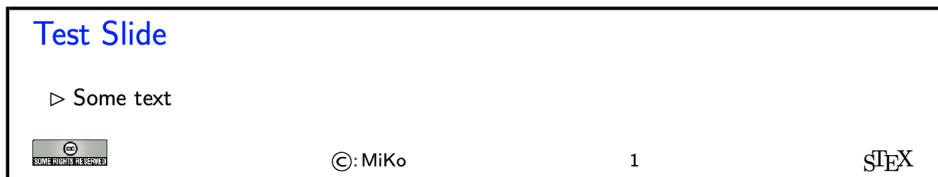
If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

`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.

6.3.4 Customizing Header and Footer Lines

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



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

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

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

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

6.3.5 Frame Images

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

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

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

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


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

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

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

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

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

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

6.3.6 Excursions

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

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

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

`\excursion`

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

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

`\activateexcursion`
`\printexcursion`
`\excursionref`

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

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

`\excursiongroup`

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

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



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

6.4 Representing Problems and Solutions

6.4.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: `hints`, `notes`, and `solutions`⁴. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

Finally, the `problem` package facilitates the management of problems in small files, so that problems can be re-used in multiple environment.

6.4.2 Problems and Solutions

<hr/> <code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> option signifies that we are in
<hr/>	a test situation, so this option does not show the solutions (of course), but leaves space
	for the students to solve them.
 <code>problem</code>	The main environment provided by the <code>problempackage</code> is (surprise surprise) the <code>problem</code>
	environment. It is used to mark up problems and exercises. The environment takes an
	optional KeyVal argument with the keys <code>id</code> as an identifier that can be reference later,
	<code>pts</code> for the points to be gained from this exercise in homework or quiz situations, <code>min</code> for
	the estimated minutes needed to solve the problem, and finally <code>title</code> for an informative
	title of the problem.

Example 40

Input:

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

```

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

```

Output:

Problem 6.4.1 (Fitting Elephants)
 How many Elephants can you fit into a Volkswagen beetle?

Hint: Think positively, this is simple!

Note: Justify your answer

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

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

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

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

\startsolutions
\stopsolutions

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

\ifsolutions

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

6.4.3 Markup for Added-Value Services

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

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

Multiple Choice Blocks

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

`\mcc` `\mcc[⟨keyvals⟩]{⟨text⟩}` takes an optional key/value argument `⟨keyvals⟩` for choice meta-data and a required argument `⟨text⟩` for the proposed answer text. The following keys are supported

- `T` for true answers, `F` for false ones,
- `Ttext` the verdict for true answers, `Ftext` for false ones, and
- `feedback` for a short feedback text given to the student.

What we see when this is formatted to PDF depends on the context. In solutions mode (we start the solutions in the code fragment below) we get

Example 41

Input:

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

Output:

Problem 6.4.2 (Functions)

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

☐ def

Correct!

☐ function

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

☐ fun

Wrong! *that is for Standard ML*

☐ public static void

Wrong! *that is for Java*

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

Example 42

Input:

```

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

```

Output:

Problem 6.4.3 (Functions)

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

☐ def

☐ function

☐ fun

☐ public static void

we get the questions without solutions (that is what the students see during the exam/quiz).

Filling-In Concrete Solutions

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

`\fillinsol`

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

Example 43

Input:

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

Output:

Problem 6.4.4 (Fitting Elephants)

How many Elefants can you fit into a Volkswagen beetle?

and the actual solution in solutions mode:

Example 44

Input:

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

Output:

Problem 6.4.5 (Fitting Elephants)

How many Elefants can you fit into a Volkswagen beetle? 4!

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

6.4.4 Including Problems

`\includeproblem`

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

The sum of the points and estimated minutes (that we specified in the `pts` and `min` keys to the `problem` environment or the `\includeproblem` macro) to the log file and the

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

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

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

6.5 Homeworks, Quizzes and Exams

6.5.1 Introduction

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the `problem` package. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

6.5.2 Package Options

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

6.5.3 Assignments

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

6.5.4 Including Assignments

`\inputassignment`

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

6.5.5 Typesetting Exams

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

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

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

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

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

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-07-21

You have one hour (sharp) for the test;

Write the solutions to the sheet.

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

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

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

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

	To be used for grading, do not write here													
prob.	6.4.1	6.4.2	6.4.3	6.4.4	6.4.5	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total	10					4	4	6	6	4	4	2	40	
reached														

good luck

EdN:8

8

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

Part II

Documentation

Chapter 7

sTeX-Basics

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

7.1 Macros and Environments

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

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

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

7.1.1 HTML Annotations

<code>\if@latexml</code>	L ^A T _E X2e conditional for L ^A T _E XML
--------------------------	---

<code>\latexml_if_p: *</code>	L ^A T _E X3 conditionals for L ^A T _E XML.
<code>\latexml_if:TF *</code>	

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

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

We have four macros for annotating generated HTML (via L^AT_EXML or R_US_TE_X) with attributes:

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

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

7.1.2 Babel Languages

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

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

7.1.3 Auxiliary Methods

<code>\stex_deactivate_macro:Nn</code>
<code>\stex_reactivate_macro:N</code>

`\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}`

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 `\par` control sequences. Expands tokens in the process.

Chapter 8

STEX-MathHub

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

8.1 Macros and Environments

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

8.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

8.1.2 MathHub Archives

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

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

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

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

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

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

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

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

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

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

8.1.3 Using Content in Archives

<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{<archive-ID>}{<filename>}</code> Expands to the full path of file <code><filename></code> in repository <code><archive-ID></code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <code>\mhinput</code>	<code>\inputref[<archive-ID>]{<filename>}</code> Both <code>\input</code> the file <code><filename></code> in archive <code><archive-ID></code> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html-mode</code> , inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<hr/> <hr/> <code>\addmhbibresource</code>	<code>\inputref[<archive-ID>]{<filename>}</code> Adds a <code>.bib</code> -file <code><filename></code> in archive <code><archive-ID></code> (relative to the top-directory of the archive!).
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf-</code> archive of the current archive group(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant <code>lib</code> -folders.
<hr/> <hr/> <code>\libusepackage</code>	<code>\libusepackage[<args>]{<filename>}</code> Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<meta{args}>]{<Arg{filename}>}</code> instead of <code>\input</code> . Throws an error, if none or more than one suitable package file is found.
<hr/> <hr/> <code>\mhgraphics</code> <code>\cmhgraphics</code>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrepos</code> . It then resolves the file path in <code>\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source-</code> folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> 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 9

STEX-References

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

9.1 Macros and Environments

\STEXreftitle

\STEXreftitle{<some title>}

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

\stex_get_document_uri:

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

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

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

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

9.1.1 Setting Reference Targets

\stex_ref_new_doc_target:n

\stex_ref_new_doc_target:n{<id>}

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

\stex_ref_new_sym_target:n

\stex_ref_new_sym_target:n{<uri>}

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

9.1.2 Using References

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

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

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

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

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

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

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

Chapter 10

STEX-Modules

This sub package contains code related to Modules

10.1 Macros and Environments

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

`\c_stex_module_<URI>_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.

`\c_stex_module_<URI>_code`

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

\stex_modules_current_namespace:

Computes the current namespace as follows:

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

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

10.1.1 The smodule environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\stex_invoke_module:n `\stex_invoke_module:n`

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

`\stex_activate_module:n`

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

Chapter 11

STEX-Module Inheritance

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

11.1 Macros and Environments

11.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

`\stex_if_smsmode_p: *`
`\stex_if_smsmode: TF *`

Tests whether SMS mode is currently active.

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

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

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

11.1.2 Imports and Inheritance

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

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

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

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

\stex_import_module_uri:nn

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

\l_stex_import_name_str
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str

stores the result in these four variables.

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

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

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

Chapter 12

STEX-Symbols

Code related to symbol declarations and notations

12.1 Macros and Environments

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

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

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

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

Chapter 13

STEX-Terms

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

13.1 Macros and Environments

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

Chapter 14

TeX-Structural Features

Code related to structural features

14.1 Macros and Environments

14.1.1 Structures

`mathstructure` TODO

Chapter 15

sTeX-Statements

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

15.1 Macros and Environments

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

Chapter 16

sTeX-Proofs: Structural Markup for Proofs

Chapter 17

sT_EX-Metatheory

17.1 Symbols

Part III
Extensions

Chapter 18

Tikzinput: Treating TIKZ code as images

18.1 Macros and Environments

Chapter 19

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

Chapter 20

NotesSlides – Slides and Course Notes

Chapter 21

`problem.sty`: An Infrastructure for formatting Problems

Chapter 22

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

Part IV
Implementation

Chapter 23

\TeX -Basics Implementation

23.1 The \TeX Document Class

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

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

```

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

```

23.2 Preliminaries

```

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

```

\stex The sTeX logo:

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

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

23.3 Messages and logging


```

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

```

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

```

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

```

(End definition for \stex_debug:nn. This function is documented on page 68.)

Redirecting messages:

```

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

```

23.4 HTML Annotations

```

107 <@@=stex_annotate>

```

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

```

108 \tl_new:N \l_stex_html_arg_tl

```

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

`\stex_html_checkempty:n`

```

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

```

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

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

`\stex_if_do_html:TF`

```

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

```

(End definition for `\stex_if_do_html:TF`. This function is documented on page 68.)

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

```

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

```

(End definition for `\stex_suppress_html:n`. This function is documented on page 68.)

`\stex_annotate:anv`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

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

```

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

23.5 Babel Languages

148 `<@=stex_language>`

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

```

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

```

23.6 Persistence

```

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

```

```

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

```

23.7 Auxiliary Methods

\stex_deactivate_macro:Nn

```

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

```

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

\stex_reactivate_macro:N

```

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

```

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

\ignorespacesandpars

```

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

```

```

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

```

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

`\MMTrule`

```

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

```

```

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

```

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

Chapter 24

STEX -MathHub Implementation

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

24.1 Generic Path Handling

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

`\stex_path_from_string:Nn`

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



```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

24.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

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

24.3 File Hooks and Tracking

501 `<@@=stex_files>`

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

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

`\g_stex_files_stack` keeps track of file changes

502 `\seq_gclear_new:N\g_stex_files_stack`

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

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

```

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

`\g_stex_currentfile_seq`

506 `\seq_gclear_new:N\g_stex_currentfile_seq`

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

`\stex_filestack_push:n`

```

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

```

```

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

```

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

\stex_filestack_pop:

```

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

```

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

Hooks for the current file:

```

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

```

24.4 MathHub Repositories

534 $\langle @@=\text{stex_mathhub} \rangle$

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

\c_stex_mathhub_str

\c_stex_mathhub_str

```

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

```

```

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

```

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

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

```

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

```

(End definition for `_stex_mathhub_do_manifest:n`.)

\l_stex_mathhub_manifest_file_seq

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

(End definition for \l_stex_mathhub_manifest_file_seq.)

__stex_mathhub_find_manifest:N

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

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

(End definition for __stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior

File variable used for MANIFEST-files

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

(End definition for \c_stex_mathhub_manifest_ior.)

__stex_mathhub_parse_manifest:n

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

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

```

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

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

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

```

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

`\stex_in_repository:nn`

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

```

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

```



```

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

```

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

24.5 Using Content in Archives

`\mhpath`

```

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

```

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

`\inputref`

`\mhinput`

```

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

```

```

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

```

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

`\addmhbibresource`

```

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

```

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

`\libinput`

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

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

`\libusepackage`

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

```

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

```

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

`\mhgraphics`
`\cmhgraphics`

```

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

```

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

`\lstinputmhlisting`
`\clstinputmhlisting`

```

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

```

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

Chapter 25

STEX -References Implementation

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

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

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

`\STEXreftitle`

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

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

25.1 Document URIs and URLs

`\l_stex_current_docns_str`

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

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

`\stex_get_document_uri:`

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

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

`\l_stex_current_docurl_str`

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

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

`\stex_get_document_url:`

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

```

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

```

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

25.2 Setting Reference Targets

```

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

```

`\stex_ref_new_doc_target:n`

```

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

```

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

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

```

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

```

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

```

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

```

`\stex_ref_new_sym_target:n`

```

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

```



```

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

```

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

25.3 Using References

```

1085 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

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

```

The actual macro:

```

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

```

```

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

```

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

`\srefsym`

```

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

```

```

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

```

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

\srefsymuri

```

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

```

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

```

1211 </package>

```

Chapter 26

STEX -Modules Implementation

```
1212 <*package>
1213
1214 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1215
1216 <@@=stex_modules>
1217
1218 Warnings and error messages
1219 \msg_new:nnn{stex}{error/unknownmodule}{
1220   No~module~#1~found
1221 }
1222 \msg_new:nnn{stex}{error/syntax}{
1223   Syntax~error:~#1
1224 }
1225 \msg_new:nnn{stex}{error/siglanguage}{
1226   Module~#1~declares~signature~#2,~but~does~not~
1227   declare~its~language
1228 }
1229 \msg_new:nnn{stex}{warning/deprecated}{
1230   #1~is~deprecated;~please~use~#2~instead!
1231 }
1232 \msg_new:nnn{stex}{error/conflictingmodules}{
1233   Conflicting~imports~for~module~#1
1234 }
```

`\l_stex_current_module_str` The current module:

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

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

`\l_stex_all_modules_seq` Stores all available modules

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

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

```

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

```

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

```

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

```

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

`\stex_add_to_current_module:n` Only allowed within modules:

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

`\stex_collect_imports:n`

```

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

```

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

`\stex_do_up_to_module:n`

```

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

```

1319 }

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

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

1320

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

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

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

```

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

```

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

26.1 The smodule environment

smodule arguments:

```

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

```

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

```

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

```


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

Are we in a nested module?

```

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

```

Next, we determine the language of the module:

```

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

```

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

```

1454 \str_if_empty:NTF \l_stex_module_sig_str {
1455   \exp_args:Nnx \prop_gset_from_keyval:cn {
1456     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1457   } {
1458     name      = \l_stex_module_name_str ,
1459     ns        = \l_stex_module_ns_str ,
1460     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1461     lang      = \l_stex_module_lang_str ,
1462     sig       = \l_stex_module_sig_str ,
1463     deprecate = \l_stex_module_deprecate_str ,
1464     meta      = \l_stex_module_meta_str
1465   }
1466   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1467   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1468   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1469   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1470   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1471 \str_if_empty:NT \l_stex_module_meta_str {
1472   \str_set:Nx \l_stex_module_meta_str {
1473     \c_stex_metatheory_ns_str ? Metatheory
1474   }
1475 }
1476 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1477   \bool_set_true:N \l_stex_in_meta_bool
1478   \exp_args:Nx \stex_add_to_current_module:n {
1479     \bool_set_true:N \l_stex_in_meta_bool
1480     \stex_activate_module:n {\l_stex_module_meta_str}
1481     \bool_set_false:N \l_stex_in_meta_bool
1482   }
1483   \stex_activate_module:n {\l_stex_module_meta_str}
1484   \bool_set_false:N \l_stex_in_meta_bool
1485 }
1486 }{
1487   \str_if_empty:NT \l_stex_module_lang_str {
1488     \msg_error:nxxx{stex}{error/siglanguage}{
1489       \l_stex_module_ns_str?\l_stex_module_name_str
1490     }\l_stex_module_sig_str}
1491   }
1492   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1493   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1494     \stex_debug:nn{modules}{(already exists)}
1495   }{
1496     \stex_debug:nn{modules}{(needs loading)}
1497     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1498     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1499     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1500     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1501     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1502     \str_set:Nx \l_tmpa_str {
1503       \stex_path_to_string:N \l_tmpa_seq /

```

```

1504     \l_tmpa_str . \l_stex_module_sig_str .tex
1505 }
1506 \IfFileExists \l_tmpa_str {
1507     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1508         \str_clear:N \l_stex_current_module_str
1509         \seq_clear:N \l_stex_all_modules_seq
1510         \stex_debug:nn{modules}{Loading~signature}
1511     }
1512 }{
1513     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1514 }
1515 }
1516 \stex_if_smsmode:F {
1517     \stex_activate_module:n {
1518         \l_stex_module_ns_str ? \l_stex_module_name_str
1519     }
1520 }
1521 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1522 }
1523 \str_if_empty:NF \l_stex_module_deprecate_str {
1524     \msg_warning:nnxx{stex}{warning/deprecated}{
1525         Module~\l_stex_current_module_str
1526     }{
1527         \l_stex_module_deprecate_str
1528     }
1529 }
1530 \seq_put_right:Nx \l_stex_all_modules_seq {
1531     \l_stex_module_ns_str ? \l_stex_module_name_str
1532 }
1533 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1534 }

```

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

smodule The module environment.

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

```

1535 \cs_new_protected:Nn \_stex_modules_begin_module: {
1536     \stex_reactivate_macro:N \STEXexport
1537     \stex_reactivate_macro:N \importmodule
1538     \stex_reactivate_macro:N \symdecl
1539     \stex_reactivate_macro:N \notation
1540     \stex_reactivate_macro:N \symdef
1541 }
1542 \stex_debug:nn{modules}{
1543     New~module:\\
1544     Namespace:~\l_stex_module_ns_str\\
1545     Name:~\l_stex_module_name_str\\
1546     Language:~\l_stex_module_lang_str\\
1547     Signature:~\l_stex_module_sig_str\\
1548     Metatheory:~\l_stex_module_meta_str\\
1549     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1550 }
1551

```

```

1552 \stex_if_do_html:T{
1553   \begin{stex_annotate_env} {theory} {
1554     \l_stex_module_ns_str ? \l_stex_module_name_str
1555   }
1556
1557   \stex_annotate_invisible:nnn{header}{} {
1558     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1559     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1560     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1561       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1562     }
1563     \str_if_empty:NF \smoduletype {
1564       \stex_annotate:nnn{type}{\smoduletype}{}
1565     }
1566   }
1567 }
1568 % TODO: Inherit metatheory for nested modules?
1569 }
1570 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `_stex_modules_begin_module:.`)

`_stex_modules_end_module:` implements `\end{module}`

```

1571 \cs_new_protected:Nn \_stex_modules_end_module: {
1572   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module}
1573   \stex_reset_up_to_module:n \l_stex_current_module_str
1574   \stex_if_smsmode:T {
1575     \stex_persist:x {
1576       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1577         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1578       }
1579       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1580         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1581       }
1582       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1583         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1584       }
1585       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1586     }
1587     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1588     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1589   }
1590 }

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1591 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1592 \NewDocumentEnvironment { smodule } { 0 } { m } {
1593   \stex_module_setup:nn{#1}{#2}
1594   %\par
1595   \stex_if_smsmode:F{
1596     \tl_if_empty:NF \smoduletitle {
1597       \exp_args:No \stex_document_title:n \smoduletitle
1598     }

```

```

1599 \tl_clear:N \l_tmpa_tl
1600 \clist_map_inline:Nn \smodulotype {
1601   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1602     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1603   }
1604 }
1605 \tl_if_empty:NTF \l_tmpa_tl {
1606   \__stex_modules_smodule_start:
1607 }{
1608   \l_tmpa_tl
1609 }
1610 }
1611 \__stex_modules_begin_module:
1612 \str_if_empty:NF \smoduleid {
1613   \stex_ref_new_doc_target:n \smoduleid
1614 }
1615 \stex_smsmode_do:
1616 } {
1617   \__stex_modules_end_module:
1618   \stex_if_smsmode:F {
1619     \end{stex_annotate_env}
1620     \clist_set:Nn \l_tmpa_clist \smodulotype
1621     \tl_clear:N \l_tmpa_tl
1622     \clist_map_inline:Nn \l_tmpa_clist {
1623       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1624         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1625       }
1626     }
1627     \tl_if_empty:NTF \l_tmpa_tl {
1628       \__stex_modules_smodule_end:
1629     }{
1630       \l_tmpa_tl
1631     }
1632   }
1633 }

```

\stexpatchmodule

```

1634 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1635 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1636
1637 \newcommand\stexpatchmodule[3] [] {
1638   \str_set:Nx \l_tmpa_str{ #1 }
1639   \str_if_empty:NTF \l_tmpa_str {
1640     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1641     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1642   }{
1643     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1644     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1645   }
1646 }

```

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

26.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1647 \NewDocumentCommand \STEXModule { m } {
1648   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1649   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1650   \tl_set:Nn \l_tmpa_tl {
1651     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1652   }
1653   \seq_map_inline:Nn \l_stex_all_modules_seq {
1654     \str_set:Nn \l_tmpb_str { ##1 }
1655     \str_if_eq:eeT { \l_tmpa_str } {
1656       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1657     } {
1658       \seq_map_break:n {
1659         \tl_set:Nn \l_tmpa_tl {
1660           \stex_invoke_module:n { ##1 }
1661         }
1662       }
1663     }
1664   }
1665   \l_tmpa_tl
1666 }
1667
1668 \cs_new_protected:Nn \stex_invoke_module:n {
1669   \stex_debug:nn{modules}{Invoking~module~#1}
1670   \peek_charcode_remove:NTF ! {
1671     \__stex_modules_invoke_uri:nN { #1 }
1672   } {
1673     \peek_charcode_remove:NTF ? {
1674       \__stex_modules_invoke_symbol:nn { #1 }
1675     } {
1676       \msg_error:nnx{stex}{error/syntax}{
1677         ?~or~!~expected~after~
1678         \c_backslash_str STEXModule{#1}
1679       }
1680     }
1681   }
1682 }
1683
1684 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1685   \str_set:Nn #2 { #1 }
1686 }
1687
1688 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1689   \stex_invoke_symbol:n{#1?#2}
1690 }

```

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

```

\stex_activate_module:n
1691 \bool_new:N \l_stex_in_meta_bool
1692 \bool_set_false:N \l_stex_in_meta_bool

```

```

1693 \cs_new_protected:Nn \stex_activate_module:n {
1694   \stex_debug:nn{modules}{Activating~module~#1}
1695   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1696     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1697     \use:c{ c_stex_module_#1_code }
1698   }
1699 }

```

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

```

1700 \endpackage

```

Chapter 27

STEX -Module Inheritance Implementation

```
1701 <*package>
1702
1703 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1704
```

27.1 SMS Mode

```
1705 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1706 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1707 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1708 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1709
1710 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1711   \makeatletter
1712   \makeatother
1713   \ExplSyntaxOn
1714   \ExplSyntaxOff
1715   \rustexBREAK
1716 }
1717
1718 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1719   \symdef
1720   \importmodule
1721   \notation
1722   \symdecl
1723   \STEXexport
1724   \inlineass
1725   \inlinedef
1726   \inlineex
1727   \endinput
1728   \setnotation
```



```

1729 \copynotation
1730 \assign
1731 \renamedekl
1732 \donotcopy
1733 \instantiate
1734 \textsymdecl
1735 }
1736
1737 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1738   \tl_to_str:n {
1739     smodule,
1740     copymodule,
1741     interpretmodule,
1742     realization,
1743     sdefinition,
1744     sexample,
1745     sassertion,
1746     sparagraph,
1747     mathstructure
1748   }
1749 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 79.)

`\stex_if_smsmode_p:`

```

\stex_if_smsmode:TF
1750 \bool_new:N \g__stex_smsmode_bool
1751 \bool_set_false:N \g__stex_smsmode_bool
1752 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1753   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1754 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

1755 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1756   \vbox_set:Nn \l_tmpa_box {
1757     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1758     \bool_gset_true:N \g__stex_smsmode_bool
1759     #2
1760     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1761   }
1762   \box_clear:N \l_tmpa_box
1763 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1764 \quark_new:N \q__stex_smsmode_break
1765
1766 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1767   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1768   \stex_smsmode_do:
1769 }
1770

```

```

1771 \cs_new_protected:Nn \__stex_smsmode_module:nn {
1772   \__stex_modules_args:n{#1}
1773   \stex_if_in_module:F {
1774     \str_if_empty:NF \l_stex_module_sig_str {
1775       \stex_modules_current_namespace:
1776       \str_set:Nx \l_stex_module_name_str { #2 }
1777       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1778         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1779         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1780         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1781         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1782         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1783         \str_set:Nx \l_tmpa_str {
1784           \stex_path_to_string:N \l_tmpa_seq /
1785           \l_tmpa_str . \l_stex_module_sig_str .tex
1786         }
1787         \IfFileExists \l_tmpa_str {
1788           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1789         }{
1790           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1791         }
1792       }
1793     }
1794   }
1795 }
1796
1797 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
1798   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
1799   \tl_if_empty:nTF{#1}{
1800     \prop_if_exist:NTF \l_stex_current_repository_prop
1801     {
1802       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
1803       \prg_return_true:
1804     } {
1805       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
1806       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
1807       \tl_if_empty:NT \l_tmpa_tl {
1808         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
1809       }
1810       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
1811       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
1812       \prg_return_true: \prg_return_false:
1813     }
1814   }\prg_return_true:
1815 }
1816
1817 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1818   \stex_filestack_push:n{#1}
1819   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1820   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1821   % ----- new -----
1822   \__stex_smsmode_in_smsmode:nn{#1}{
1823     \let\importmodule\__stex_smsmode_importmodule:
1824     \let\stex_module_setup:nn\__stex_smsmode_module:nn

```

```

1825 \let\__stex_modules_begin_module:\relax
1826 \let\__stex_modules_end_module:\relax
1827 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1828 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1829 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1830 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1831 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1832 \everyeof{\q__stex_smsmode_break\noexpand}
1833 \expandafter\expandafter\expandafter
1834 \stex_smsmode_do:
1835 \csname @ @ input\endcsname "#1"\relax
1836
1837 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1838   \stex_filestack_push:n{##1}
1839   \expandafter\expandafter\expandafter
1840   \stex_smsmode_do:
1841   \csname @ @ input\endcsname "##1"\relax
1842   \stex_filestack_pop:
1843 }
1844 }
1845 % ----- new -----
1846 \__stex_smsmode_in_smsmode:nn{#1} {
1847   #2
1848   % ----- new -----
1849   \begingroup
1850   \%stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1851   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1852     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
1853       \stex_import_module_uri:nn ##1
1854       \stex_import_require_module:nnnn
1855       \l_stex_import_ns_str
1856       \l_stex_import_archive_str
1857       \l_stex_import_path_str
1858       \l_stex_import_name_str \endgroup
1859     }
1860   }
1861   \endgroup
1862   \%stex_debug:nn{smsmode}{Actually~loading~file~#1}
1863   % ----- new -----
1864   \everyeof{\q__stex_smsmode_break\noexpand}
1865   \expandafter\expandafter\expandafter
1866   \stex_smsmode_do:
1867   \csname @ @ input\endcsname "#1"\relax
1868 }
1869 \stex_filestack_pop:
1870 }

```

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

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

```

1871 \cs_new_protected:Npn \stex_smsmode_do: {
1872   \stex_if_smsmode:T {
1873     \__stex_smsmode_do:w

```

```

1874 }
1875 }
1876 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1877   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1878     \expandafter\if\expandafter\relax\noexpand#1
1879     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1880   \else\expandafter\__stex_smsmode_do:w\fi
1881 }{
1882   \__stex_smsmode_do:w % #1
1883 }
1884 }
1885 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1886   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1887     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1888       #1\__stex_smsmode_do:w
1889     }{
1890       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1891         #1
1892       }{
1893         \cs_if_eq:NNTF \begin #1 {
1894           \__stex_smsmode_check_begin:n
1895         }{
1896           \cs_if_eq:NNTF \end #1 {
1897             \__stex_smsmode_check_end:n
1898           }{
1899             \__stex_smsmode_do:w
1900           }
1901         }
1902       }
1903     }
1904   }
1905 }
1906
1907 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1908   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1909     \begin{#1}
1910   }{
1911     \__stex_smsmode_do:w
1912   }
1913 }
1914 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1915   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1916     \end{#1}\__stex_smsmode_do:w
1917   }{
1918     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1919   }
1920 }

```

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

27.2 Inheritance

```

1921 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```
1922 \cs_new_protected:Nn \stex_import_module_uri:nn {
1923   \str_set:Nx \l_stex_import_archive_str { #1 }
1924   \str_set:Nn \l_stex_import_path_str { #2 }
1925
1926   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1927   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1928   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1929
1930   \stex_modules_current_namespace:
1931   \bool_lazy_all:nTF {
1932     {\str_if_empty_p:N \l_stex_import_archive_str}
1933     {\str_if_empty_p:N \l_stex_import_path_str}
1934     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1935   }{
1936     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1937     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1938   }{
1939     \str_if_empty:NT \l_stex_import_archive_str {
1940       \prop_if_exist:NT \l_stex_current_repository_prop {
1941         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1942       }
1943     }
1944     \str_if_empty:NTF \l_stex_import_archive_str {
1945       \str_if_empty:NF \l_stex_import_path_str {
1946         \stex_path_from_string:Nn \l_tmpb_seq {
1947           \l_stex_module_ns_str / .. / \l_stex_import_path_str
1948         }
1949         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
1950         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file:///}
1951       }
1952     }{
1953       \stex_require_repository:n \l_stex_import_archive_str
1954       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1955       \l_stex_import_ns_str
1956       \str_if_empty:NF \l_stex_import_path_str {
1957         \str_set:Nx \l_stex_import_ns_str {
1958           \l_stex_import_ns_str / \l_stex_import_path_str
1959         }
1960       }
1961     }
1962   }
1963 }
```

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

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

Store the return values of `\stex_import_module_uri:nn`.

```
1964 \str_new:N \l_stex_import_name_str
1965 \str_new:N \l_stex_import_archive_str
1966 \str_new:N \l_stex_import_path_str
1967 \str_new:N \l_stex_import_ns_str
```

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

```

\stex_import_require_module:nnnnn {\langle ns \rangle} {\langle archive-ID \rangle} {\langle path \rangle} {\langle name \rangle}
1968 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1969 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1970
1971 \stex_debug:nn{requiremodule}{Here:\~1:\~2:\~3:\~4}
1972
1973 \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1974 \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1975
1976 %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1977
1978 % archive
1979 \str_set:Nx \l_tmpa_str { #2 }
1980 \str_if_empty:NTF \l_tmpa_str {
1981 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1982 \seq_put_right:Nn \l_tmpa_seq {...}
1983 } {
1984 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1985 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1986 \seq_put_right:Nn \l_tmpa_seq { source }
1987 }
1988
1989 % path
1990 \str_set:Nx \l_tmpb_str { #3 }
1991 \str_if_empty:NTF \l_tmpb_str {
1992 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1993
1994 \ltx@ifpackageloaded{babel} {
1995 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1996 { \language } \l_tmpb_str {
1997 \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1998 }
1999 } {
2000 \str_clear:N \l_tmpb_str
2001 }
2002
2003 \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2004 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2005 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2006 }{
2007 \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2008 \IfFileExists{ \l_tmpa_str.tex }{
2009 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2010 }{
2011 % try english as default
2012 \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2013 \IfFileExists{ \l_tmpa_str.en.tex }{
2014 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2015 }{
2016 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2017 }
2018 }
2019 }
2020

```

```

2021 } {
2022   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2023   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2024
2025   \ltx@ifpackageloaded{babel} {
2026     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2027       { \language } \l_tmpb_str {
2028       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2029     }
2030   } {
2031     \str_clear:N \l_tmpb_str
2032   }
2033
2034   \stex_path_canonicalize:N \l_tmpb_seq
2035   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2036
2037   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2038   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2039     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2040   }{
2041     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2042     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2043       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2044     }{
2045       % try english as default
2046       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2047       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2048         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2049       }{
2050         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2051         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2052           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2053         }{
2054           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2055           \IfFileExists{ \l_tmpa_str.tex }{
2056             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2057           }{
2058             % try english as default
2059             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2060             \IfFileExists{ \l_tmpa_str.en.tex }{
2061               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2062             }{
2063               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2064             }
2065           }
2066         }
2067       }
2068     }
2069   }
2070 }
2071
2072 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2073   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2074     \seq_clear:N \l_stex_all_modules_seq

```

```

2075     \str_clear:N \l_stex_current_module_str
2076     \str_set:Nx \l_tmpb_str { #2 }
2077     \str_if_empty:NF \l_tmpb_str {
2078       \stex_set_current_repository:n { #2 }
2079     }
2080     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2081   }
2082
2083   \stex_if_module_exists:nF { #1 ? #4 } {
2084     \msg_error:nnx{stex}{error/unknownmodule}{
2085       #1?#4~(in~file~\g__stex_importmodule_file_str)
2086     }
2087   }
2088 }
2089
2090 }
2091 \stex_activate_module:n { #1 ? #4 }
2092 }

```

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

`\importmodule`

```

2093 \NewDocumentCommand \importmodule { 0{} m } {
2094   \stex_import_module_uri:nn { #1 } { #2 }
2095   \stex_debug:nn{modules}{Importing~module:~
2096     \l_stex_import_ns_str ? \l_stex_import_name_str
2097   }
2098   \stex_import_require_module:nnnn
2099   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2100   { \l_stex_import_path_str } { \l_stex_import_name_str }
2101   \stex_if_smsmode:F {
2102     \stex_annotate_invisible:nnn
2103     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2104   }
2105   \exp_args:Nx \stex_add_to_current_module:n {
2106     \stex_import_require_module:nnnn
2107     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2108     { \l_stex_import_path_str } { \l_stex_import_name_str }
2109   }
2110   \exp_args:Nx \stex_add_import_to_current_module:n {
2111     \l_stex_import_ns_str ? \l_stex_import_name_str
2112   }
2113   \stex_smsmode_do:
2114   \ignorespacesandpars
2115 }
2116 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

2117 \NewDocumentCommand \usemodule { 0{} m } {
2118   \stex_if_smsmode:F {
2119     \stex_import_module_uri:nn { #1 } { #2 }
2120     \stex_import_require_module:nnnn
2121     { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```



```

2122     { \l_stex_import_path_str } { \l_stex_import_name_str }
2123     \stex_annotate_invisible:nnn
2124     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2125   }
2126   \stex_smsmode_do:
2127   \ignorespacesandpars
2128 }

```

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

```

2129 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2130   \tl_if_empty:nF{#2}{
2131     \clist_set:Nn \l_tmpa_clist {#2}
2132     \clist_map_inline:Nn \l_tmpa_clist {
2133       \tl_if_head_eq_charcode:nNTF {##1} [{
2134         #1 ##1
2135       } {
2136         #1{##1}
2137       }
2138     }
2139   }
2140 }
2141 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2142
2143
2144 \endpackage

```

Chapter 28

STEX -Symbols Implementation

```
2145 <*package>
2146
2147 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2148
2149 Warnings and error messages
2150 \msg_new:nnn{stex}{error/wrongargs}{
2151   args~value~in~symbol~declaration~for~#1~
2152   needs~to~be~i,~a,~b~or~B,~but~#2~given
2153 }
2154 \msg_new:nnn{stex}{error/unknownsymbol}{
2155   No~symbol~#1~found!
2156 }
2157 \msg_new:nnn{stex}{error/seqlength}{
2158   Expected~#1~arguments;~got~#2!
2159 }
2160 \msg_new:nnn{stex}{error/unknownnotation}{
2161   Unknown~notation~#1~for~#2!
2162 }
```

28.1 Symbol Declarations

```
2162 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2163 \cs_new_protected:Nn \stex_all_symbols:n {
2164   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2165   \seq_map_inline:Nn \l_stex_all_modules_seq {
2166     \seq_map_inline:cn{c_stex_module_##1_constants}{
2167       \__stex_symdecl_all_symbols_cs{##1?####1}
2168     }
2169   }
2170 }
```

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

\STEXsymbol

```
2171 \NewDocumentCommand \STEXsymbol { m } {
2172   \stex_get_symbol:n { #1 }
2173   \exp_args:No
2174   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2175 }
```

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

symdecl arguments:

```
2176 \keys_define:nn { stex / symdecl } {
2177   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2178   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2179   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2180   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2181   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2182   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2183   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2184   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2185   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2186   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2187   assoc     .choices:nn =
2188     {bin,binl,binr,pre,conj,pwconj}
2189     {\str_set:Nx \l_stex_symdecl ASSOCTYPE_STR {\l_keys_choice_tl}}
2190 }
2191
2192 \bool_new:N \l_stex_symdecl_make_macro_bool
2193
2194 \cs_new_protected:Nn \__stex_symdecl_args:n {
2195   \str_clear:N \l_stex_symdecl_name_str
2196   \str_clear:N \l_stex_symdecl_args_str
2197   \str_clear:N \l_stex_symdecl_deprecate_str
2198   \str_clear:N \l_stex_symdecl_reorder_str
2199   \str_clear:N \l_stex_symdecl ASSOCTYPE_STR
2200   \bool_set_false:N \l_stex_symdecl_local_bool
2201   \tl_clear:N \l_stex_symdecl_type_tl
2202   \tl_clear:N \l_stex_symdecl_definiens_tl
2203
2204   \keys_set:nn { stex / symdecl } { #1 }
2205 }
```

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

```
2206
2207 \NewDocumentCommand \symdecl { s m O{} } {
2208   \__stex_symdecl_args:n { #3 }
2209   \IfBooleanTF #1 {
2210     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2211   } {
2212     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2213   }
2214   \stex_symdecl_do:n { #2 }
2215   \stex_smsmode_do:
2216 }
```

```

2217
2218 \cs_new_protected:Nn \stex_symdecl_do:nn {
2219   \__stex_symdecl_args:n{#1}
2220   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2221   \stex_symdecl_do:n{#2}
2222 }
2223
2224 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2225 \cs_new_protected:Nn \stex_symdecl_do:n {
2226   \stex_if_in_module:F {
2227     % TODO throw error? some default namespace?
2228   }
2229
2230   \str_if_empty:NT \l_stex_symdecl_name_str {
2231     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2232   }
2233
2234   \prop_if_exist:cT { l_stex_symdecl_
2235     \l_stex_current_module_str ?
2236     \l_stex_symdecl_name_str
2237     _prop
2238   }{
2239     % TODO throw error (beware of circular dependencies)
2240   }
2241
2242   \prop_clear:N \l_tmpa_prop
2243   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2244   \seq_clear:N \l_tmpa_seq
2245   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2246   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2247
2248   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2249     \str_if_empty:NF \l_stex_module_deprecate_str {
2250       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2251     }
2252   }
2253   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2254
2255   \exp_args:No \stex_add_constant_to_current_module:n {
2256     \l_stex_symdecl_name_str
2257   }
2258
2259   % arity/args
2260   \int_zero:N \l_tmpb_int
2261
2262   \bool_set_true:N \l_tmpa_bool
2263   \str_map_inline:Nn \l_stex_symdecl_args_str {
2264     \token_case_meaning:NnF ##1 {
2265       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2266       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2267     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2268     {\tl_to_str:n a} {
2269         \bool_set_false:N \l_tmpa_bool
2270         \int_incr:N \l_tmpb_int
2271     }
2272     {\tl_to_str:n B} {
2273         \bool_set_false:N \l_tmpa_bool
2274         \int_incr:N \l_tmpb_int
2275     }
2276   }{
2277     \msg_error:nnxx{stex}{error/wrongargs}{
2278       \l_stex_current_module_str ?
2279       \l_stex_symdecl_name_str
2280     }{##1}
2281   }
2282 }
2283 \bool_if:NTF \l_tmpa_bool {
2284   % possibly numeric
2285   \str_if_empty:NTF \l_stex_symdecl_args_str {
2286     \prop_put:Nnn \l_tmpa_prop { args } {}
2287     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2288   }{
2289     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2290     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2291     \str_clear:N \l_tmpa_str
2292     \int_step_inline:nn \l_tmpa_int {
2293       \str_put_right:Nn \l_tmpa_str i
2294     }
2295     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2296   }
2297 } {
2298   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2299   \prop_put:Nnx \l_tmpa_prop { arity }
2300   { \str_count:N \l_stex_symdecl_args_str }
2301 }
2302 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2303
2304 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2305   \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2306 }{
2307   \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2308 }
2309
2310 % semantic macro
2311
2312 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2313   \exp_args:Nx \stex_do_up_to_module:n {
2314     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2315       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2316     }}
2317   }
2318 }
2319
2320 \stex_debug:nn{symbols}{New~symbol:~

```

```

2321 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2322 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2323 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2324 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2325 }
2326
2327 % circular dependencies require this:
2328 \stex_if_do_html:T {
2329   \stex_annotate_invisible:nnn {symdecl} {
2330     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2331   } {
2332     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2333       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2334     }
2335     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{ }
2336     \stex_annotate_invisible:nnn{macroname}{#1}{ }
2337     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2338       \stex_annotate_invisible:nnn{definiens}{ }
2339       {\l_stex_symdecl_definiens_tl$}
2340     }
2341     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2342       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{ }
2343     }
2344     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2345       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{ }
2346     }
2347   }
2348 }
2349 \prop_if_exist:cF {
2350   \l_stex_symdecl_
2351   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2352   _prop
2353 } {
2354   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2355   \__stex_symdecl_restore_symbol:nnnnnnn
2356   {\l_stex_symdecl_name_str}
2357   { \prop_item:Nn \l_tmpa_prop {args} }
2358   { \prop_item:Nn \l_tmpa_prop {arity} }
2359   { \prop_item:Nn \l_tmpa_prop {assocs} }
2360   { \prop_item:Nn \l_tmpa_prop {defined} }
2361   {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2362   {\l_stex_current_module_str}
2363 }
2364 }
2365 }
2366 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2367   \prop_clear:N \l_tmpa_prop
2368   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2369   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2370   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2371   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2372   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2373   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2374   \tl_if_empty:nF{#6}{

```

```

2375     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2376   }
2377   \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2378   \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2379 }

```

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

`\textsymdecl`

```

2380
2381 \keys_define:nn { stex / textsymdecl } {
2382   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2383   type      .tl_set:N    = \l__stex_symdecl_type_tl
2384 }
2385
2386 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2387   \str_clear:N \l__stex_symdecl_name_str
2388   \tl_clear:N \l__stex_symdecl_type_tl
2389   \keys_set:nn { stex / textsymdecl } { #1 }
2390 }
2391
2392 \NewDocumentCommand \textsymdecl {m O{} m} {
2393   \_stex_textsymdecl_args:n { #2 }
2394   \str_if_empty:NTF \l__stex_symdecl_name_str {
2395     \__stex_symdecl_args:n{name=#1,#2}
2396   }{
2397     \__stex_symdecl_args:n{#2}
2398   }
2399   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2400   \stex_symdecl_do:n{#1-sym}
2401   \stex_execute_in_module:n{
2402     \cs_set_nopar:cpn{#1name}{
2403       \ifvmode\hbox_unpack:N\c_empty_box\fi
2404       \ifmmode\hbox{#3}\else#3\fi\hspace
2405     }
2406     \cs_set_nopar:cpn{#1}{
2407       \ifmmode\csname#1-sym\expandafter\endcsname\else
2408       \ifvmode\hbox_unpack:N\c_empty_box\fi
2409       \symref{#1-sym}{#3}\expandafter\hspace
2410       \fi
2411     }
2412   }
2413   \stex_execute_in_module:x{
2414     \__stex_notation_restore_notation:nnnnn
2415     {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl
2416     }{0}
2417     {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{
2418       \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2419     }}
2420   }
2421 }
2422 \stex_smsmode_do:
2423 }

```

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

`\stex_get_symbol:n`

```
2424 \str_new:N \l_stex_get_symbol_uri_str
2425
2426 \cs_new_protected:Nn \stex_get_symbol:n {
2427   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2428     \tl_set:Nn \l_tmpa_tl { #1 }
2429     \__stex_symdecl_get_symbol_from_cs:
2430   }{
2431     % argument is a string
2432     % is it a command name?
2433     \cs_if_exist:cTF { #1 }{
2434       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2435       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2436       \str_if_empty:NTF \l_tmpa_str {
2437         \exp_args:Nx \cs_if_eq:NNTF {
2438           \tl_head:N \l_tmpa_tl
2439         } \stex_invoke_symbol:n {
2440           \__stex_symdecl_get_symbol_from_cs:
2441         }{
2442           \__stex_symdecl_get_symbol_from_string:n { #1 }
2443         }
2444       } {
2445         \__stex_symdecl_get_symbol_from_string:n { #1 }
2446       }
2447     }{
2448       % argument is not a command name
2449       \__stex_symdecl_get_symbol_from_string:n { #1 }
2450       % \l_stex_all_symbols_seq
2451     }
2452   }
2453   \str_if_eq:eeF {
2454     \prop_item:cn {
2455       l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2456     }{ deprecate }
2457   }{}{
2458     \msg_warning:nnxx{stex}{warning/deprecated}{
2459       Symbol~\l_stex_get_symbol_uri_str
2460     }{
2461       \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2462     }
2463   }
2464 }
2465
2466 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2467   \tl_set:Nn \l_tmpa_tl {
2468     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2469   }
2470   \str_set:Nn \l_tmpa_str { #1 }
2471
2472   %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2473
2474   \str_if_in:NnTF \l_tmpa_str ? {
2475     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2476     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```



```

2477 \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2478 }{
2479 \str_clear:N \l_tmpb_str
2480 }
2481 \str_if_empty:NTF \l_tmpb_str {
2482 \seq_map_inline:Nn \l_stex_all_modules_seq {
2483 \seq_map_inline:cn{c_stex_module_###1_constants}{
2484 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2485 \seq_map_break:n{\seq_map_break:n{
2486 \tl_set:Nn \l_tmpa_tl {
2487 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2488 }
2489 }}
2490 }
2491 }
2492 }
2493 }{
2494 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2495 \seq_map_inline:Nn \l_stex_all_modules_seq {
2496 \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2497 \seq_map_inline:cn{c_stex_module_###1_constants}{
2498 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2499 \seq_map_break:n{\seq_map_break:n{
2500 \tl_set:Nn \l_tmpa_tl {
2501 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2502 }
2503 }}
2504 }
2505 }
2506 }
2507 }
2508 }
2509
2510 \l_tmpa_tl
2511 }
2512
2513 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2514 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2515 { \tl_tail:N \l_tmpa_tl }
2516 \tl_if_single:NTF \l_tmpa_tl {
2517 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2518 \exp_after:wN \str_set:Nn \exp_after:wN
2519 \l_stex_get_symbol_uri_str \l_tmpa_tl
2520 }{
2521 % TODO
2522 % tail is not a single group
2523 }
2524 }{
2525 % TODO
2526 % tail is not a single group
2527 }
2528 }

```

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

28.2 Notations

```

2529 <@@=stex_notation>

      notation arguments:
2530 \keys_define:nn { stex / notation } {
2531 % lang .tl_set_x:N = \l__stex_notation_lang_str ,
2532 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2533 prec .str_set_x:N = \l__stex_notation_prec_str ,
2534 op .tl_set:N = \l__stex_notation_op_tl ,
2535 primary .bool_set:N = \l__stex_notation_primary_bool ,
2536 primary .default:n = {true} ,
2537 unknown .code:n = \str_set:Nx
2538     \l__stex_notation_variant_str \l_keys_key_str
2539 }
2540
2541 \cs_new_protected:Nn \stex_notation_args:n {
2542 % \str_clear:N \l__stex_notation_lang_str
2543 \str_clear:N \l__stex_notation_variant_str
2544 \str_clear:N \l__stex_notation_prec_str
2545 \tl_clear:N \l__stex_notation_op_tl
2546 \bool_set_false:N \l__stex_notation_primary_bool
2547
2548 \keys_set:nn { stex / notation } { #1 }
2549 }

\notation

2550 \NewDocumentCommand \notation { s m O{}} {
2551 \stex_notation_args:n { #3 }
2552 \tl_clear:N \l_stex_symdecl_definiens_tl
2553 \stex_get_symbol:n { #2 }
2554 \tl_set:Nn \l_stex_notation_after_do_tl {
2555     \__stex_notation_final:
2556     \IfBooleanTF#1{
2557         \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2558     }{}
2559     \stex_smsmode_do:\ignorespacesandpars
2560 }
2561 \stex_notation_do:nnnnn
2562 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { args } }
2563 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
2564 { \l__stex_notation_variant_str }
2565 { \l__stex_notation_prec_str }
2566 }
2567 \stex_deactivate_macro:Nn \notation {module-environments}

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

\stex_notation_do:nnnnn

2568 \seq_new:N \l__stex_notation_precedences_seq
2569 \tl_new:N \l__stex_notation_opprec_tl
2570 \int_new:N \l__stex_notation_currarg_int
2571 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2572
2573 \cs_new_protected:Nn \stex_notation_do:nnnnn {

```

```

2574 \let\STEXInternalCurrentSymbolStr\relax
2575 \seq_clear:N \l__stex_notation_precedences_seq
2576 \tl_clear:N \l__stex_notation_opprec_tl
2577 \str_set:Nx \l__stex_notation_args_str { #1 }
2578 \str_set:Nx \l__stex_notation_arity_str { #2 }
2579 \str_set:Nx \l__stex_notation_suffix_str { #3 }
2580 \str_set:Nx \l__stex_notation_prec_str { #4 }
2581
2582 % precedences
2583 \str_if_empty:NTF \l__stex_notation_prec_str {
2584   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2585     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2586   }{
2587     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2588   }
2589 } {
2590   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2591     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2592     \int_step_inline:nn { \l__stex_notation_arity_str } {
2593       \exp_args:NNo
2594       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2595     }
2596   }{
2597     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2598     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2599       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2600       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2601         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2602           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2603         \seq_map_inline:Nn \l_tmpa_seq {
2604           \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }
2605         }
2606       }
2607     }{
2608       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2609         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2610       }{
2611         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2612       }
2613     }
2614   }
2615 }
2616
2617 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2618 \int_step_inline:nn { \l__stex_notation_arity_str } {
2619   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2620     \exp_args:NNo
2621     \seq_put_right:No \l__stex_notation_precedences_seq {
2622       \l__stex_notation_opprec_tl
2623     }
2624   }
2625 }
2626 \tl_clear:N \l__stex_notation_dummyargs_tl
2627

```

```

2628 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2629   \exp_args:NNe
2630   \cs_set:Npn \l_stex_notation_macrocode_cs {
2631     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2632     { \l__stex_notation_suffix_str }
2633     { \l__stex_notation_opprec_tl }
2634     { \exp_not:n { #5 } }
2635   }
2636   \l_stex_notation_after_do_tl
2637 }{
2638   \str_if_in:NnTF \l__stex_notation_args_str b {
2639     \exp_args:Nne \use:nn
2640     {
2641       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2642       \cs_set:Npn \l__stex_notation_arity_str } { {
2643         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2644         { \l__stex_notation_suffix_str }
2645         { \l__stex_notation_opprec_tl }
2646         { \exp_not:n { #5 } }
2647       }}
2648   }{
2649     \str_if_in:NnTF \l__stex_notation_args_str B {
2650       \exp_args:Nne \use:nn
2651       {
2652         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2653         \cs_set:Npn \l__stex_notation_arity_str } { {
2654           \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2655           { \l__stex_notation_suffix_str }
2656           { \l__stex_notation_opprec_tl }
2657           { \exp_not:n { #5 } }
2658         } }
2659     }{
2660       \exp_args:Nne \use:nn
2661       {
2662         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2663         \cs_set:Npn \l__stex_notation_arity_str } { {
2664           \STEXInternalTermMathOMAiinii { \STEXInternalCurrentSymbolStr }
2665           { \l__stex_notation_suffix_str }
2666           { \l__stex_notation_opprec_tl }
2667           { \exp_not:n { #5 } }
2668         } }
2669     }
2670   }
2671
2672   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2673   \int_zero:N \l__stex_notation_currarg_int
2674   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2675   \__stex_notation_arguments:
2676 }
2677 }

```

(End definition for \stex_notation_do:nnnnn. This function is documented on page ??.)

`__stex_notation_arguments:` Takes care of annotating the arguments in a notation macro

```

2678 \cs_new_protected:Nn \__stex_notation_arguments: {
2679   \int_incr:N \l__stex_notation_currarg_int
2680   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2681     \l_stex_notation_after_do_tl
2682   }{
2683     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2684     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2685     \str_if_eq:VnTF \l_tmpa_str a {
2686       \__stex_notation_argument_assoc:nn{a}
2687     }{
2688       \str_if_eq:VnTF \l_tmpa_str B {
2689         \__stex_notation_argument_assoc:nn{B}
2690       }{
2691         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2692         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2693           { \STEXInternalTermMathArgiii
2694             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2695             { \l_tmpb_str }
2696             { ####\int_use:N \l__stex_notation_currarg_int }
2697           }
2698         }
2699         \__stex_notation_arguments:
2700       }
2701     }
2702   }
2703 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```

2704 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2705
2706   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2707     {\l__stex_notation_arity_str}{
2708       #2
2709     }
2710   \int_zero:N \l_tmpa_int
2711   \tl_clear:N \l_tmpa_tl
2712   \str_map_inline:Nn \l__stex_notation_args_str {
2713     \int_incr:N \l_tmpa_int
2714     \tl_put_right:Nx \l_tmpa_tl {
2715       \str_if_eq:nnTF {##1}{a}{ {} }{ }{
2716         \str_if_eq:nnTF {##1}{B}{ {} }{ }{
2717           {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2718         }
2719       }
2720     }
2721   }
2722   \exp_after:wN\exp_after:wN\exp_after:wN \def
2723   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2724   \exp_after:wN\exp_after:wN\exp_after:wN ##
2725   \exp_after:wN\exp_after:wN\exp_after:wN 1
2726   \exp_after:wN\exp_after:wN\exp_after:wN ##
2727   \exp_after:wN\exp_after:wN\exp_after:wN 2

```

```

2728 \exp_after:wN\exp_after:wN\exp_after:wN {
2729   \exp_after:wN \exp_after:wN \exp_after:wN
2730   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2731     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2732   }
2733 }
2734
2735 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2736 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2737   \STEXInternalTermMathAssocArgiiii
2738   { #1\int_use:N \l__stex_notation_currarg_int }
2739   { \l_tmpa_str }
2740   { ####\int_use:N \l__stex_notation_currarg_int }
2741   { \l_tmpa_cs {####1} {####2} }
2742 } }
2743 \__stex_notation_arguments:
2744 }

```

(End definition for __stex_notation_argument_assoc:nn.)

__stex_notation_final: Called after processing all notation arguments

```

2745 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2746   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2747   \cs_set_nopar:Npn {#3}{#4}
2748   \tl_if_empty:nF {#5}{
2749     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2750   }
2751   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2752     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2753   }
2754 }
2755
2756 \cs_new_protected:Nn \__stex_notation_final: {
2757
2758   \stex_execute_in_module:x {
2759     \__stex_notation_restore_notation:nnnnn
2760     {\l_stex_get_symbol_uri_str}
2761     {\l__stex_notation_suffix_str}
2762     {\l__stex_notation_arity_str}
2763     {
2764       \exp_after:wN \exp_after:wN \exp_after:wN
2765       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2766       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
2767     }
2768     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2769   }
2770
2771   \stex_debug:nn{symbols}{
2772     Notation~\l__stex_notation_suffix_str
2773     ~for~\l_stex_get_symbol_uri_str^^J
2774     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2775     Argument~precedences:~
2776     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2777     Notation: \cs_meaning:c {

```

```

2778     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2779     \l__stex_notation_suffix_str
2780     _cs
2781 }
2782 }
2783 % HTML annotations
2784 \stex_if_do_html:T {
2785   \stex_annotate_invisible:nnn { notation }
2786   { \l_stex_get_symbol_uri_str } {
2787     \stex_annotate_invisible:nnn { notationfragment }
2788     { \l__stex_notation_suffix_str }{}
2789     \stex_annotate_invisible:nnn { precedence }
2790     { \l__stex_notation_prec_str }{}
2791
2792     \int_zero:N \l_tmpa_int
2793     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2794     \tl_clear:N \l_tmpa_tl
2795     \int_step_inline:nn { \l__stex_notation_arity_str }{
2796       \int_incr:N \l_tmpa_int
2797       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2798       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
2799       \str_if_eq:VnTF \l_tmpb_str a {
2800         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2801           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2802           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2803         } }
2804       }{
2805         \str_if_eq:VnTF \l_tmpb_str B {
2806           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2807             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2808             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2809           } }
2810         }{
2811           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2812             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2813           } }
2814         }
2815       }
2816     }
2817     \stex_annotate_invisible:nnn { notationcomp }{}{
2818       \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
2819       $ \exp_args:Nno \use:nn { \use:c {
2820         stex_notation_ \STEXInternalCurrentSymbolStr
2821         \c_hash_str \l__stex_notation_suffix_str _cs
2822       } } { \l_tmpa_tl } $
2823     }
2824     \tl_if_empty:NF \l__stex_notation_op_tl {
2825       \stex_annotate_invisible:nnn { notationopcomp }{}{
2826         $\l__stex_notation_op_tl$
2827       }
2828     }
2829   }
2830 }
2831 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2832 \keys_define:nn { stex / setnotation } {
2833   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2834   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2835   unknown .code:n      = \str_set:Nx
2836     \l__stex_notation_variant_str \l_keys_key_str
2837 }
2838
2839 \cs_new_protected:Nn \stex_setnotation_args:n {
2840   % \str_clear:N \l__stex_notation_lang_str
2841   \str_clear:N \l__stex_notation_variant_str
2842   \keys_set:nn { stex / setnotation } { #1 }
2843 }
2844
2845 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
2846   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2847     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2848     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2849   }
2850 }
2851
2852 \cs_new_protected:Nn \stex_setnotation:n {
2853   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2854     { \l__stex_notation_variant_str }{
2855     \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2856     \stex_debug:nn {notations}{
2857       Setting~default~notation~
2858       {\l__stex_notation_variant_str }~for~
2859       #1 \\
2860       \expandafter\meaning\csname
2861       l_stex_symdecl_#1_notations\endcsname
2862     }
2863   }{
2864     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2865   }
2866 }
2867
2868 \NewDocumentCommand \setnotation {m m} {
2869   \stex_get_symbol:n { #1 }
2870   \_stex_setnotation_args:n { #2 }
2871   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2872   \stex_smsmode_do:\ignorespacesandpars
2873 }
2874
2875 \cs_new_protected:Nn \stex_copy_notations:nn {
2876   \stex_debug:nn {notations}{
2877     Copying~notations~from~#2~to~#1\\
2878     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2879   }
2880   \tl_clear:N \l_tmpa_tl
2881   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2882     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }

```



```

2883 }
2884 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
2885   \stex_debug:nn{Here}{Here:~##1}
2886   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2887   \edef \l_tmpa_tl {
2888     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2889     \exp_after:wN\exp_after:wN\exp_after:wN {
2890       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2891     }
2892   }
2893
2894   \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2895   \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2896   \exp_after:wN { \l_tmpa_tl }
2897
2898   \edef \l_tmpa_tl {
2899     \exp_after:wN \exp_not:n \exp_after:wN {
2900       \l_tmpa_tl {##### 1}{##### 2}
2901     }
2902   }
2903
2904   \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
2905
2906   \stex_execute_in_module:x {
2907     \__stex_notation_restore_notation:nnnnn
2908     {#1}{##1}
2909     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2910     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2911     {
2912       \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2913         \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2914       }
2915     }
2916   }\endgroup
2917 }
2918 }
2919
2920 \NewDocumentCommand \copynotation {m m} {
2921   \stex_get_symbol:n { #1 }
2922   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2923   \stex_get_symbol:n { #2 }
2924   \exp_args:Noo
2925   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2926   \stex_smsmode_do:\ignorespacesandpars
2927 }
2928

```

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

\symdef

```

2929 \keys_define:nn { stex / symdef } {
2930   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2931   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2932   args      .str_set_x:N = \l_stex_symdecl_args_str ,

```

```

2933 type .tl_set:N = \l_stex_symdecl_type_tl ,
2934 def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2935 reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
2936 op .tl_set:N = \l__stex_notation_op_tl ,
2937 % lang .str_set_x:N = \l__stex_notation_lang_str ,
2938 variant .str_set_x:N = \l__stex_notation_variant_str ,
2939 prec .str_set_x:N = \l__stex_notation_prec_str ,
2940 assoc .choices:nn =
2941 {bin,binl,binr,pre,conj,pwconj}
2942 {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2943 unknown .code:n = \str_set:Nx
2944 \l__stex_notation_variant_str \l_keys_key_str
2945 }
2946
2947 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2948 \str_clear:N \l_stex_symdecl_name_str
2949 \str_clear:N \l_stex_symdecl_args_str
2950 \str_clear:N \l_stex_symdecl_assoctype_str
2951 \str_clear:N \l_stex_symdecl_reorder_str
2952 \bool_set_false:N \l_stex_symdecl_local_bool
2953 \tl_clear:N \l_stex_symdecl_type_tl
2954 \tl_clear:N \l_stex_symdecl_definiens_tl
2955 % \str_clear:N \l__stex_notation_lang_str
2956 \str_clear:N \l__stex_notation_variant_str
2957 \str_clear:N \l__stex_notation_prec_str
2958 \tl_clear:N \l__stex_notation_op_tl
2959
2960 \keys_set:nn { stex / symdef } { #1 }
2961 }
2962
2963 \NewDocumentCommand \symdef { m O{} } {
2964 \__stex_notation_symdef_args:n { #2 }
2965 \bool_set_true:N \l_stex_symdecl_make_macro_bool
2966 \stex_symdecl_do:n { #1 }
2967 \tl_set:Nn \l_stex_notation_after_do_tl {
2968 \__stex_notation_final:
2969 \stex_smsmode_do:\ignorespacesandpars
2970 }
2971 \str_set:Nx \l_stex_get_symbol_uri_str {
2972 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2973 }
2974 \exp_args:Nx \stex_notation_do:nnnnn
2975 { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2976 { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2977 { \l__stex_notation_variant_str }
2978 { \l__stex_notation_prec_str }
2979 }
2980 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

28.3 Variables

```

2981 <@@=stex_variables>

```

```

2982
2983 \keys_define:nn { stex / vardef } {
2984   name      .str_set_x:N = \l__stex_variables_name_str ,
2985   args      .str_set_x:N = \l__stex_variables_args_str ,
2986   type      .tl_set:N    = \l__stex_variables_type_tl ,
2987   def       .tl_set:N    = \l__stex_variables_def_tl ,
2988   op        .tl_set:N    = \l__stex_variables_op_tl ,
2989   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2990   reorder   .str_set_x:N = \l__stex_variables_reorder_str ,
2991   assoc     .choices:nn  =
2992     {bin,binl,binr,pre,conj,pwconj}
2993     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2994   bind      .choices:nn  =
2995     {forall,exists}
2996     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2997 }
2998
2999 \cs_new_protected:Nn \__stex_variables_args:n {
3000   \str_clear:N \l__stex_variables_name_str
3001   \str_clear:N \l__stex_variables_args_str
3002   \str_clear:N \l__stex_variables_prec_str
3003   \str_clear:N \l__stex_variables_assoctype_str
3004   \str_clear:N \l__stex_variables_reorder_str
3005   \str_clear:N \l__stex_variables_bind_str
3006   \tl_clear:N \l__stex_variables_type_tl
3007   \tl_clear:N \l__stex_variables_def_tl
3008   \tl_clear:N \l__stex_variables_op_tl
3009
3010   \keys_set:nn { stex / vardef } { #1 }
3011 }
3012
3013 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
3014   \__stex_variables_args:n {#2}
3015   \str_if_empty:NT \l__stex_variables_name_str {
3016     \str_set:Nx \l__stex_variables_name_str { #1 }
3017   }
3018   \prop_clear:N \l_tmpa_prop
3019   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3020
3021   \int_zero:N \l_tmpb_int
3022   \bool_set_true:N \l_tmpa_bool
3023   \str_map_inline:Nn \l__stex_variables_args_str {
3024     \token_case_meaning:NnF ##1 {
3025       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3026       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3027       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3028       {\tl_to_str:n a} {
3029         \bool_set_false:N \l_tmpa_bool
3030         \int_incr:N \l_tmpb_int
3031       }
3032       {\tl_to_str:n B} {
3033         \bool_set_false:N \l_tmpa_bool
3034         \int_incr:N \l_tmpb_int
3035       }

```

```

3036   }{
3037     \msg_error:nxxx{stex}{error/wrongargs}{
3038       variable~\l__stex_variables_name_str
3039     }{##1}
3040   }
3041 }
3042 \bool_if:NTF \l_tmpa_bool {
3043   % possibly numeric
3044   \str_if_empty:NTF \l__stex_variables_args_str {
3045     \prop_put:Nnn \l_tmpa_prop { args } {}
3046     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3047   }{
3048     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3049     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3050     \str_clear:N \l_tmpa_str
3051     \int_step_inline:nn \l_tmpa_int {
3052       \str_put_right:Nn \l_tmpa_str i
3053     }
3054     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3055     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3056   }
3057 } {
3058   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3059   \prop_put:Nnx \l_tmpa_prop { arity }
3060   { \str_count:N \l__stex_variables_args_str }
3061 }
3062 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3063 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3064
3065 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
3066
3067 \tl_if_empty:NF \l__stex_variables_op_tl {
3068   \cs_set:cpx {
3069     stex_var_op_notation_\l__stex_variables_name_str _cs
3070   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3071 }
3072
3073 \tl_set:Nn \l_stex_notation_after_do_tl {
3074   \exp_args:Nne \use:nn {
3075     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3076     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3077   } {{
3078     \exp_after:wN \exp_after:wN \exp_after:wN
3079     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3080     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3081   }}
3082 \stex_if_do_html:T {
3083   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3084     \stex_annotate_invisible:nnn { precedence }
3085     { \l__stex_variables_prec_str }{}
3086     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l
3087     \stex_annotate_invisible:nnn{args}{ \l__stex_variables_args_str }{}
3088     \stex_annotate_invisible:nnn{macroname}{#1}{}
3089     \tl_if_empty:NF \l__stex_variables_def_tl {

```

```

3090     \stex_annotate_invisible:nnn{definiens}{}
3091     {${\l__stex_variables_def_tl$}
3092   }
3093   \str_if_empty:NF \l__stex_variables_assoctype_str {
3094     \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3095   }
3096   \str_if_empty:NF \l__stex_variables_reorder_str {
3097     \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3098   }
3099   \int_zero:N \l_tmpa_int
3100   \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3101   \tl_clear:N \l_tmpa_tl
3102   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3103     \int_incr:N \l_tmpa_int
3104     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3105     \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3106     \str_if_eq:VnTF \l_tmpb_str a {
3107       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3108         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3109         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3110       } }
3111     }{
3112       \str_if_eq:VnTF \l_tmpb_str B {
3113         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3114           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3115           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3116         } }
3117       }{
3118         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3119           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3120         } }
3121       }
3122     }
3123   }
3124   \stex_annotate_invisible:nnn { notationcomp }{}{
3125     \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3126     $ \exp_args:Nno \use:nn { \use:c {
3127       stex_var_notation_\l__stex_variables_name_str _cs
3128     } } { \l_tmpa_tl } $
3129   }
3130   \tl_if_empty:NF \l__stex_variables_op_tl {
3131     \stex_annotate_invisible:nnn { notationopcomp }{}{
3132       ${\l__stex_variables_op_tl$}
3133     }
3134   }
3135   }
3136   \str_if_empty:NF \l__stex_variables_bind_str {
3137     \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3138   }
3139   }\ignorespacesandpars
3140 }
3141
3142 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3143 }

```

```

3144
3145 \cs_new:Nn \_stex_reset:N {
3146   \tl_if_exist:NTF #1 {
3147     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3148   }{
3149     \let \exp_not:N #1 \exp_not:N \undefined
3150   }
3151 }
3152
3153 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3154   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3155   \exp_args:Nnx \use:nn {
3156     % TODO
3157     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3158       #2
3159     }
3160   }{
3161     \_stex_reset:N \varnot
3162     \_stex_reset:N \vartype
3163     \_stex_reset:N \vardefi
3164   }
3165 }
3166
3167 \NewDocumentCommand \vardef { s } {
3168   \IfBooleanTF#1 {
3169     \__stex_variables_do_complex:nn
3170   }{
3171     \__stex_variables_do_simple:nnn
3172   }
3173 }
3174
3175 \NewDocumentCommand \svar { 0{} m }{
3176   \tl_if_empty:nTF {#1}{
3177     \str_set:Nn \l_tmpa_str { #2 }
3178   }{
3179     \str_set:Nn \l_tmpa_str { #1 }
3180   }
3181   \_stex_term_omv:nn {
3182     var://\l_tmpa_str
3183   }{
3184     \exp_args:Nnx \use:nn {
3185       \def\comp{\_varcomp}
3186       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3187       \comp{ #2 }
3188     }{
3189       \_stex_reset:N \comp
3190       \_stex_reset:N \STEXInternalCurrentSymbolStr
3191     }
3192   }
3193 }
3194
3195
3196
3197 \keys_define:nn { stex / varseq } {

```

```

3198 name      .str_set_x:N = \l__stex_variables_name_str ,
3199 args      .int_set:N = \l__stex_variables_args_int ,
3200 type      .tl_set:N = \l__stex_variables_type_tl ,
3201 mid       .tl_set:N = \l__stex_variables_mid_tl ,
3202 bind      .choices:nn =
3203           {forall,exists}
3204           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3205 }
3206
3207 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3208   \str_clear:N \l__stex_variables_name_str
3209   \int_set:Nn \l__stex_variables_args_int 1
3210   \tl_clear:N \l__stex_variables_type_tl
3211   \str_clear:N \l__stex_variables_bind_str
3212
3213   \keys_set:nn { stex / varseq } { #1 }
3214 }
3215
3216 \NewDocumentCommand \varseq {m O{} m m m}{
3217   \__stex_variables_seq_args:n { #2 }
3218   \str_if_empty:NT \l__stex_variables_name_str {
3219     \str_set:Nx \l__stex_variables_name_str { #1 }
3220   }
3221   \prop_clear:N \l_tmpa_prop
3222   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3223
3224   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3225   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3226     \msg_error:nnxx{stex}{error/seqlength}
3227     {\int_use:N \l__stex_variables_args_int}
3228     {\seq_count:N \l_tmpa_seq}
3229   }
3230   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3231   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3232     \msg_error:nnxx{stex}{error/seqlength}
3233     {\int_use:N \l__stex_variables_args_int}
3234     {\seq_count:N \l_tmpb_seq}
3235   }
3236   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3237   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3238
3239   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3240   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3241
3242   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3243   \int_step_inline:nn \l__stex_variables_args_int {
3244     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3245   }
3246   \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3247   \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3248   \tl_if_empty:NF \l__stex_variables_mid_tl {
3249     \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3250     \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3251   }

```

```

3252 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3253 \int_step_inline:nn \l__stex_variables_args_int {
3254   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3255 }
3256 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3257 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3258
3259
3260 \prop_put:Nno \l_tmpa_prop { notation } \l_tmpa_tl
3261
3262 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3263
3264 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3265
3266 \int_step_inline:nn \l__stex_variables_args_int {
3267   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3268     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3269   }}
3270 }
3271
3272 \tl_set:Nx \l_tmpa_tl {
3273   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str }{}{0}{
3274     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3275   }
3276 }
3277
3278 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3279
3280 \exp_args:Nno \use:nn {
3281   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3282   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3283
3284   \stex_debug:nn{sequences}{New~Sequence:~
3285     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3286     \prop_to_keyval:N \l_tmpa_prop
3287   }
3288   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3289     \tl_if_empty:NF \l__stex_variables_type_tl {
3290       \stex_annotate:nnn {type}{\l__stex_variables_type_tl$}
3291     }
3292     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3293     \str_if_empty:NF \l__stex_variables_bind_str {
3294       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3295     }
3296     \stex_annotate:nnn{startindex}{\l__stex_variables_startindex}{\l__stex_variables_startindex$}
3297     \stex_annotate:nnn{endindex}{\l__stex_variables_endindex}{\l__stex_variables_endindex$}
3298
3299     \tl_clear:N \l_tmpa_tl
3300     \int_step_inline:nn \l__stex_variables_args_int {
3301       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3302         \stex_annotate:nnn{argmarker}{##1}{}
3303       } }
3304     }
3305     \stex_annotate_invisible:nnn { notationcomp }{}{

```



```

3306     \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3307     $ \exp_args:Nno \use:nn { \use:c {
3308         stex_varseq_\l__stex_variables_name_str _cs
3309     } } { \l_tmpa_tl } $
3310 }
3311 \stex_annotate_invisible:nnn { notationopcomp }{}{
3312     $ \prop_item:Nn \l_tmpa_prop { notation } $
3313 }
3314
3315 }}
3316
3317 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3318 \ignorespacesandpars
3319 }
3320
3321 \end{package}

```

Chapter 29

STEX -Terms Implementation

```
3322 <*package>
3323
3324 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3325
3326 <@@=stex_terms>
    Warnings and error messages
3327 \msg_new:nnn{stex}{error/nonotation}{
3328   Symbol~#1~invoked,~but~has~no~notation#2!
3329 }
3330 \msg_new:nnn{stex}{error/notationarg}{
3331   Error~in~parsing~notation~#1
3332 }
3333 \msg_new:nnn{stex}{error/noop}{
3334   Symbol~#1~has~no~operator~notation~for~notation~#2
3335 }
3336 \msg_new:nnn{stex}{error/notallowed}{
3337   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3338 }
3339 \msg_new:nnn{stex}{error/doubleargument}{
3340   Argument~#1~of~symbol~#2~already~assigned
3341 }
3342 \msg_new:nnn{stex}{error/overarity}{
3343   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3344 }
3345
```

29.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3346
3347
3348 \bool_new:N \l_stex_allow_semantic_bool
3349 \bool_set_true:N \l_stex_allow_semantic_bool
3350
```

```

3351 \cs_new_protected:Nn \stex_invoke_symbol:n {
3352   \ifvmode\indent\fi
3353   \bool_if:NTF \l_stex_allow_semantic_bool {
3354     \str_if_eq:eeF {
3355       \prop_item:cn {
3356         l_stex_symdecl_#1_prop
3357       }{ deprecate }
3358     }{}{
3359       \msg_warning:nxxx{stex}{warning/deprecated}{
3360         Symbol~#1
3361       }{
3362         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3363       }
3364     }
3365     \if_mode_math:
3366       \exp_after:wN \__stex_terms_invoke_math:n
3367     \else:
3368       \exp_after:wN \__stex_terms_invoke_text:n
3369     \fi: { #1 }
3370   }{
3371     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3372   }
3373 }
3374
3375 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3376   \peek_charcode_remove:NTF ! {
3377     \__stex_terms_invoke_op_custom:nn {#1}
3378   }{
3379     \__stex_terms_invoke_custom:nn {#1}
3380   }
3381 }
3382
3383 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3384   \peek_charcode_remove:NTF ! {
3385     % operator
3386     \peek_charcode_remove:NTF * {
3387       % custom op
3388       \__stex_terms_invoke_op_custom:nn {#1}
3389     }{
3390       % op notation
3391       \peek_charcode:NTF [ {
3392         \__stex_terms_invoke_op_notation:nw {#1}
3393       }{
3394         \__stex_terms_invoke_op_notation:nw {#1}[]
3395       }
3396     }
3397   }{
3398     \peek_charcode_remove:NTF * {
3399       \__stex_terms_invoke_custom:nn {#1}
3400       % custom
3401     }{
3402       % normal
3403       \peek_charcode:NTF [ {
3404         \__stex_terms_invoke_notation:nw {#1}

```

```

3405     }{
3406         \__stex_terms_invoke_notation:nw {#1}[]
3407     }
3408 }
3409 }
3410 }
3411
3412
3413 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3414     \exp_args:Nnx \use:nn {
3415         \def\comp{\_comp}
3416         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3417         \bool_set_false:N \l_stex_allow_semantic_bool
3418         \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3419             \comp{ #2 }
3420         }
3421     }{
3422         \stex_reset:N \comp
3423         \stex_reset:N \STEXInternalCurrentSymbolStr
3424         \bool_set_true:N \l_stex_allow_semantic_bool
3425     }
3426 }
3427
3428 \keys_define:nn { stex / terms } {
3429     % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3430     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3431     unknown .code:n      = \str_set:Nx
3432         \l_stex_notation_variant_str \l_keys_key_str
3433 }
3434
3435 \cs_new_protected:Nn \__stex_terms_args:n {
3436     % \str_clear:N \l_stex_notation_lang_str
3437     \str_clear:N \l_stex_notation_variant_str
3438
3439     \keys_set:nn { stex / terms } { #1 }
3440 }
3441
3442 \cs_new_protected:Nn \stex_find_notation:nn {
3443     \__stex_terms_args:n { #2 }
3444     \seq_if_empty:cTF {
3445         l_stex_symdecl_ #1 _notations
3446     } {
3447         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3448     } {
3449         \str_if_empty:NTF \l_stex_notation_variant_str {
3450             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3451         }{
3452             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3453                 \l_stex_notation_variant_str
3454             }{
3455                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3456             }{
3457                 \msg_error:nnxx{stex}{error/nonotation}{#1}{
3458                     ~\l_stex_notation_variant_str

```

```

3459     }
3460   }
3461 }
3462 }
3463 }
3464
3465 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3466   \exp_args:Nnx \use:nn {
3467     \def\comp{\_comp}
3468     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3469     \stex_find_notation:nn { #1 }{ #2 }
3470     \bool_set_false:N \l_stex_allow_semantic_bool
3471     \cs_if_exist:cTF {
3472       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3473     }{
3474       \_stex_term_oms:nnn { #1 }{
3475         #1 \c_hash_str \l_stex_notation_variant_str
3476       }{
3477         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3478       }
3479     }{
3480       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3481         \cs_if_exist:cTF {
3482           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3483         }{
3484           \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3485             \_stex_reset:N \comp
3486             \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3487             \_stex_reset:N \STEXInternalCurrentSymbolStr
3488             \bool_set_true:N \l_stex_allow_semantic_bool
3489           }
3490           \def\comp{\_comp}
3491           \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3492           \bool_set_false:N \l_stex_allow_semantic_bool
3493           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3494         }{
3495           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3496             ~\l_stex_notation_variant_str
3497           }
3498         }
3499       }{
3500         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3501       }
3502     }
3503   }{
3504     \_stex_reset:N \comp
3505     \_stex_reset:N \STEXInternalCurrentSymbolStr
3506     \bool_set_true:N \l_stex_allow_semantic_bool
3507   }
3508 }
3509
3510 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3511   \stex_find_notation:nn { #1 }{ #2 }
3512   \cs_if_exist:cTF {

```

```

3513     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3514   }{
3515     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3516       \_stex_reset:N \comp
3517       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3518       \_stex_reset:N \STEXInternalCurrentSymbolStr
3519       \bool_set_true:N \l_stex_allow_semantic_bool
3520     }
3521     \def\comp{\_comp}
3522     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3523     \bool_set_false:N \l_stex_allow_semantic_bool
3524     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3525   }{
3526     \msg_error:nnxx{stex}{error/nonotation}{#1}{
3527       ~\l_stex_notation_variant_str
3528     }
3529   }
3530 }
3531
3532 \prop_new:N \l__stex_terms_custom_args_prop
3533
3534 \cs_new_protected:Nn \__stex_terms_custom_comp:n { \bool_set_false:N \l_stex_allow_semantic_bool }
3535
3536 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3537   \exp_args:Nnx \use:nn {
3538     \def\comp{\_stex_terms_custom_comp:n}
3539     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3540     \prop_clear:N \l__stex_terms_custom_args_prop
3541     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3542     \prop_get:cnN {
3543       l_stex_symdecl_#1 _prop
3544     } { args } \l_tmpa_str
3545     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3546     \tl_set:Nn \arg { \_stex_terms_arg: }
3547     \str_if_empty:NTF \l_tmpa_str {
3548       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3549     }{
3550       \str_if_in:NnTF \l_tmpa_str b {
3551         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3552       }{
3553         \str_if_in:NnTF \l_tmpa_str B {
3554           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3555         }{
3556           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3557         }
3558       }
3559     }
3560     % TODO check that all arguments exist
3561   }{
3562     \_stex_reset:N \STEXInternalCurrentSymbolStr
3563     \_stex_reset:N \arg
3564     \_stex_reset:N \comp
3565     \_stex_reset:N \l__stex_terms_custom_args_prop
3566     %\bool_set_true:N \l_stex_allow_semantic_bool

```

```

3567 }
3568 }
3569
3570 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3571   \tl_if_empty:nTF {#2}{
3572     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3573     \bool_set_true:N \l_tmpa_bool
3574     \bool_do_while:Nn \l_tmpa_bool {
3575       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3576         \int_incr:N \l_tmpa_int
3577       }{
3578         \bool_set_false:N \l_tmpa_bool
3579       }
3580     }
3581   }{
3582     \int_set:Nn \l_tmpa_int { #2 }
3583   }
3584   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3585   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3586     \msg_error:nnxxx{stex}{error/overarity}
3587     {\int_use:N \l_tmpa_int}
3588     {\STEXInternalCurrentSymbolStr}
3589     {\str_count:N \l_tmpa_str}
3590   }
3591   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3592   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3593     \bool_lazy_any:nF {
3594       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3595       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3596     }{
3597       \msg_error:nnxx{stex}{error/doubleargument}
3598       {\int_use:N \l_tmpa_int}
3599       {\STEXInternalCurrentSymbolStr}
3600     }
3601   }
3602   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3603   \bool_if:NnTF \l_stex_allow_semantic_bool \use_i:nn {
3604     \bool_set_true:N \l_stex_allow_semantic_bool
3605     \use:nn
3606   }
3607   {
3608     \IfBooleanTF#1{
3609       \stex_annotate_invisible:n { %TODO
3610         \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3611       }
3612     }{ %TODO
3613       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3614     }
3615     {\bool_set_false:N \l_stex_allow_semantic_bool}
3616   }
3617
3618
3619 \cs_new_protected:Nn \_stex_term_arg:nn {
3620   \bool_set_true:N \l_stex_allow_semantic_bool

```

```

3621 \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3622 \bool_set_false:N \l_stex_allow_semantic_bool
3623 }
3624
3625 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
3626 \exp_args:Nnx \use:nn
3627 { \int_set:Nn \l__stex_terms_downprec { #2 }
3628 \stex_term_arg:nn { #1 }{ #3 }
3629 }
3630 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3631 }

```

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

`\STEXInternalTermMathAssocArgiiii`

```

3632 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4 {
3633 \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3634 \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#1}{#2}}
3635 \tl_if_empty:nTF { #3 }{
3636 \STEXInternalTermMathArgiii{#1}{#2}{#3}
3637 }{
3638 \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3639 \expandafter\if\expandafter\relax\noexpand#3
3640 \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3641 \else
3642 \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3643 \fi
3644 \l_tmpa_tl
3645 }{
3646 \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3647 }
3648 }
3649 }
3650
3651 \cs_new_protected:Npn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3652 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3653 \str_if_empty:NNTF \l_tmpa_str {
3654 \exp_args:Nx \cs_if_eq:NNTF {
3655 \tl_head:N #1
3656 } \stex_invoke_sequence:n {
3657 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3658 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3659 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3660 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3661 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3662 \exp_not:n{\exp_args:Nnx \use:nn} {
3663 \exp_not:n {
3664 \def\comp{\_varcomp}
3665 \str_set:Nn \STEXInternalCurrentSymbolStr
3666 } {varseq://\l_tmpa_str}
3667 \exp_not:n{ ##1 }
3668 }{
3669 \exp_not:n {
3670 \stex_reset:N \comp

```



```

3671         \_stex_reset:N \STEXInternalCurrentSymbolStr
3672     }
3673 }
3674 }}}
3675 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3676 \seq_reverse:N \l_tmpa_seq
3677 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3678 \seq_map_inline:Nn \l_tmpa_seq {
3679     \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
3680         \exp_args:Nno
3681         \l_tmpa_cs { ##1 } \l_tmpa_tl
3682     }
3683 }
3684 \tl_set:Nx \l_tmpa_tl {
3685     \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3686         \exp_args:No \exp_not:n \l_tmpa_tl
3687     }
3688 }
3689 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3690 }{
3691     \_stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3692 }
3693 } {
3694     \_stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3695 }
3696 }
3697 }
3698
3699 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nn {
3700     \clist_set:Nn \l_tmpa_clist{ #2 }
3701     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3702         \tl_set:Nn \l_tmpa_tl { \_stex_term_arg:nn{A#1}{ #2 } }
3703     }{
3704         \clist_reverse:N \l_tmpa_clist
3705         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3706         \tl_set:Nx \l_tmpa_tl { \_stex_term_arg:nn{A#1}{
3707             \exp_args:No \exp_not:n \l_tmpa_tl
3708         }}
3709         \clist_map_inline:Nn \l_tmpa_clist {
3710             \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
3711                 \exp_args:Nno
3712                 \l_tmpa_cs { \_stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3713             }
3714         }
3715     }
3716     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3717 }

```

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

29.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3718 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3719 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3720 \int_new:N \l__stex_terms_downprec
3721 \int_set_eq:NN \l__stex_terms_downprec \infprec

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

Bracketing:

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
3722 \tl_set:Nn \l__stex_terms_left_bracket_str (
3723 \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
3724 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3725   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3726     \bool_set_false:N \l__stex_terms_brackets_done_bool
3727     #2
3728   } {
3729     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3730       \bool_if:NTF \l__stex_inarray_bool { #2 }{
3731         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3732         \dobrackets { #2 }
3733       }
3734     }{ #2 }
3735   }
3736 }

(End definition for \__stex_terms_maybe_brackets:nn.)

\dobrackets
3737 \bool_new:N \l__stex_terms_brackets_done_bool
3738 %\RequirePackage{scalerel}
3739 \cs_new_protected:Npn \dobrackets #1 {
3740   %\ThisStyle{\if D\m@switch
3741   %   \exp_args:Nnx \use:nn
3742   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3743   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3744   % \else
3745   \exp_args:Nnx \use:nn
3746   {
3747     \bool_set_true:N \l__stex_terms_brackets_done_bool
3748     \int_set:Nn \l__stex_terms_downprec \infprec
3749     \l__stex_terms_left_bracket_str
3750     #1
3751   }
3752   {
3753     \bool_set_false:N \l__stex_terms_brackets_done_bool
3754     \l__stex_terms_right_bracket_str
3755     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3756   }
3757   %\fi}
3758 }

```

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

`\withbrackets`

```

3759 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3760   \exp_args:Nnx \use:nn
3761   {
3762     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3763     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3764     #3
3765   }
3766   {
3767     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3768     {\l__stex_terms_left_bracket_str}
3769     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3770     {\l__stex_terms_right_bracket_str}
3771   }
3772 }

```

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

`\STEXinvisible`

```

3773 \cs_new_protected:Npn \STEXinvisible #1 {
3774   \stex_annotate_invisible:n { #1 }
3775 }

```

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

OMDoc terms:

`\STEXInternalTermMathOMSiiii`

```

3776 \cs_new_protected:Nn \_stex_term_oms:nnn {
3777   \stex_annotate:nnn{ OMID }{ #2 }{
3778     #3
3779   }
3780 }
3781
3782 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
3783   \__stex_terms_maybe_brackets:nn { #3 }{
3784     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3785   }
3786 }

```

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

`_stex_term_math_omv:nn`

```

3787 \cs_new_protected:Nn \_stex_term_omv:nn {
3788   \stex_annotate:nnn{ OMV }{ #1 }{
3789     #2
3790   }
3791 }

```

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

`\STEXInternalTermMathOMAIiii`

```

3792 \cs_new_protected:Nn \stex_term_oma:nnn {
3793   \stex_annotate:nnn{ OMA }{ #2 }{
3794     #3
3795   }
3796 }
3797
3798 \cs_new_protected:Npn \STEXInternalTermMathOMAIiii #1#2#3#4 {
3799   \__stex_terms_maybe_brackets:nn { #3 }{
3800     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3801   }
3802 }

```

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

`\STEXInternalTermMathOMBiii`

```

3803 \cs_new_protected:Nn \stex_term_ombind:nnn {
3804   \stex_annotate:nnn{ OMBIND }{ #2 }{
3805     #3
3806   }
3807 }
3808
3809 \cs_new_protected:Npn \STEXInternalTermMathOMBiii #1#2#3#4 {
3810   \__stex_terms_maybe_brackets:nn { #3 }{
3811     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3812   }
3813 }

```

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

`\symref`

`\symname`

```

3814 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3815
3816 \keys_define:nn { stex / symname } {
3817   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3818   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3819   root     .tl_set_x:N      = \l__stex_terms_root_tl
3820 }
3821
3822 \cs_new_protected:Nn \stex_symname_args:n {
3823   \tl_clear:N \l__stex_terms_post_tl
3824   \tl_clear:N \l__stex_terms_pre_tl
3825   \tl_clear:N \l__stex_terms_root_str
3826   \keys_set:nn { stex / symname } { #1 }
3827 }
3828
3829 \NewDocumentCommand \symref { m m }{
3830   \let\compemph_uri_prev:\compemph@uri
3831   \let\compemph@uri\symrefemph@uri
3832   \STEXsymbol{#1}!{ #2 }
3833   \let\compemph@uri\compemph_uri_prev:
3834 }
3835
3836 \NewDocumentCommand \synonym { 0{ } m m }{

```

```

3837 \stex_symname_args:n { #1 }
3838 \let\compemph_uri_prev:\compemph@uri
3839 \let\compemph@uri\symrefemph@uri
3840 % TODO
3841 \STEXsymbol{#2}!\{ \l__stex_terms_pre_tl #3 \l__stex_terms_post_tl }
3842 \let\compemph@uri\compemph_uri_prev:
3843 }
3844
3845 \NewDocumentCommand \symname { 0{ } m }{
3846 \stex_symname_args:n { #1 }
3847 \stex_get_symbol:n { #2 }
3848 \str_set:Nx \l_tmpa_str {
3849 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3850 }
3851 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3852
3853 \let\compemph_uri_prev:\compemph@uri
3854 \let\compemph@uri\symrefemph@uri
3855 \exp_args:NNx \use:nn
3856 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3857 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3858 } }
3859 \let\compemph@uri\compemph_uri_prev:
3860 }
3861
3862 \NewDocumentCommand \Symname { 0{ } m }{
3863 \stex_symname_args:n { #1 }
3864 \stex_get_symbol:n { #2 }
3865 \str_set:Nx \l_tmpa_str {
3866 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3867 }
3868 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3869 \let\compemph_uri_prev:\compemph@uri
3870 \let\compemph@uri\symrefemph@uri
3871 \exp_args:NNx \use:nn
3872 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3873 \exp_after:wN \stex_capitalize:n \l_tmpa_str
3874 \l__stex_terms_post_tl
3875 } }
3876 \let\compemph@uri\compemph_uri_prev:
3877 }

```

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

29.3 Notation Components

```

3878 <@@=stex_notationcomps>

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

```

```

3879 \cs_new_protected:Npn \_comp #1 {
3880 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3881 \stex_html_backend:TF {
3882 \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3883 }{

```

```

3884     \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3885   }
3886 }
3887 }
3888
3889 \cs_new_protected:Npn \_varcomp #1 {
3890   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3891     \stex_html_backend:TF {
3892       \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3893     }{
3894       \exp_args:Nnx \varempemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3895     }
3896   }
3897 }
3898
3899 \def\comp{\_comp}
3900
3901 \cs_new_protected:Npn \compemph@uri #1 #2 {
3902   \compemph{ #1 }
3903 }
3904
3905
3906 \cs_new_protected:Npn \compemph #1 {
3907   #1
3908 }
3909
3910 \cs_new_protected:Npn \defemph@uri #1 #2 {
3911   \defemph{#1}
3912 }
3913
3914 \cs_new_protected:Npn \defemph #1 {
3915   \textbf{#1}
3916 }
3917
3918 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3919   \symrefemph{#1}
3920 }
3921
3922 \cs_new_protected:Npn \symrefemph #1 {
3923   \emph{#1}
3924 }
3925
3926 \cs_new_protected:Npn \varempemph@uri #1 #2 {
3927   \varempemph{#1}
3928 }
3929
3930 \cs_new_protected:Npn \varempemph #1 {
3931   #1
3932 }

```

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

\ellipses

```

3933 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3934 \bool_new:N \l_stex_inparray_bool
\parrayline 3935 \bool_set_false:N \l_stex_inparray_bool
\parraylineh 3936 \NewDocumentCommand \parray { m m } {
\parraycell 3937 \begin{group}
3938 \bool_set_true:N \l_stex_inparray_bool
3939 \begin{array}{#1}
3940 #2
3941 \end{array}
3942 \endgroup
3943 }
3944
3945 \NewDocumentCommand \prmatrix { m } {
3946 \begin{group}
3947 \bool_set_true:N \l_stex_inparray_bool
3948 \begin{matrix}
3949 #1
3950 \end{matrix}
3951 \endgroup
3952 }
3953
3954 \def \maybepline {
3955 \bool_if:NT \l_stex_inparray_bool {\hline}
3956 }
3957
3958 \def \parrayline #1 #2 {
3959 #1 #2 \bool_if:NT \l_stex_inparray_bool {\}
3960 }
3961
3962 \def \pmrow #1 { \parrayline{}{ #1 } }
3963
3964 \def \parraylineh #1 #2 {
3965 #1 #2 \bool_if:NT \l_stex_inparray_bool {\hline}
3966 }
3967
3968 \def \parraycell #1 {
3969 #1 \bool_if:NT \l_stex_inparray_bool {&}
3970 }

```

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

29.4 Variables

3971 `<@@=stex_variables>`

`\stex_invoke_variable:n` Invokes a variable

```

3972 \cs_new_protected:Nn \stex_invoke_variable:n {
3973 \if_mode_math:
3974 \exp_after:wN \__stex_variables_invoke_math:n
3975 \else:
3976 \exp_after:wN \__stex_variables_invoke_text:n

```

```

3977 \fi: {#1}
3978 }
3979
3980 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3981 \peek_charcode_remove:NTF ! {
3982 \__stex_variables_invoke_op_custom:nn {#1}
3983 }{
3984 \__stex_variables_invoke_custom:nn {#1}
3985 }
3986 }
3987
3988
3989 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3990 \peek_charcode_remove:NTF ! {
3991 \peek_charcode_remove:NTF ! {
3992 \peek_charcode:NTF [ {
3993 % TODO throw error
3994 }{
3995 \__stex_variables_invoke_op_custom:nn
3996 }
3997 }{
3998 \__stex_variables_invoke_op:n { #1 }
3999 }
4000 }{
4001 \peek_charcode_remove:NTF * {
4002 \__stex_variables_invoke_custom:nn { #1 }
4003 }{
4004 \__stex_variables_invoke_math_ii:n { #1 }
4005 }
4006 }
4007 }
4008
4009 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4010 \exp_args:Nnx \use:nn {
4011 \def\comp{\_varcomp}
4012 \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4013 \bool_set_false:N \l_stex_allow_semantic_bool
4014 \stex_term_omv:nn {var://#1}{
4015 \comp{ #2 }
4016 }
4017 }{
4018 \stex_reset:N \comp
4019 \stex_reset:N \STEXInternalCurrentSymbolStr
4020 \bool_set_true:N \l_stex_allow_semantic_bool
4021 }
4022 }
4023
4024 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4025 \cs_if_exist:cTF {
4026 stex_var_op_notation_ #1 _cs
4027 }{
4028 \exp_args:Nnx \use:nn {
4029 \def\comp{\_varcomp}
4030 \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }

```



```

4031     \stex_term_omv:nn { var://#1 }{
4032       \use:c{stex_var_op_notation_ #1 _cs }
4033     }
4034   }{
4035     \stex_reset:N \comp
4036     \stex_reset:N \STEXInternalCurrentSymbolStr
4037   }
4038 }{
4039   \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
4040     \__stex_variables_invoke_math_ii:n {#1}
4041   }{
4042     \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
4043   }
4044 }
4045 }
4046
4047 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4048   \cs_if_exist:cTF {
4049     stex_var_notation_#1_cs
4050   }{
4051     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4052       \stex_reset:N \comp
4053       \stex_reset:N \STEXInternalSymbolAfterInvokationTL
4054       \stex_reset:N \STEXInternalCurrentSymbolStr
4055       \bool_set_true:N \l_stex_allow_semantic_bool
4056     }
4057     \def\comp{\_varcomp}
4058     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4059     \bool_set_false:N \l_stex_allow_semantic_bool
4060     \use:c{stex_var_notation_#1_cs}
4061   }{
4062     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
4063   }
4064 }
4065
4066 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4067   \exp_args:Nnx \use:nn {
4068     \def\comp{\_varcomp}
4069     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4070     \prop_clear:N \l__stex_terms_custom_args_prop
4071     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4072     \prop_get:cnN {
4073       l_stex_variable_#1_prop
4074     }{ args } \l_tmpa_str
4075     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4076     \tl_set:Nn \arg { \__stex_terms_arg: }
4077     \str_if_empty:NTF \l_tmpa_str {
4078       \stex_term_omv:nn {var://#1}{\ignorespaces#2}
4079     }{
4080       \str_if_in:NnTF \l_tmpa_str b {
4081         \stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4082       }{
4083         \str_if_in:NnTF \l_tmpa_str B {
4084           \stex_term_ombind:nnn {var://#1}{\ignorespaces#2}

```

```

4085     }{
4086       \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4087     }
4088   }
4089 }
4090 % TODO check that all arguments exist
4091 }{
4092   \_stex_reset:N \STEXInternalCurrentSymbolStr
4093   \_stex_reset:N \arg
4094   \_stex_reset:N \comp
4095   \_stex_reset:N \l__stex_terms_custom_args_prop
4096   %\bool_set_true:N \l_stex_allow_semantic_bool
4097 }
4098 }

```

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

29.5 Sequences

```

4099 <@@=stex_sequences>
4100
4101 \cs_new_protected:Nn \stex_invoke_sequence:n {
4102   \peek_charcode_remove:NTF ! {
4103     \_stex_term_omv:nn {varseq://#1}{
4104       \exp_args:Nnx \use:nn {
4105         \def\comp{\_varcomp}
4106         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4107         \prop_item:cn{stex_varseq_#1_prop}{notation}
4108       }{
4109         \_stex_reset:N \comp
4110         \_stex_reset:N \STEXInternalCurrentSymbolStr
4111       }
4112     }
4113   }{
4114     \bool_set_false:N \l_stex_allow_semantic_bool
4115     \def\comp{\_varcomp}
4116     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4117     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4118       \_stex_reset:N \comp
4119       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4120       \_stex_reset:N \STEXInternalCurrentSymbolStr
4121       \bool_set_true:N \l_stex_allow_semantic_bool
4122     }
4123     \use:c { stex_varseq_#1_cs }
4124   }
4125 }
4126 </package>

```

Chapter 30

STEX -Structural Features Implementation

```
4127 ⟨*package⟩
4128
4129 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4130
4131 Warnings and error messages
4132 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4133   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4134 }
4135 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4136   Symbol~#1~not~assigned~in~interpretmodule~#2
4137 }
4138 \msg_new:nnn{stex}{error/unknownstructure}{
4139   No~structure~#1~found!
4140 }
4141
4142 \msg_new:nnn{stex}{error/unknownfield}{
4143   No~field~#1~in~instance~#2~found!\#3
4144 }
4145
4146 \msg_new:nnn{stex}{error/keyval}{
4147   Invalid~key=value~pair~#1
4148 }
4149 \msg_new:nnn{stex}{error/instantiate/missing}{
4150   Assignments~missing~in~instantiate:~#1
4151 }
4152 \msg_new:nnn{stex}{error/incompatible}{
4153   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4154 }
4155
```

30.1 Imports with modification

```

4156 <@@=stex_copymodule>
4157 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4158   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4159     \tl_set:Nn \l_tmpa_tl { #1 }
4160     \__stex_copymodule_get_symbol_from_cs:
4161   }{
4162     % argument is a string
4163     % is it a command name?
4164     \cs_if_exist:cTF { #1 }{
4165       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4166       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4167       \str_if_empty:NNTF \l_tmpa_str {
4168         \exp_args:Nx \cs_if_eq:NNTF {
4169           \tl_head:N \l_tmpa_tl
4170         } \stex_invoke_symbol:n {
4171           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4172         }{
4173           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4174         }
4175       } {
4176         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4177       }
4178     }{
4179       % argument is not a command name
4180       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4181       % \l_stex_all_symbols_seq
4182     }
4183   }
4184 }
4185
4186 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4187   \str_set:Nn \l_tmpa_str { #1 }
4188   \bool_set_false:N \l_tmpa_bool
4189   \bool_if:NF \l_tmpa_bool {
4190     \tl_set:Nn \l_tmpa_tl {
4191       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4192     }
4193     \str_set:Nn \l_tmpa_str { #1 }
4194     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4195     \seq_map_inline:Nn #2 {
4196       \str_set:Nn \l_tmpb_str { ##1 }
4197       \str_if_eq:eeT { \l_tmpa_str } {
4198         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4199       } {
4200         \seq_map_break:n {
4201           \tl_set:Nn \l_tmpa_tl {
4202             \str_set:Nn \l_stex_get_symbol_uri_str {
4203               ##1
4204             }
4205           }
4206         }
4207       }

```

```

4208     }
4209     \l_tmpa_tl
4210   }
4211 }
4212
4213 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4214   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4215     { \tl_tail:N \l_tmpa_tl }
4216   \tl_if_single:NTF \l_tmpa_tl {
4217     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4218       \exp_after:wN \str_set:Nn \exp_after:wN
4219         \l_stex_get_symbol_uri_str \l_tmpa_tl
4220       \__stex_copymodule_get_symbol_check:n { #1 }
4221     }{
4222       % TODO
4223       % tail is not a single group
4224     }
4225   }{
4226     % TODO
4227     % tail is not a single group
4228   }
4229 }
4230
4231 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4232   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4233     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4234       :~\seq_use:Nn #1 {,~}
4235     }
4236   }
4237 }
4238
4239 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4240   % import module
4241   \stex_import_module_uri:nn { #1 } { #2 }
4242   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4243   \stex_import_require_module:nnnn
4244     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4245     { \l_stex_import_path_str } { \l_stex_import_name_str }
4246
4247   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4248   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4249
4250   % fields
4251   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4252   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4253     \seq_map_inline:cn {c_stex_module_##1_constants}{
4254       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4255         ##1 ? ####1
4256       }
4257     }
4258   }
4259
4260   % setup prop
4261   \seq_clear:N \l_tmpa_seq

```

```

4262 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4263   name      = \l_stex_current_copymodule_name_str ,
4264   module    = \l_stex_current_module_str ,
4265   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4266   includes  = \l_tmpa_seq %,
4267 % fields    = \l_tmpa_seq
4268 }
4269 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4270   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4271 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4272 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4273
4274 \stex_if_do_html:T {
4275   \begin{stex_annotate_env} {#4} {
4276     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4277   }
4278   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4279 }
4280 }
4281
4282 \cs_new_protected:Nn \stex_copymodule_end:n {
4283   % apply to every field
4284   \def \l_tmpa_cs ##1 ##2 {#1}
4285
4286   \tl_clear:N \__stex_copymodule_module_tl
4287   \tl_clear:N \__stex_copymodule_exec_tl
4288
4289   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4290   \seq_clear:N \__stex_copymodule_fields_seq
4291
4292   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4293     \seq_map_inline:cn {c_stex_module_##1_constants}{
4294
4295       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4296       \l_tmpa_cs{##1}{####1}
4297
4298       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4299         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4300         \stex_if_do_html:T {
4301           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4302             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4303           }
4304         }
4305       }{
4306         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4307       }
4308
4309       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4310       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4311       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4312
4313       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4314         \stex_if_do_html:T {
4315           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4316         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4317     }
4318 }
4319 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4320 }
4321
4322 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4323 \tl_put_right:Nx \__stex_copymodule_module_tl {
4324     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4325     \prop_set_from_keyval:cn {
4326         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4327     }{
4328         \prop_to_keyval:N \l_tmpa_prop
4329     }
4330 }
4331
4332 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4333     \stex_if_do_html:T {
4334         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4335             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4336         }
4337     }
4338     \tl_put_right:Nx \__stex_copymodule_module_tl {
4339         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4340             \stex_invoke_symbol:n {
4341                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4342             }
4343         }
4344     }
4345 }
4346
4347 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4348
4349 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4350     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4351 }
4352
4353 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4354     \stex_if_do_html:TF{
4355         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4356     }{
4357         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4358     }
4359 }
4360 }
4361 }
4362
4363
4364 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4365 \tl_put_left:Nx \__stex_copymodule_module_tl {
4366     \prop_set_from_keyval:cn {
4367         l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4368     }{
4369         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4370   }
4371 }
4372
4373 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4374   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4375 }
4376
4377 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4378 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4379 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4380
4381 \__stex_copymodule_exec_tl
4382 \stex_if_do_html:T {
4383   \end{stex_annotate_env}
4384 }
4385 }
4386
4387 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4388   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4389   \stex_deactivate_macro:Nn \symdecl {module~environments}
4390   \stex_deactivate_macro:Nn \symdef {module~environments}
4391   \stex_deactivate_macro:Nn \notation {module~environments}
4392   \stex_reactivate_macro:N \assign
4393   \stex_reactivate_macro:N \renamedekl
4394   \stex_reactivate_macro:N \donotcopy
4395   \stex_smsmode_do:
4396 }{
4397   \stex_copymodule_end:n {}
4398 }
4399
4400 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4401   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4402   \stex_deactivate_macro:Nn \symdecl {module~environments}
4403   \stex_deactivate_macro:Nn \symdef {module~environments}
4404   \stex_deactivate_macro:Nn \notation {module~environments}
4405   \stex_reactivate_macro:N \assign
4406   \stex_reactivate_macro:N \renamedekl
4407   \stex_reactivate_macro:N \donotcopy
4408   \stex_smsmode_do:
4409 }{
4410   \stex_copymodule_end:n {
4411     \tl_if_exist:cF {
4412       l__stex_copymodule_copymodule_##1?##2_def_tl
4413     }{
4414       \str_if_eq:eeF {
4415         \prop_item:cn{
4416           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4417       }{ true }{
4418         \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4419           ##1?##2
4420         }{\l_stex_current_copymodule_name_str}
4421       }
4422     }
4423   }

```



```

4424 }
4425
4426 \iffalse \begin{stex_annotate_env} \fi
4427 \NewDocumentEnvironment {realization} { 0 } { m } {
4428   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4429   \stex_deactivate_macro:Nn \symdecl {module~environments}
4430   \stex_deactivate_macro:Nn \symdef {module~environments}
4431   \stex_deactivate_macro:Nn \notation {module~environments}
4432   \stex_reactivate_macro:N \donotcopy
4433   \stex_reactivate_macro:N \assign
4434   \stex_smsmode_do:
4435 } {
4436   \stex_import_module_uri:nn { #1 } { #2 }
4437   \tl_clear:N \__stex_copymodule_exec_tl
4438   \tl_set:Nx \__stex_copymodule_module_tl {
4439     \stex_import_require_module:nnnn
4440     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4441     { \l_stex_import_path_str } { \l_stex_import_name_str }
4442   }
4443
4444   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4445     \seq_map_inline:cn {c_stex_module_##1_constants}{
4446       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4447       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4448         \stex_if_do_html:T {
4449           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4450             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4451               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4452             }
4453           }
4454         }
4455         \tl_put_right:Nx \__stex_copymodule_module_tl {
4456           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4457         }
4458       }
4459     }
4460
4461     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4462
4463     \__stex_copymodule_exec_tl
4464     \stex_if_do_html:T {\end{stex_annotate_env}}
4465   }
4466
4467 \NewDocumentCommand \donotcopy { m } {
4468   \str_clear:N \l_stex_import_name_str
4469   \str_set:Nn \l_tmpa_str { #1 }
4470   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4471   \seq_map_inline:Nn \l_stex_all_modules_seq {
4472     \str_set:Nn \l_tmpb_str { ##1 }
4473     \str_if_eq:eeT { \l_tmpa_str } {
4474       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4475     } {
4476       \seq_map_break:n {
4477         \stex_if_do_html:T {

```

```

4478         \stex_if_smsmode:F {
4479             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4480                 \stex_annotate:nnn{domain}{##1}{}}
4481         }
4482     }
4483 }
4484 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4485 }
4486 }
4487 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4488     \str_set:Nn \l_tmpb_str { #####1 }
4489     \str_if_eq:eeT { \l_tmpa_str } {
4490         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4491     } {
4492         \seq_map_break:n {\seq_map_break:n {
4493             \stex_if_do_html:T {
4494                 \stex_if_smsmode:F {
4495                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4496                         \stex_annotate:nnn{domain}{
4497                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4498                         }{}
4499                     }
4500                 }
4501             }
4502             \str_set:Nx \l_stex_import_name_str {
4503                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4504             }
4505         }}
4506     }
4507 }
4508 }
4509 \str_if_empty:NTF \l_stex_import_name_str {
4510     % TODO throw error
4511 }{
4512     \stex_collect_imports:n {\l_stex_import_name_str }
4513     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4514         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4515         \seq_map_inline:cn {c_stex_module_###1_constants}{
4516             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4517             \bool_lazy_any:nT {
4518                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4519                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4520                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4521             }{
4522                 % TODO throw error
4523             }
4524         }
4525     }
4526     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4527     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4528     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4529 }
4530 \stex_smsmode_do:
4531 }

```

```

4532
4533 \NewDocumentCommand \assign { m m }{
4534   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4535   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4536   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4537   \stex_smsmode_do:
4538 }
4539
4540 \keys_define:nn { stex / renamedecl } {
4541   name          .str_set_x:N = \l_stex_renamedecl_name_str
4542 }
4543 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4544   \str_clear:N \l_stex_renamedecl_name_str
4545   \keys_set:nn { stex / renamedecl } { #1 }
4546 }
4547
4548 \NewDocumentCommand \renamedecl { O{} m m }{
4549   \__stex_copymodule_renamedecl_args:n { #1 }
4550   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4551   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4552   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4553   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4554     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4555       \l_stex_get_symbol_uri_str
4556     } }
4557   } {
4558     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4559     \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4560     \prop_set_eq:cc {l_stex_symdecl_
4561       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4562     }_prop
4563     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4564     \seq_set_eq:cc {l_stex_symdecl_
4565       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4566     }_notations
4567     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4568     \prop_put:cnx {l_stex_symdecl_
4569       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4570     }_prop
4571     }{ name }{ \l_stex_renamedecl_name_str }
4572     \prop_put:cnx {l_stex_symdecl_
4573       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4574     }_prop
4575     }{ module }{ \l_stex_current_module_str }
4576     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4577       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4578     }
4579     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4580       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4581     } }
4582   }
4583   \stex_smsmode_do:
4584 }
4585

```

```

4586 \stex_deactivate_macro:Nn \assign {copymodules}
4587 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4588 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4589
4590

```

30.2 The feature environment

structural@feature

```

4591 <@@=stex_features>
4592
4593 \NewDocumentEnvironment{structural_feature_module}{m m m}{
4594   \stex_if_in_module:F {
4595     \msg_set:nnn{stex}{error/nomodule}{
4596       Structural~Feature~has~to~occur~in~a~module:\\
4597       Feature~#2~of~type~#1\\
4598       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4599     }
4600     \msg_error:nn{stex}{error/nomodule}
4601   }
4602
4603   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4604
4605   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4606
4607   \stex_if_do_html:T {
4608     \begin{stex_annotate_env}{feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4609     \stex_annotate_invisible:nnn{header}{}{ #3 }
4610   }
4611 }{
4612   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4613   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4614   \stex_debug:nn{features}{
4615     Feature: \l_stex_last_feature_str
4616   }
4617   \stex_if_do_html:T {
4618     \end{stex_annotate_env}
4619   }
4620 }

```

30.3 Structure

structure

```

4621 <@@=stex_structures>
4622 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4623   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4624     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4625   }
4626   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4627   {#1}{#2}
4628 }
4629

```

```

4630 \keys_define:nn { stex / features / structure } {
4631   name          .str_set_x:N = \l__stex_structures_name_str ,
4632 }
4633
4634 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4635   \str_clear:N \l__stex_structures_name_str
4636   \keys_set:nn { stex / features / structure } { #1 }
4637 }
4638
4639 \NewDocumentEnvironment{mathstructure}{m O{}}{
4640   \__stex_structures_structure_args:n { #2 }
4641   \str_if_empty:NT \l__stex_structures_name_str {
4642     \str_set:Nx \l__stex_structures_name_str { #1 }
4643   }
4644   \stex_suppress_html:n {
4645     \bool_set_true:N \l_stex_symdecl_make_macro_bool
4646     \exp_args:Nx \stex_symdecl_do:nn {
4647       name = \l__stex_structures_name_str ,
4648       def = {\STEXsymbol{module-type}}{
4649         \STEXInternalTermMathOMSiiii {
4650           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4651             { ns } ?
4652           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4653             { name } / \l__stex_structures_name_str - structure
4654         }{}{0}{}
4655       }}
4656     }{ #1 }
4657   }
4658   \exp_args:Nnnx
4659   \begin{structural_feature_module}{ structure }
4660     { \l__stex_structures_name_str }{}
4661   \stex_smsmode_do:
4662 }{
4663   \end{structural_feature_module}
4664   \_stex_reset_up_to_module:n \l_stex_last_feature_str
4665   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4666   \seq_clear:N \l_tmpa_seq
4667   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4668     \seq_map_inline:cn{c_stex_module_##1_constants}{
4669       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4670     }
4671   }
4672   \exp_args:Nnno
4673   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4674   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4675   \stex_add_structure_to_current_module:nn
4676     \l__stex_structures_name_str
4677     \l_stex_last_feature_str
4678
4679   \stex_execute_in_module:x {
4680     \tl_set:cn { #1 }{
4681       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4682     }
4683   }

```

```

4684 }
4685
4686 \cs_new:Nn \stex_invoke_structure:nn {
4687   \stex_invoke_symbol:n { #1?#2 }
4688 }
4689
4690 \cs_new_protected:Nn \stex_get_structure:n {
4691   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4692     \tl_set:Nn \l_tmpa_tl { #1 }
4693     \__stex_structures_get_from_cs:
4694   }{
4695     \cs_if_exist:cTF { #1 }{
4696       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4697       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4698       \str_if_empty:NTF \l_tmpa_str {
4699         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4700           \__stex_structures_get_from_cs:
4701         }{
4702           \__stex_structures_get_from_string:n { #1 }
4703         }
4704       }{
4705         \__stex_structures_get_from_string:n { #1 }
4706       }
4707     }{
4708       \__stex_structures_get_from_string:n { #1 }
4709     }
4710   }
4711 }
4712
4713 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4714   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4715     { \tl_tail:N \l_tmpa_tl }
4716   \str_set:Nx \l_tmpa_str {
4717     \exp_after:wN \use_i:nn \l_tmpa_tl
4718   }
4719   \str_set:Nx \l_tmpb_str {
4720     \exp_after:wN \use_ii:nn \l_tmpa_tl
4721   }
4722   \str_set:Nx \l_stex_get_structure_str {
4723     \l_tmpa_str ? \l_tmpb_str
4724   }
4725   \str_set:Nx \l_stex_get_structure_module_str {
4726     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4727   }
4728 }
4729
4730 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4731   \tl_set:Nn \l_tmpa_tl {
4732     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4733   }
4734   \str_set:Nn \l_tmpa_str { #1 }
4735   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4736
4737   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

4738 \prop_if_exist:cT {c_stex_module_##1_structures} {
4739 \prop_map_inline:cn {c_stex_module_##1_structures} {
4740 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4741 \prop_map_break:n{\seq_map_break:n{
4742 \tl_set:Nn \l_tmpa_tl {
4743 \str_set:Nn \l_stex_get_structure_str {##1?####1}
4744 \str_set:Nn \l_stex_get_structure_module_str {####2}
4745 }
4746 }}
4747 }
4748 }
4749 }
4750 }
4751 \l_tmpa_tl
4752 }

```

\instantiate

```

4753
4754 \keys_define:nn { stex / instantiate } {
4755 name .str_set_x:N = \l__stex_structures_name_str
4756 }
4757 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4758 \str_clear:N \l__stex_structures_name_str
4759 \keys_set:nn { stex / instantiate } { #1 }
4760 }
4761
4762 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4763 \beginingroup
4764 \stex_get_structure:n {#3}
4765 \__stex_structures_instantiate_args:n { #2 }
4766 \str_if_empty:NT \l__stex_structures_name_str {
4767 \str_set:Nn \l__stex_structures_name_str { #1 }
4768 }
4769 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4770 \seq_clear:N \l__stex_structures_fields_seq
4771 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4772 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4773 \seq_map_inline:cn {c_stex_module_##1_constants}{
4774 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4775 }
4776 }
4777
4778 \tl_if_empty:nF{#5}{
4779 \seq_set_split:Nnn \l_tmpa_seq , {#5}
4780 \prop_clear:N \l_tmpa_prop
4781 \seq_map_inline:Nn \l_tmpa_seq {
4782 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4783 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4784 \msg_error:nnn{stex}{error/keyval}{##1}
4785 }
4786 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4787 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4788 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4789 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```

```

4790     \exp_args:Nxx \str_if_eq:nnF
4791     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4792     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4793     \msg_error:nnxxxx{stex}{error/incompatible}
4794     {l__stex_structures_dom_str}
4795     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4796     {\l_stex_get_symbol_uri_str}
4797     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4798   }
4799   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4800 }
4801 }
4802
4803 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4804   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4805   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4806
4807   \stex_add_constant_to_current_module:n {\l_tmpa_str}
4808   \stex_execute_in_module:x {
4809     \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4810     name   = \l_tmpa_str ,
4811     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4812     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4813     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4814   }
4815   \seq_clear:c {l_stex_symdecl\l_stex_current_module_str?\l_tmpa_str _notations}
4816 }
4817
4818 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4819   \stex_find_notation:nn{##1}{}
4820   \stex_execute_in_module:x {
4821     \seq_put_right:cn {l_stex_symdecl\l_stex_current_module_str?\l_tmpa_str _notation
4822   }
4823
4824   \stex_copy_control_sequence_ii:ccN
4825   {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4826   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4827   \l_tmpa_tl
4828   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4829
4830
4831   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4832     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4833     \stex_execute_in_module:x {
4834       \tl_set:cn
4835       {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4836       { \exp_args:No \exp_not:n \l_tmpa_cs}
4837     }
4838   }
4839
4840 }
4841
4842 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4843 }

```



```

4844
4845 \stex_execute_in_module:x {
4846   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4847     domain = \l_stex_get_structure_module_str ,
4848     \prop_to_keyval:N \l_tmpa_prop
4849   }
4850   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4851 }
4852 \stex_debug:nn{instantiate}{
4853   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4854   \prop_to_keyval:N \l_tmpa_prop
4855 }
4856 \exp_args:Nxx \stex_symdecl_do:nn {
4857   type={\STEXsymbol{module-type}{
4858     \STEXInternalTermMathOMSiiii {
4859       \l_stex_get_structure_module_str
4860     }{}{0}{}}
4861   }}
4862 }{\l__stex_structures_name_str}
4863 % {
4864   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4865   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4866   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4867 % }
4868 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4869 \endgroup
4870 \stex_smsmode_do:\ignorespacesandpars
4871 }
4872
4873 \cs_new_protected:Nn \stex_symbol_or_var:n {
4874   \cs_if_exist:cTF{#1}{
4875     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4876     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4877     \str_if_empty:NTF \l_tmpa_str {
4878       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4879       \stex_invoke_variable:n {
4880         \bool_set_true:N \l_stex_symbol_or_var_bool
4881         \bool_set_false:N \l_stex_instance_or_symbol_bool
4882         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4883         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4884         \str_set:Nx \l_stex_get_symbol_uri_str {
4885           \exp_after:wN \use:n \l_tmpa_tl
4886         }
4887       }{ % TODO \stex_invoke_varinstance:n
4888         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
4889           \bool_set_true:N \l_stex_symbol_or_var_bool
4890           \bool_set_true:N \l_stex_instance_or_symbol_bool
4891           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4892           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4893           \str_set:Nx \l_stex_get_symbol_uri_str {
4894             \exp_after:wN \use:n \l_tmpa_tl
4895           }
4896         }{
4897           \bool_set_false:N \l_stex_symbol_or_var_bool

```

```

4898         \stex_get_symbol:n{#1}
4899     }
4900 }
4901 }{
4902     \__stex_structures_symbolorvar_from_string:n{ #1 }
4903 }
4904 }{
4905     \__stex_structures_symbolorvar_from_string:n{ #1 }
4906 }
4907 }
4908
4909 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4910     \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4911         \bool_set_true:N \l_stex_symbol_or_var_bool
4912         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4913     }{
4914         \bool_set_false:N \l_stex_symbol_or_var_bool
4915         \stex_get_symbol:n{#1}
4916     }
4917 }
4918
4919 \keys_define:nn { stex / varinstantiate } {
4920     name          .str_set_x:N = \l__stex_structures_name_str,
4921     bind          .choices:nn =
4922         {forall,exists}
4923         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4924 }
4925 }
4926 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4927     \str_clear:N \l__stex_structures_name_str
4928     \str_clear:N \l__stex_structures_bind_str
4929     \keys_set:nn { stex / varinstantiate } { #1 }
4930 }
4931
4932 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4933     \begingroup
4934         \stex_get_structure:n {#3}
4935         \__stex_structures_varinstantiate_args:n { #2 }
4936         \str_if_empty:NT \l__stex_structures_name_str {
4937             \str_set:Nn \l__stex_structures_name_str { #1 }
4938         }
4939         \stex_if_do_html:TF{
4940             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4941         }{\use:n}
4942         {
4943             \stex_if_do_html:T{
4944                 \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
4945             }
4946             \seq_clear:N \l__stex_structures_fields_seq
4947             \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4948             \seq_map_inline:Nn \l_stex_collect_imports_seq {
4949                 \seq_map_inline:cn {c_stex_module_##1_constants}{
4950                     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4951                 }

```

```

4952     }
4953     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4954     \prop_clear:N \l_tmpa_prop
4955     \tl_if_empty:nF {#5} {
4956         \seq_set_split:Nnn \l_tmpa_seq , {#5}
4957         \seq_map_inline:Nn \l_tmpa_seq {
4958             \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4959             \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4960                 \msg_error:nnn{stex}{error/keyval}{##1}
4961             }
4962             \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
4963             \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4964             \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq
4965             \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4966             \stex_if_do_html:T{
4967                 \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
4968                     \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{}\l_stex_get_symbol_uri_str}{}}
4969             }
4970             \bool_if:NTF \l_stex_symbol_or_var_bool {
4971                 \exp_args:Nxx \str_if_eq:nnF
4972                     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4973                     {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{\
4974                     \msg_error:nnxxx{stex}{error/incompatible}
4975                     {\l__stex_structures_dom_str}
4976                     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4977                     {\l_stex_get_symbol_uri_str}
4978                     {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}}
4979             }
4980             \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {}}
4981         }{
4982             \exp_args:Nxx \str_if_eq:nnF
4983                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4984                 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{\
4985                 \msg_error:nnxxx{stex}{error/incompatible}
4986                 {\l__stex_structures_dom_str}
4987                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4988                 {\l_stex_get_symbol_uri_str}
4989                 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}}
4990         }
4991         \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {}}
4992     }
4993 }
4994 }
4995 \tl_gclear:N \g__stex_structures_aftergroup_tl
4996 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4997     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l__stex_structures_fields_seq ##1}
4998     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4999     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5000         \stex_find_notation:nn{##1}{}
5001         \cs_gset_eq:cc{g__stex_structures_tmpa\l_tmpa_str _cs}
5002             {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5003         \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa\l_tmpa_str _cs}}
5004         \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5005             \cs_gset_eq:cc {g__stex_structures_tmpa_op\l_tmpa_str _cs}

```

```

5006         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5007         \stex_debug:nn{varinstantiate}{Operator-Notation:~\cs_meaning:c{g__stex_struct
5008     }
5009 }
5010
5011 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5012     \prop_set_from_keyval:cn { \l_stex_variable_ \l_tmpa_str _prop}{
5013         name    = \l_tmpa_str ,
5014         args    = \prop_item:cn { \l_stex_symdecl_##1_prop}{args} ,
5015         arity   = \prop_item:cn { \l_stex_symdecl_##1_prop}{arity} ,
5016         assocs  = \prop_item:cn { \l_stex_symdecl_##1_prop}{assocs}
5017     }
5018     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
5019     {g__stex_structures_tmpa_\l_tmpa_str _cs}
5020     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
5021     {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5022 }
5023 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn { \l_stex_symdecl_##1_prop}{name}}{\stex_inv
5024 }
5025 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5026     \prop_set_from_keyval:cn { \l_stex_varinstance_\l__stex_structures_name_str _prop }{
5027         domain = \l_stex_get_structure_module_str ,
5028         \prop_to_keyval:N \l_tmpa_prop
5029     }
5030     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5031     \tl_set:cn { \l_stex_varinstance_\l__stex_structures_name_str _op_tl }{
5032         \exp_args:Nnx \exp_not:N \use:nn {
5033             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5034             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5035                 \exp_not:n{
5036                     \_varcomp{#4}
5037                 }
5038             }
5039         }{
5040             \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5041         }
5042     }
5043 }
5044 }
5045 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{g__stex_structures_a
5046 \aftergroup\g__stex_structures_aftergroup_tl
5047 \endgroup
5048 \stex_smsmode_do:\ignorespacesandpars
5049 }
5050
5051 \cs_new_protected:Nn \stex_invoke_instance:n {
5052     \peek_charcode_remove:NTF ! {
5053         \stex_invoke_symbol:n{#1}
5054     }{
5055         \_stex_invoke_instance:nn {#1}
5056     }
5057 }
5058
5059

```

```

5060 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5061   \peek_charcode_remove:NTF ! {
5062     \exp_args:Nnx \use:nn {
5063       \def\comp{\_varcomp}
5064       \use:c{l_stex_varinstance_#1_op_tl}
5065     }{
5066       \_stex_reset:N \comp
5067     }
5068   }{
5069     \_stex_invoke_varinstance:nn {#1}
5070   }
5071 }
5072
5073 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5074   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5075     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5076   }{
5077     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5078     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5079       \prop_to_keyval:N \l_tmpa_prop
5080     }
5081   }
5082 }
5083
5084 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5085   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5086     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5087     \l_tmpa_tl
5088   }{
5089     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5090   }
5091 }

```

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

\stex_invoke_structure:nnn

```

5092 % #1: URI of the instance
5093 % #2: URI of the instantiated module
5094 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5095   \tl_if_empty:nTF{ #3 }{
5096     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5097       c_stex_feature_ #2 _prop
5098     }
5099     \tl_clear:N \l_tmpa_tl
5100     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5101     \seq_map_inline:Nn \l_tmpa_seq {
5102       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5103       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5104       \cs_if_exist:cT {
5105         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5106       }{
5107         \tl_if_empty:NF \l_tmpa_tl {
5108           \tl_put_right:Nn \l_tmpa_tl {,}
5109         }

```

```

5110         \tl_put_right:Nx \l_tmpa_tl {
5111             \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5112         }
5113     }
5114 }
5115 \exp_args:No \mathstruct \l_tmpa_tl
5116 }{
5117     \stex_invoke_symbol:n{#1/#3}
5118 }
5119 }

```

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

```

5120 </package>

```

Chapter 31

STEX -Statements Implementation

```
5121 <*package>
5122
5123 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5124
5125 <@@=stex_statements>
    Warnings and error messages
5126

\titleemph
5127 \def\titleemph#1{\textbf{#1}}

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

31.1 Definitions

definiendum

```
5128 \keys_define:nn {stex / definiendum }{
5129   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5130   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5131   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5132   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5133 }
5134 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5135   \str_clear:N \l__stex_statements_definiendum_root_str
5136   \tl_clear:N \l__stex_statements_definiendum_post_tl
5137   \str_clear:N \l__stex_statements_definiendum_gfa_str
5138   \keys_set:nn { stex / definiendum }{ #1 }
5139 }
5140 \NewDocumentCommand \definiendum { O{} m m } {
5141   \__stex_statements_definiendum_args:n { #1 }
5142   \stex_get_symbol:n { #2 }
5143   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5144   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5145     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5146     \tl_set:Nn \l_tmpa_tl { #3 }
5147   } {
5148     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5149     \tl_set:Nn \l_tmpa_tl {
5150       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5151     }
5152   }
5153 } {
5154   \tl_set:Nn \l_tmpa_tl { #3 }
5155 }
5156
5157 % TODO root
5158 \stex_html_backend:TF {
5159   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5160 } {
5161   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5162 }
5163 }
5164 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

5165
5166 \NewDocumentCommand \definame { 0{ } m } {
5167   \__stex_statements_definiendum_args:n { #1 }
5168   % TODO: root
5169   \stex_get_symbol:n { #2 }
5170   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5171   \str_set:Nx \l_tmpa_str {
5172     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5173   }
5174   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5175   \stex_html_backend:TF {
5176     \stex_if_do_html:T {
5177       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5178         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5179       }
5180     }
5181   } {
5182     \exp_args:Nnx \defemph@uri {
5183       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5184     } { \l_stex_get_symbol_uri_str }
5185   }
5186 }
5187 \stex_deactivate_macro:Nn \definame {definition~environments}
5188
5189 \NewDocumentCommand \Definame { 0{ } m } {
5190   \__stex_statements_definiendum_args:n { #1 }
5191   \stex_get_symbol:n { #2 }
5192   \str_set:Nx \l_tmpa_str {
5193     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5194   }
5195   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```



```

5196 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5197 \stex_html_backend:TF {
5198   \stex_if_do_html:T {
5199     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5200       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5201     }
5202   }
5203 } {
5204   \exp_args:Nnx \defemph@uri {
5205     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5206   } { \l_stex_get_symbol_uri_str }
5207 }
5208 }
5209 \stex_deactivate_macro:Nn \Definame {definition-environments}
5210
5211 \NewDocumentCommand \premise { m }{
5212   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5213 }
5214 \NewDocumentCommand \conclusion { m }{
5215   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5216 }
5217 \NewDocumentCommand \definiens { 0{} m }{
5218   \str_clear:N \l_stex_get_symbol_uri_str
5219   \tl_if_empty:nF {#1} {
5220     \stex_get_symbol:n { #1 }
5221   }
5222   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5223     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5224       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5225     }{
5226       % TODO throw error
5227     }
5228   }
5229   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5230   {\l_stex_current_module_str}{
5231     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5232   }{true}{
5233     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5234     \exp_args:Nx \stex_add_to_current_module:n {
5235       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5236     }
5237   }
5238 }
5239 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5240 }
5241
5242 \NewDocumentCommand \varbindforall {m}{
5243   \stex_symbol_or_var:n {#1}
5244   \bool_if:NTF\l_stex_symbol_or_var_bool{
5245     \stex_if_do_html:T {
5246       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5247     }
5248   }{
5249     % todo throw error

```

```

5250 }
5251 }
5252
5253 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5254 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5255 \stex_deactivate_macro:Nn \definiens {definition~environments}
5256 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5257

```

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

sdefinition

```

5258
5259 \keys_define:nn {stex / sdefinition }{
5260   type      .str_set_x:N = \sdefinitiontype,
5261   id        .str_set_x:N = \sdefinitionid,
5262   name      .str_set_x:N = \sdefinitionname,
5263   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5264   title     .tl_set:N     = \sdefinitiontitle
5265 }
5266 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5267   \str_clear:N \sdefinitiontype
5268   \str_clear:N \sdefinitionid
5269   \str_clear:N \sdefinitionname
5270   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5271   \tl_clear:N \sdefinitiontitle
5272   \keys_set:nn { stex / sdefinition }{ #1 }
5273 }
5274
5275 \NewDocumentEnvironment{sdefinition}{0{}}{
5276   \__stex_statements_sdefinition_args:n{ #1 }
5277   \stex_reactivate_macro:N \definiendum
5278   \stex_reactivate_macro:N \definame
5279   \stex_reactivate_macro:N \Definame
5280   \stex_reactivate_macro:N \premise
5281   \stex_reactivate_macro:N \definiens
5282   \stex_reactivate_macro:N \varbindforall
5283   \stex_if_smsmode:F{
5284     \seq_clear:N \l_tmpb_seq
5285     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5286       \tl_if_empty:nF{ ##1 }{
5287         \stex_get_symbol:n { ##1 }
5288         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5289           \l_stex_get_symbol_uri_str
5290         }
5291       }
5292     }
5293     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5294     \exp_args:Nnnx
5295     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5296     \str_if_empty:NF \sdefinitiontype {
5297       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5298     }
5299     \str_if_empty:NF \sdefinitionname {

```

```

5300     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5301   }
5302   \clist_set:No \l_tmpa_clist \sdefinitiontype
5303   \tl_clear:N \l_tmpa_tl
5304   \clist_map_inline:Nn \l_tmpa_clist {
5305     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5306       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5307     }
5308   }
5309   \tl_if_empty:NTF \l_tmpa_tl {
5310     \__stex_statements_sdefinition_start:
5311   }{
5312     \l_tmpa_tl
5313   }
5314 }
5315 \stex_ref_new_doc_target:n \sdefinitionid
5316 \stex_smsmode_do:
5317 ){
5318   \stex_suppress_html:n {
5319     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5320   }
5321   \stex_if_smsmode:F {
5322     \clist_set:No \l_tmpa_clist \sdefinitiontype
5323     \tl_clear:N \l_tmpa_tl
5324     \clist_map_inline:Nn \l_tmpa_clist {
5325       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5326         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5327       }
5328     }
5329     \tl_if_empty:NTF \l_tmpa_tl {
5330       \__stex_statements_sdefinition_end:
5331     }{
5332       \l_tmpa_tl
5333     }
5334     \end{stex_annotate_env}
5335   }
5336 }

```

\stexpatchdefinition

```

5337 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5338   \stex_par:\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
5339     ~(\sdefinitiontitle)
5340   }~}
5341 }
5342 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5343
5344 \newcommand\stexpatchdefinition[3]{} {
5345   \str_set:Nx \l_tmpa_str{ #1 }
5346   \str_if_empty:NTF \l_tmpa_str {
5347     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5348     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5349   }{
5350     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5351     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5352     }
5353 }

```

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

`\inlinedef` inline:

```

5354 \keys_define:nn {stex / inlinedef }{
5355   type      .str_set_x:N = \sdefinitiontype,
5356   id        .str_set_x:N = \sdefinitionid,
5357   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5358   name      .str_set_x:N = \sdefinitionname
5359 }
5360 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5361   \str_clear:N \sdefinitiontype
5362   \str_clear:N \sdefinitionid
5363   \str_clear:N \sdefinitionname
5364   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5365   \keys_set:nn { stex / inlinedef }{ #1 }
5366 }
5367 \NewDocumentCommand \inlinedef { 0{} m } {
5368   \begin{group}
5369     \__stex_statements_inlinedef_args:n{ #1 }
5370     \stex_reactivate_macro:N \definiendum
5371     \stex_reactivate_macro:N \definame
5372     \stex_reactivate_macro:N \Definame
5373     \stex_reactivate_macro:N \premise
5374     \stex_reactivate_macro:N \definiens
5375     \stex_reactivate_macro:N \varbindforall
5376     \stex_ref_new_doc_target:n \sdefinitionid
5377     \stex_if_smsmode:TF{\stex_suppress_html:n {
5378       \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5379     }}{
5380       \seq_clear:N \l_tmpb_seq
5381       \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5382         \tl_if_empty:nF{ ##1 }{
5383           \stex_get_symbol:n { ##1 }
5384           \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5385             \l_stex_get_symbol_uri_str
5386           }
5387         }
5388       }
5389       \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5390       \exp_args:Nnx
5391       \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5392         \str_if_empty:NF \sdefinitiontype {
5393           \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5394         }
5395         #2
5396         \str_if_empty:NF \sdefinitionname {
5397           \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5398           \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5399         }
5400       }
5401     }

```

```

5402 \endgroup
5403 \stex_smsmode_do:
5404 }

```

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

31.2 Assertions

sassertion

```

5405 \keys_define:nn {stex / sassertion }{
5406   type      .str_set_x:N = \sassertiontype,
5407   id        .str_set_x:N = \sassertionid,
5408   title     .tl_set:N     = \sassertiontitle ,
5409   for       .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5410   name      .str_set_x:N  = \sassertionname
5411 }
5412
5413 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5414   \str_clear:N \sassertiontype
5415   \str_clear:N \sassertionid
5416   \str_clear:N \sassertionname
5417   \clist_clear:N \l__stex_statements_sassertion_for_clist
5418   \tl_clear:N \sassertiontitle
5419   \keys_set:nn { stex / sassertion }{ #1 }
5420 }
5421
5422 %\tl_new:N \g__stex_statements_aftergroup_tl
5423
5424 \NewDocumentEnvironment{sassertion}{0}{}{
5425   \__stex_statements_sassertion_args:n{ #1 }
5426   \stex_reactivate_macro:N \premise
5427   \stex_reactivate_macro:N \conclusion
5428   \stex_reactivate_macro:N \varbindforall
5429   \stex_if_smsmode:F {
5430     \seq_clear:N \l_tmpb_seq
5431     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5432       \tl_if_empty:nF{ ##1 }{
5433         \stex_get_symbol:n { ##1 }
5434         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5435           \l_stex_get_symbol_uri_str
5436         }
5437       }
5438     }
5439     \exp_args:Nnnx
5440     \begin{sassertion_env}{assertion}{\seq_use:Nn \l_tmpb_seq {},}}
5441     \str_if_empty:NF \sassertiontype {
5442       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5443     }
5444     \str_if_empty:NF \sassertionname {
5445       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5446     }
5447     \clist_set:N \l_tmpa_clist \sassertiontype
5448     \tl_clear:N \l_tmpa_tl

```

```

5449 \clist_map_inline:Nn \l_tmpa_clist {
5450   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5451     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5452   }
5453 }
5454 \tl_if_empty:NTF \l_tmpa_tl {
5455   \__stex_statements_sassertion_start:
5456 }{
5457   \l_tmpa_tl
5458 }
5459 }
5460 \str_if_empty:NTF \sassertionid {
5461   \str_if_empty:NF \sassertionname {
5462     \stex_ref_new_doc_target:n {}
5463   }
5464 } {
5465   \stex_ref_new_doc_target:n \sassertionid
5466 }
5467 \stex_smsmode_do:
5468 ){
5469   \str_if_empty:NF \sassertionname {
5470     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5471     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5472   }
5473   \stex_if_smsmode:F {
5474     \clist_set:No \l_tmpa_clist \sassertiontype
5475     \tl_clear:N \l_tmpa_tl
5476     \clist_map_inline:Nn \l_tmpa_clist {
5477       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5478         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5479       }
5480     }
5481     \tl_if_empty:NTF \l_tmpa_tl {
5482       \__stex_statements_sassertion_end:
5483     }{
5484       \l_tmpa_tl
5485     }
5486     \end{stex_annotate_env}
5487   }
5488 }

```

\stexpatchassertion

```

5489
5490 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5491   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5492     (\sassertiontitle)
5493   }~}
5494 }
5495 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5496
5497 \newcommand\stexpatchassertion[3] [] {
5498   \str_set:Nx \l_tmpa_str{ #1 }
5499   \str_if_empty:NTF \l_tmpa_str {
5500     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

5501     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5502   }{
5503     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5504     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5505   }
5506 }

```

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

`\inlineass` inline:

```

5507 \keys_define:nn {stex / inlineass }{
5508   type      .str_set_x:N = \sassertiontype,
5509   id        .str_set_x:N = \sassertionid,
5510   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5511   name      .str_set_x:N = \sassertionname
5512 }
5513 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5514   \str_clear:N \sassertiontype
5515   \str_clear:N \sassertionid
5516   \str_clear:N \sassertionname
5517   \clist_clear:N \l__stex_statements_sassertion_for_clist
5518   \keys_set:nn { stex / inlineass }{ #1 }
5519 }
5520 \NewDocumentCommand \inlineass { 0{} m } {
5521   \beginngroup
5522   \stex_reactivate_macro:N \premise
5523   \stex_reactivate_macro:N \conclusion
5524   \stex_reactivate_macro:N \varbindforall
5525   \__stex_statements_inlineass_args:n{ #1 }
5526   \str_if_empty:NTF \sassertionid {
5527     \str_if_empty:NF \sassertionname {
5528       \stex_ref_new_doc_target:n {}
5529     }
5530   } {
5531     \stex_ref_new_doc_target:n \sassertionid
5532   }
5533
5534   \stex_if_smsmode:TF{
5535     \str_if_empty:NF \sassertionname {
5536       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5537       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5538     }
5539   }{
5540     \seq_clear:N \l_tmpb_seq
5541     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5542       \tl_if_empty:nF{ ##1 }{
5543         \stex_get_symbol:n { ##1 }
5544         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5545           \l_stex_get_symbol_uri_str
5546         }
5547       }
5548     }
5549     \exp_args:Nnx
5550     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{

```

```

5551 \str_if_empty:NF \sassertiontype {
5552   \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{ }
5553 }
5554 #2
5555 \str_if_empty:NF \sassertionname {
5556   \stex_suppress_html:n{\stex_symdecl_do:nn{ }\sassertionname}}
5557   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5558   \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5559 }
5560 }
5561 }
5562 \endgroup
5563 \stex_smsmode_do:
5564 }

```

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

31.3 Examples

`sexample`

```

5565
5566 \keys_define:nn {stex / sexample }{
5567   type      .str_set_x:N = \exampletype,
5568   id        .str_set_x:N = \sexampleid,
5569   title     .tl_set:N    = \sexamplename,
5570   name      .str_set_x:N = \sexamplename ,
5571   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5572 }
5573 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5574   \str_clear:N \sexampletype
5575   \str_clear:N \sexampleid
5576   \str_clear:N \sexamplename
5577   \tl_clear:N \sexamplename
5578   \clist_clear:N \l__stex_statements_sexample_for_clist
5579   \keys_set:nn { stex / sexample }{ #1 }
5580 }
5581
5582 \NewDocumentEnvironment{sexample}{0{}}{
5583   \__stex_statements_sexample_args:n{ #1 }
5584   \stex_reactivate_macro:N \premise
5585   \stex_reactivate_macro:N \conclusion
5586   \stex_if_smsmode:F {
5587     \seq_clear:N \l_tmpb_seq
5588     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5589       \tl_if_empty:nF{ ##1 }{
5590         \stex_get_symbol:n { ##1 }
5591         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5592           \l_stex_get_symbol_uri_str
5593         }
5594       }
5595     }
5596     \exp_args:Nnnx
5597     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```



```

5598 \str_if_empty:NF \sexamplotype {
5599 \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}}
5600 }
5601 \str_if_empty:NF \sexamplename {
5602 \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}}
5603 }
5604 \clist_set:No \l_tmpa_clist \sexamplotype
5605 \tl_clear:N \l_tmpa_tl
5606 \clist_map_inline:Nn \l_tmpa_clist {
5607 \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5608 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5609 }
5610 }
5611 \tl_if_empty:NTF \l_tmpa_tl {
5612 \__stex_statements_sexample_start:
5613 }{
5614 \l_tmpa_tl
5615 }
5616 }
5617 \str_if_empty:NF \sexampleid {
5618 \stex_ref_new_doc_target:n \sexampleid
5619 }
5620 \stex_smsmode_do:
5621 }{
5622 \str_if_empty:NF \sexamplename {
5623 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5624 }
5625 \stex_if_smsmode:F {
5626 \clist_set:No \l_tmpa_clist \sexamplotype
5627 \tl_clear:N \l_tmpa_tl
5628 \clist_map_inline:Nn \l_tmpa_clist {
5629 \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5630 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5631 }
5632 }
5633 \tl_if_empty:NTF \l_tmpa_tl {
5634 \__stex_statements_sexample_end:
5635 }{
5636 \l_tmpa_tl
5637 }
5638 \end{stex_annotate_env}
5639 }
5640 }

```

\stexpatchexample

```

5641
5642 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5643 \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
5644 (\sexampltitle)
5645 }~}
5646 }
5647 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5648
5649 \newcommand\stexpatchexample[3]{} {

```

```

5650 \str_set:Nx \l_tmpa_str{ #1 }
5651 \str_if_empty:NTF \l_tmpa_str {
5652   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5653   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5654 }{
5655   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5656   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5657 }
5658 }

```

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

`\inlineex` inline:

```

5659 \keys_define:nn {stex / inlineex }{
5660   type      .str_set_x:N = \sexamplotype,
5661   id        .str_set_x:N = \sexampleid,
5662   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5663   name      .str_set_x:N = \sexamplename
5664 }
5665 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5666   \str_clear:N \sexamplotype
5667   \str_clear:N \sexampleid
5668   \str_clear:N \sexamplename
5669   \clist_clear:N \l__stex_statements_sexample_for_clist
5670   \keys_set:nn { stex / inlineex }{ #1 }
5671 }
5672 \NewDocumentCommand \inlineex { 0{} m } {
5673   \begingroup
5674   \stex_reactivate_macro:N \premise
5675   \stex_reactivate_macro:N \conclusion
5676   \__stex_statements_inlineex_args:n{ #1 }
5677   \str_if_empty:NF \sexampleid {
5678     \stex_ref_new_doc_target:n \sexampleid
5679   }
5680   \stex_if_smsmode:TF{
5681     \str_if_empty:NF \sexamplename {
5682       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
5683   }
5684 }{
5685   \seq_clear:N \l_tmpb_seq
5686   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5687     \tl_if_empty:nF{ ##1 }{
5688       \stex_get_symbol:n { ##1 }
5689       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5690         \l_stex_get_symbol_uri_str
5691       }
5692     }
5693   }
5694   \exp_args:Nnx
5695   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
5696     \str_if_empty:NF \sexamplotype {
5697       \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5698     }
5699     #2

```

```

5700     \str_if_empty:NF \sexamplename {
5701       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5702       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5703     }
5704   }
5705 }
5706 \endgroup
5707 \stex_smsmode_do:
5708 }

```

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

31.4 Logical Paragraphs

`sparagraph`

```

5709 \keys_define:nn { stex / sparagraph } {
5710   id      .str_set:x:N = \sparagraphid ,
5711   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
5712   type    .str_set:x:N = \sparagraphtype ,
5713   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5714   from    .tl_set:N    = \sparagraphfrom ,
5715   to      .tl_set:N    = \sparagraphto ,
5716   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
5717   name     .str_set:N   = \sparagraphname ,
5718   imports .tl_set:N    = \l__stex_statements_sparagraph_imports_tl
5719 }
5720
5721 \cs_new_protected:Nn \stex_sparagraph_args:n {
5722   \tl_clear:N \l_stex_sparagraph_title_tl
5723   \tl_clear:N \sparagraphfrom
5724   \tl_clear:N \sparagraphto
5725   \tl_clear:N \l_stex_sparagraph_start_tl
5726   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5727   \str_clear:N \sparagraphid
5728   \str_clear:N \sparagraphtype
5729   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5730   \str_clear:N \sparagraphname
5731   \keys_set:nn { stex / sparagraph } { #1 }
5732 }
5733 \newif\if@in@omtext\@in@omtextfalse
5734
5735 \NewDocumentEnvironment {sparagraph} { 0{ } } {
5736   \stex_sparagraph_args:n { #1 }
5737   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5738     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5739   }{
5740     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5741   }
5742   \@in@omtexttrue
5743   \stex_if_smsmode:F {
5744     \seq_clear:N \l_tmpb_seq
5745     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5746       \tl_if_empty:nF{ ##1 }{

```

```

5747     \stex_get_symbol:n { ##1 }
5748     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5749         \l_stex_get_symbol_uri_str
5750     }
5751 }
5752 }
5753 \exp_args:Nnnx
5754 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
5755 \str_if_empty:NF \sparagraphtype {
5756     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5757 }
5758 \str_if_empty:NF \sparagraphfrom {
5759     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5760 }
5761 \str_if_empty:NF \sparagraphto {
5762     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5763 }
5764 \str_if_empty:NF \sparagraphname {
5765     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5766 }
5767 \clist_set:No \l_tmpa_clist \sparagraphtype
5768 \tl_clear:N \l_tmpa_tl
5769 \clist_map_inline:Nn \sparagraphtype {
5770     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5771         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5772     }
5773 }
5774 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5775 \tl_if_empty:NTF \l_tmpa_tl {
5776     \__stex_statements_sparagraph_start:
5777 }{
5778     \l_tmpa_tl
5779 }
5780 }
5781 \clist_set:No \l_tmpa_clist \sparagraphtype
5782 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5783     {
5784         \stex_reactivate_macro:N \definiendum
5785         \stex_reactivate_macro:N \definame
5786         \stex_reactivate_macro:N \Definame
5787         \stex_reactivate_macro:N \premise
5788         \stex_reactivate_macro:N \definiens
5789     }
5790 \str_if_empty:NTF \sparagraphid {
5791     \str_if_empty:NTF \sparagraphname {
5792         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5793             \stex_ref_new_doc_target:n {}
5794         }
5795     } {
5796         \stex_ref_new_doc_target:n {}
5797     }
5798 } {
5799     \stex_ref_new_doc_target:n \sparagraphid
5800 }

```

```

5801 \exp_args:NNx
5802 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5803   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5804     \tl_if_empty:nF{ ##1 }{
5805       \stex_get_symbol:n { ##1 }
5806       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5807     }
5808   }
5809 }
5810 \stex_smsmode_do:
5811 \ignorespacesandpars
5812 }{
5813   \str_if_empty:NF \sparagraphname {
5814     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5815     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5816   }
5817   \stex_if_smsmode:F {
5818     \clist_set:No \l_tmpa_clist \sparagraphtype
5819     \tl_clear:N \l_tmpa_tl
5820     \clist_map_inline:Nn \l_tmpa_clist {
5821       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5822         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5823       }
5824     }
5825     \tl_if_empty:NTF \l_tmpa_tl {
5826       \__stex_statements_sparagraph_end:
5827     }{
5828       \l_tmpa_tl
5829     }
5830     \end{stex_annotate_env}
5831   }
5832 }

```

\stexpatchparagraph

```

5833
5834 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5835   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5836     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5837       \titleemph{\l_stex_sparagraph_title_tl}:~
5838     }
5839   }{
5840     \titleemph{\l_stex_sparagraph_start_tl}~
5841   }
5842 }
5843 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5844
5845 \newcommand\stexpatchparagraph[3] [] {
5846   \str_set:Nx \l_tmpa_str{ #1 }
5847   \str_if_empty:NTF \l_tmpa_str {
5848     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5849     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5850   }{
5851     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5852     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

5853     }
5854 }
5855
5856 \keys_define:nn { stex / inlinepara } {
5857   id      .str_set:N = \sparagraphid ,
5858   type    .str_set:N = \sparagraphtype ,
5859   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5860   from    .tl_set:N   = \sparagraphfrom ,
5861   to      .tl_set:N   = \sparagraphto ,
5862   name    .str_set:N   = \sparagraphname
5863 }
5864 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5865   \tl_clear:N \sparagraphfrom
5866   \tl_clear:N \sparagraphto
5867   \str_clear:N \sparagraphid
5868   \str_clear:N \sparagraphtype
5869   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5870   \str_clear:N \sparagraphname
5871   \keys_set:nn { stex / inlinepara }{ #1 }
5872 }
5873 \NewDocumentCommand \inlinepara { 0{} m } {
5874   \begingroup
5875   \__stex_statements_inlinepara_args:n{ #1 }
5876   \clist_set:Nn \l_tmpa_clist \sparagraphtype
5877   \str_if_empty:NTF \sparagraphid {
5878     \str_if_empty:NTF \sparagraphname {
5879       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5880         \stex_ref_new_doc_target:n {}
5881       }
5882     } {
5883       \stex_ref_new_doc_target:n {}
5884     }
5885   } {
5886     \stex_ref_new_doc_target:n \sparagraphid
5887   }
5888   \stex_if_smsmode:TF{
5889     \str_if_empty:NF \sparagraphname {
5890       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5891     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5892   }
5893 }{
5894   \seq_clear:N \l_tmpb_seq
5895   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5896     \tl_if_empty:nF{ ##1 }{
5897       \stex_get_symbol:n { ##1 }
5898       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5899         \l_stex_get_symbol_uri_str
5900       }
5901     }
5902   }
5903   \exp_args:Nnx
5904   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
5905     \str_if_empty:NF \sparagraphtype {
5906       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

5907     }
5908     \str_if_empty:NF \sparagraphfrom {
5909         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5910     }
5911     \str_if_empty:NF \sparagraphto {
5912         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5913     }
5914     \str_if_empty:NF \sparagraphname {
5915         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5916         \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5917         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5918     }
5919     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5920         \clist_map_inline:Nn \l_tmpb_seq {
5921             \stex_ref_new_sym_target:n {##1}
5922         }
5923     }
5924     #2
5925 }
5926 }
5927 \endgroup
5928 \stex_smsmode_do:
5929 }
5930

```

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

```

5931 </package>

```

Chapter 32

The Implementation

```
5932 <*package>
5933 <@@=stex_sproof>
5934
5935 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
5936
```

32.1 Proofs

We first define some keys for the proof environment.

```
5937 \keys_define:nn { stex / spf } {
5938   id          .str_set_x:N = \spfid,
5939   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5940   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5941   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5942   type        .str_set_x:N = \spftype,
5943   title       .tl_set:N    = \spftitle,
5944   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5945   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5946   term        .tl_set:N    = \l__stex_sproof_spf_term_tl,
5947   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
5948   hide        .bool_set:N  = \l__stex_sproof_spf_hide_bool
5949 }
5950 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5951   \str_clear:N \spfid
5952   \tl_clear:N \l__stex_sproof_spf_for_tl
5953   \tl_clear:N \l__stex_sproof_spf_from_tl
5954   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5955   \str_clear:N \spftype
5956   \tl_clear:N \spftitle
5957   \tl_clear:N \l__stex_sproof_spf_continues_tl
5958   \tl_clear:N \l__stex_sproof_spf_term_tl
5959   \tl_clear:N \l__stex_sproof_spf_functions_tl
5960   \tl_clear:N \l__stex_sproof_spf_method_tl
5961   \bool_set_false:N \l__stex_sproof_spf_hide_bool
5962   \keys_set:nn { stex / spf }{ #1 }
5963 }
5964 \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`

```
5965 \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.

```
5966 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5967 \cs_new_protected:Npn \sproofnumber {
5968   \int_set:Nn \l_tmpa_int {1}
5969   \bool_while_do:nn {
5970     \int_compare_p:nNn {
5971       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5972     } > 0
5973   }{
5974     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5975     \int_incr:N \l_tmpa_int
5976   }
5977 }
5978 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5979   \int_set:Nn \l_tmpa_int {1}
5980   \bool_while_do:nn {
5981     \int_compare_p:nNn {
5982       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5983     } > 0
5984   }{
5985     \int_incr:N \l_tmpa_int
5986   }
5987   \int_compare:nNnF \l_tmpa_int = 1 {
5988     \int_decr:N \l_tmpa_int
5989   }
5990   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5991     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5992   }
5993 }
5994
5995 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5996   \int_set:Nn \l_tmpa_int {1}
5997   \bool_while_do:nn {
5998     \int_compare_p:nNn {
5999       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6000     } > 0
6001   }{
6002     \int_incr:N \l_tmpa_int
6003   }
6004   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6005 }
6006
```

```

6007 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6008   \int_set:Nn \l_tmpa_int {1}
6009   \bool_while_do:nn {
6010     \int_compare_p:nNn {
6011       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6012     } > 0
6013   }{
6014     \int_incr:N \l_tmpa_int
6015   }
6016   \int_decr:N \l_tmpa_int
6017   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6018 }

```

\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

```

6019 \def\sproof@box{
6020   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6021 }
6022 \def\sproofend{
6023   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6024     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6025   }
6026 }

```

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

spf@*kw

```

6027 \def\spf@proofsketch@kw{Proof~Sketch}
6028 \def\spf@proof@kw{Proof}
6029 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6030 \AddToHook{begindocument}{
6031   \ltx@ifpackageloaded{babel}{
6032     \makeatletter
6033     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6034     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6035       \input{sproof-ngerman.ldf}
6036     }
6037     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6038       \input{sproof-finnish.ldf}
6039     }
6040     \clist_if_in:NnT \l_tmpa_clist {french}{
6041       \input{sproof-french.ldf}
6042     }
6043     \clist_if_in:NnT \l_tmpa_clist {russian}{
6044       \input{sproof-russian.ldf}
6045     }
6046     \makeatother
6047   }{}
6048 }

```

spfsketch

```

6049 \newcommand\spfsketch[2] [] {
6050   \begin{group}
6051   \let \premise \stex_proof_premise:
6052   \_stex_sproof_spf_args:n{#1}
6053   \stex_if_smsmode:TF {
6054     \str_if_empty:NF \spfid {
6055       \stex_ref_new_doc_target:n \spfid
6056     }
6057   }{
6058     \seq_clear:N \l_tmpa_seq
6059     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6060       \tl_if_empty:nF{ ##1 }{
6061         \stex_get_symbol:n { ##1 }
6062         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6063           \l_stex_get_symbol_uri_str
6064         }
6065       }
6066     }
6067     \exp_args:Nnx
6068     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6069       \str_if_empty:NF \spftype {
6070         \stex_annotate_invisible:nnn{type}{\spftype}{ }
6071       }
6072       \clist_set:No \l_tmpa_clist \spftype
6073       \tl_set:Nn \l_tmpa_tl {
6074         \titleemph{
6075           \tl_if_empty:NTF \spftitle {
6076             \spf@proofsketch@kw
6077           }{
6078             \spftitle
6079           }
6080         }:-
6081       }
6082       \clist_map_inline:Nn \l_tmpa_clist {
6083         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6084           \tl_clear:N \l_tmpa_tl
6085         }
6086       }
6087       \str_if_empty:NF \spfid {
6088         \stex_ref_new_doc_target:n \spfid
6089       }
6090       \l_tmpa_tl #2 \sproofend
6091     }
6092   }
6093   \endgroup
6094   \stex_smsmode_do:
6095 }
6096

```

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

```

\_stex_sproof_maybe_comment:
\_stex_sproof_maybe_comment_end:
\_stex_sproof_start_comment:
6097 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6098
6099 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6100   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6101     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6102   }
6103 }
6104 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6105   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6106 }
6107 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6108   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6109 }
6110

```

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

\stexcommentfont

```

6111 \cs_new_protected:Npn \stexcommentfont {
6112   \small\itshape
6113 }

```

(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.

```

6114 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6115   \seq_clear:N \l_tmpa_seq
6116   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6117     \tl_if_empty:NF{ ##1 }{
6118       \stex_get_symbol:n { ##1 }
6119       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6120         \l_stex_get_symbol_uri_str
6121       }
6122     }
6123   }
6124   \exp_args:Nnnx
6125   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6126   \str_if_empty:NF \spftype {
6127     \stex_annotate_invisible:nnn{type}{\spftype}{}
6128   }
6129   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6130   \str_if_empty:NF \spfid {
6131     \stex_ref_new_doc_target:n \spfid
6132   }
6133   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6134   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6135     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6136   }
6137   \begin{list}{}{
6138     \setlength\topsep{0pt}
6139     \setlength\parsep{0pt}
6140     \setlength\rightmargin{0pt}

```

```

6141 } \_stex_sproof_maybe_comment:
6142 }
6143 \cs_new_protected:Nn \_stex_sproof_end_env:n {
6144   \stex_if_smsmode:F{
6145     \_stex_sproof_maybe_comment_end:
6146     \end{list}
6147     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6148       \stex_html_backend:F{\egroup}
6149     }
6150     \clist_set:No \l_tmpa_clist \spftype
6151     #1
6152     \end{stex_annotate_env}
6153     \end{stex_annotate_env}
6154   }
6155 }
6156 }
6157 \NewDocumentEnvironment{sproof}{s O{} m}{
6158   \intarray_gzero:N \l__stex_sproof_counter_intarray
6159   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6160   \stex_reactivate_macro:N \yield
6161   \stex_reactivate_macro:N \eqstep
6162   \stex_reactivate_macro:N \assumption
6163   \stex_reactivate_macro:N \conclude
6164   \stex_reactivate_macro:N \spfstep
6165   \_stex_sproof_spf_args:n{#2}
6166   \stex_if_smsmode:TF {
6167     \str_if_empty:NF \spfid {
6168       \stex_ref_new_doc_target:n \spfid
6169     }
6170   }{
6171     \_stex_sproof_start_env:nnn{sproof}{#3}{
6172       \clist_set:No \l_tmpa_clist \spftype
6173       \tl_clear:N \l_tmpa_tl
6174       \clist_map_inline:Nn \l_tmpa_clist {
6175         \tl_if_exist:cT {\_stex_sproof_sproof_##1_start:}{
6176           \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_start:}}
6177         }
6178         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6179           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6180         }
6181       }
6182       \tl_if_empty:NTF \l_tmpa_tl {
6183         \_stex_sproof_sproof_start:
6184       }{
6185         \l_tmpa_tl
6186       }
6187     }
6188   }
6189   \stex_smsmode_do:
6190 }{\_stex_sproof_end_env:n{
6191   \tl_clear:N \l_tmpa_tl
6192   \clist_map_inline:Nn \l_tmpa_clist {
6193     \tl_if_exist:cT {\_stex_sproof_sproof_##1_end:}{
6194       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_end:}}

```

```

6195     }
6196   }
6197   \tl_if_empty:NTF \l_tmpa_tl {
6198     \__stex_sproof_sproof_end:
6199   }{
6200     \l_tmpa_tl
6201   }
6202 }}
6203 \NewDocumentEnvironment{subproof}{s O{} m}{
6204   \__stex_sproof_spf_args:n{#2}
6205   \stex_if_smsmode:TF {
6206     \str_if_empty:NF \spfid {
6207       \stex_ref_new_doc_target:n \spfid
6208     }
6209   }{
6210     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6211   }
6212   \__stex_sproof_add_counter:
6213   \stex_smsmode_do:
6214 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6215   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6216     \__stex_sproof_inc_counter:
6217   }
6218   \aftergroup\__stex_sproof_maybe_comment:
6219 }
6220 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6221
6222 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6223   \par\noindent\titleemph{
6224     \tl_if_empty:NTF \spftype {
6225       \spf@proof@kw
6226     }{
6227       \spftype
6228     }
6229   }:
6230 }
6231 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6232
6233 \newcommand\stexpatchproof[3] [] {
6234   \str_set:Nx \l_tmpa_str{ #1 }
6235   \str_if_empty:NTF \l_tmpa_str {
6236     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6237     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6238   }{
6239     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6240     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6241   }
6242 }

\pstep
\conclude
\assumption
\have
\eqstep

```

```

6247 type          .str_set_x:N = \spftype,
6248 title         .tl_set:N = \spftitle,
6249 method        .tl_set:N = \l__stex_sproof_spf_method_tl,
6250 term          .tl_set:N = \l__stex_sproof_spf_term_tl
6251 }
6252 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6253 \str_clear:N \spfstepid
6254 \clist_clear:N \l__stex_sproof_spf_for_clist
6255 \str_clear:N \spftype
6256 \tl_clear:N \l__stex_sproof_spf_method_tl
6257 \tl_clear:N \l__stex_sproof_spf_term_tl
6258 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6259 \keys_set:nn { stex / spfsteps }{ #1 }
6260 }
6261
6262 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6263 \NewDocumentCommand #1 {s O{} +m} {
6264 \__stex_sproof_maybe_comment_end:
6265
6266 \__stex_sproof_spfstep_args:n{##2}
6267 \stex_annotate:nnn{spfstep}{#2}{
6268 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6269 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6270 }
6271 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6272 #4
6273 }{
6274 \item[\IfBooleanTF ##1 {}{#3}]
6275 }
6276 \ignorespacesandpars ##3
6277 }
6278 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6279 \__stex_sproof_maybe_comment:
6280 }
6281 \stex_deactivate_macro:Nn #1 {sproof~environments}
6282 }
6283
6284 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6285 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6286 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6287
6288 \NewDocumentCommand \eqstep {s m}{
6289 \__stex_sproof_maybe_comment_end:
6290 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6291 $=$
6292 }{
6293 \item[$=$]
6294 }
6295 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6296 \__stex_sproof_maybe_comment:
6297 }
6298 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6299
6300 \NewDocumentCommand \yield {+m}{

```

```

6301 \stex_annotate:nnn{spfyield}{\}{ #1 }
6302 }
6303 \stex_deactivate_macro:Nn \yield {sproof~environments}
6304
6305 \NewDocumentEnvironment{spfblock}{\}{\{
6306 \item[]
6307 \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6308 }{
6309 \aftergroup\__stex_sproof_maybe_comment:
6310 }
6311 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6312

```

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

`\spfidea`

```

6313 \NewDocumentCommand\spfidea{0\} +m){
6314 \__stex_sproof_spf_args:n{#1}
6315 \titleemph{
6316 \tl_if_empty:NTF \spftype {Proof~Idea}{
6317 \spftype
6318 }:
6319 }~#2
6320 \sproofend
6321 }

```

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

```

6322 \newcommand\spfjust[1]{
6323 #1
6324 }
6325 \</package>

```

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

Chapter 33

STEX -Others Implementation

```
6326 <*package>
6327
6328 %%%%%%%%%% others.dtx %%%%%%%%%%
6329
6330 <@@=stex_others>
        Warnings and error messages
6331 % None

\MSC Math subject classifier

6332 \NewDocumentCommand \MSC {m} {
6333 % TODO
6334 }

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

6335 \@ifpackageloaded{tikzinput}{
6336 \RequirePackage{stex-tikzinput}
6337 }{}
6338
6339 \bool_if:NT \c_stex_persist_mode_bool {
6340 \let__stex_notation_restore_notation_old:nnnnn
6341 \__stex_notation_restore_notation:nnnnn
6342 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6343 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6344 \ExplSyntaxOn
6345 }
6346 \def__stex_notation_restore_notation:nnnnn{
6347 \ExplSyntaxOff
6348 \catcode'\sim10
6349 \__stex_notation_restore_notation_new:nnnnn
6350 }
6351 \input{\jobname.sms}
6352 \let__stex_notation_restore_notation:nnnnn
6353 \__stex_notation_restore_notation_old:nnnnn
6354 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```

6355     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6356     \l_tmpa_str
6357     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6358     \c_stex_mathhub_main_manifest_prop
6359     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6360   }
6361 }
6362 </package>

```

Chapter 34

STEX -Metatheory Implementation

```
6363 <*package>
6364 <@@=stex_modules>
6365
6366 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6367
6368 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6369 \begingroup
6370 \stex_module_setup:nn{
6371   ns=\c_stex_metatheory_ns_str,
6372   meta=NONE
6373 }{Metatheory}
6374 \stex_reactivate_macro:N \symdecl
6375 \stex_reactivate_macro:N \notation
6376 \stex_reactivate_macro:N \symdef
6377 \ExplSyntaxOff
6378 \csname stex_suppress_html:n\endcsname{
6379   % is-a (a:A, a \in A, a is an A, etc.)
6380   \symdecl{isa}[args=ai]
6381   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6382   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6383   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6384
6385   % bind (\forall, \Pi, \lambda etc.)
6386   \symdecl{bind}[args=Bi,assoc=pre]
6387   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6388   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6389   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6390
6391   % implicit bind
6392   \symdecl{implicitbind}[args=Bi,assoc=pre]
6393   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\;_I\;\cdot)}]{\comp{\; #1 \comp{
6394     \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to_I\; } #2}{##1 \comp,
6395     \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6396
6397   % dummy variable
```

```

6398 \symdecl{dummyvar}
6399 \notation{dummyvar}[underscore]{\comp\_}
6400 \notation{dummyvar}[dot]{\comp\cdot}
6401 \notation{dummyvar}[dash]{\comp{\rm --}}
6402
6403 %fromto (function space, Hom-set, implication etc.)
6404 \symdecl{fromto}[args=ai]
6405 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6406 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6407
6408 % mapto (lambda etc.)
6409 \symdecl{mapto}[args=Bi]
6410 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6411 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6412 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6413
6414 % function/operator application
6415 \symdecl{apply}[args=ia]
6416 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6417 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6418
6419 % collection of propositions/booleans/truth values
6420 \symdecl{prop}[name=proposition]
6421 \notation{prop}[prop]{\comp{\rm prop}}
6422 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6423
6424 \symdecl{judgmentholds}[args=1]
6425 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6426
6427 % sequences
6428 \symdecl{seqtype}[args=1]
6429 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6430
6431 \symdecl{seqexpr}[args=a]
6432 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6433
6434 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
6435 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6436 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6437 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp
6438 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp
6439 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6440 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6441 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6442 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6443
6444 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6445 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
6446
6447 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\, \ellipses}}{##1\comp,##2}
6448 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\, \ellipses,}#2}{##1\comp,##2}
6449 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\, \ellipses,}#2\comp{\, \ellipses,}#3}
6450
6451 % nat literals

```

```

6452 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6453
6454 % letin (''let'', local definitions, variable substitution)
6455 \symdecl{letin}[args=bii]
6456 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
6457 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6458 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6459
6460 % structures
6461 \symdecl*{module-type}[args=1]
6462 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6463 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6464 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\angle}{##1 \comp, ##2}
6465
6466 % objects
6467 \symdecl{object}
6468 \notation{object}{\comp{\mathtt{OBJECT}}}
6469
6470 }
6471
6472 % The following are abbreviations in the sTeX corpus that are left over from earlier
6473 % developments. They will eventually be phased out.
6474
6475 \ExplSyntaxOn
6476 \stex_add_to_current_module:n{
6477   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6478   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6479   \def\livar{\csname sequence-index\endcsname[li]}
6480   \def\uivar{\csname sequence-index\endcsname[ui]}
6481   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6482   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6483 }
6484 \__stex_modules_end_module:
6485 \endgroup
6486 \</package>

```

Chapter 35

Tikzinput Implementation

```
6487 <@@=tikzinput>
6488 <*package>
6489
6490 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6491
6492 \ProvidesExplPackage{tikzinput}{2022/05/24}{3.1.0}{tikzinput package}
6493 \RequirePackage{l3keys2e}
6494
6495 \keys_define:nn { tikzinput } {
6496   image .bool_set:N = \c_tikzinput_image_bool,
6497   image .default:n = false ,
6498   unknown .code:n = {}
6499 }
6500
6501 \ProcessKeysOptions { tikzinput }
6502
6503 \bool_if:NTF \c_tikzinput_image_bool {
6504   \RequirePackage{graphicx}
6505
6506   \providecommand\usetikzlibrary[]{}
6507   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6508 }{
6509   \RequirePackage{tikz}
6510   \RequirePackage{standalone}
6511
6512   \newcommand \tikzinput [2] [] {
6513     \setkeys{Gin}{#1}
6514     \ifx \Gin@ewidth \Gin@exclamation
6515       \ifx \Gin@eheight \Gin@exclamation
6516         \input { #2 }
6517       \else
6518         \resizebox{!}{ \Gin@eheight }{
6519           \input { #2 }
6520         }
6521       \fi
6522     \else
6523       \ifx \Gin@eheight \Gin@exclamation
6524         \resizebox{ \Gin@ewidth }{!}{
```

```

6525         \input { #2 }
6526     }
6527     \else
6528         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6529             \input { #2 }
6530         }
6531     \fi
6532 \fi
6533 }
6534 }
6535
6536 \newcommand \ctikzinput [2] [] {
6537     \begin{center}
6538         \tikzinput [#1] {#2}
6539     \end{center}
6540 }
6541
6542 \@ifpackageloaded{stex}{
6543     \RequirePackage{stex-tikzinput}
6544 }{}
6545
6546 </package>
6547 <*stex>
6548 \ProvidesExplPackage{stex-tikzinput}{2022/05/24}{3.1.0}{stex-tikzinput}
6549 \RequirePackage{stex}
6550 \RequirePackage{tikzinput}
6551
6552 \newcommand\mhtikzinput[2] []{%
6553     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
6554     \stex_in_repository:nn\Gin@mhrepos{
6555         \tikzinput[#1]{\mhp@path{##1}{#2}}
6556     }
6557 }
6558 \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6559
6560 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6561     \pgfkeys@spdef\pgf@temp{#1}
6562     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6563     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6564     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6565     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6566     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6567     \catcode'\@=11
6568     \catcode'\|=12
6569     \catcode'\$=3
6570     \pgfutil@InputIfFileExists{#2}{-}{-}
6571     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6572     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6573     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6574 }
6575
6576
6577 \newcommand\libusetikzlibrary[1]{

```

```

6578 \prop_if_exist:NF \l_stex_current_repository_prop {
6579   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6580 }
6581 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6582   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6583 }
6584 \seq_clear:N \l__tikzinput_libinput_files_seq
6585 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6586 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6587
6588 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6589   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6590   \IfFileExists{ \l_tmpa_str }{
6591     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6592   }{}
6593   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6594   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6595 }
6596
6597 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6598 \IfFileExists{ \l_tmpa_str }{
6599   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6600 }{}
6601
6602 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6603   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6604 }{
6605   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6606     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6607       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6608     }
6609   }{
6610     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6611   }
6612 }
6613 }
6614 </stex>

```


Chapter 36

document-structure.sty Implementation

```
6615 <*package>
6616 <@@=document_structure>
6617 \ProvidesExplPackage{document-structure}{2022/05/24}{3.1.0}{Modular Document Structure}
6618 \RequirePackage{13keys2e}
```

36.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option xxx will just set the appropriate switches to true (otherwise they stay false).

```
6619
6620 \keys_define:nn{ document-structure }{
6621   class      .str_set_x:N = \c_document_structure_class_str,
6622   topsect    .str_set_x:N = \c_document_structure_topsect_str,
6623   unknown    .code:n      = {
6624     \PassOptionsToClass{\CurrentOption}{stex}
6625     \PassOptionsToClass{\CurrentOption}{tikzinput}
6626   }
6627   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6628 }
6629 \ProcessKeysOptions{ document-structure }
6630 \str_if_empty:NT \c_document_structure_class_str {
6631   \str_set:Nn \c_document_structure_class_str {article}
6632 }
6633 \str_if_empty:NT \c_document_structure_topsect_str {
6634   \str_set:Nn \c_document_structure_topsect_str {section}
6635 }
```

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

```
6636 \RequirePackage{xspace}
6637 \RequirePackage{comment}
6638 \RequirePackage{stex}
6639 \AddToHook{begindocument}{
```

```

6640 \ltx@ifpackageloaded{babel}{
6641     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6642     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6643         \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6644     }
6645 }{}
6646 }

```

`\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.

```

6647 \int_new:N \l_document_structure_section_level_int
6648 \str_case:NnF \c_document_structure_topsect_str {
6649     {part}}{
6650     \int_set:Nn \l_document_structure_section_level_int {0}
6651 }
6652 {chapter}}{
6653     \int_set:Nn \l_document_structure_section_level_int {1}
6654 }
6655 }{
6656     \str_case:NnF \c_document_structure_class_str {
6657         {book}}{
6658             \int_set:Nn \l_document_structure_section_level_int {0}
6659         }
6660         {report}}{
6661             \int_set:Nn \l_document_structure_section_level_int {0}
6662         }
6663     }{
6664         \int_set:Nn \l_document_structure_section_level_int {2}
6665     }
6666 }

```

36.2 Document Structure

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

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

EdN:9

```

6667 \def\current@section@level{document}%
6668 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6669 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

6670 \cs_new_protected:Npn \skipfragment {

```

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

```

6671 \ifcase\l_document_structure_section_level_int
6672 \or\stepcounter{part}
6673 \or\stepcounter{chapter}
6674 \or\stepcounter{section}
6675 \or\stepcounter{subsection}
6676 \or\stepcounter{subsubsection}
6677 \or\stepcounter{paragraph}
6678 \or\stepcounter{subparagraph}
6679 \fi
6680 }

```

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

blindfragment

```

6681 \newcommand\at@begin@blindsfragment[1]{
6682 \newenvironment{blindfragment}
6683 {
6684 \int_incr:N\l_document_structure_section_level_int
6685 \at@begin@blindsfragment\l_document_structure_section_level_int
6686 }{}

```

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

```

6687 \newcommand\sfragment@nonum[2]{
6688 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6689 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6690 }

```

(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.

```

6691 \newcommand\sfragment@num[2]{
6692 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6693 \@nameuse{#1}{#2}
6694 }{
6695 \cs_if_exist:NTF\rdfmata@sectioning{
6696 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6697 }{
6698 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6699 }
6700 }
6701 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6702 }

```

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

sfragment

```

6703 \keys_define:nn { document-structure / sfragment }{
6704 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6705 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6706 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6707 contributors   .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6708 srccite        .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6709 type          .tl_set:N     = \l__document_structure_sfragment_type_tl,
6710 short         .tl_set:N     = \l__document_structure_sfragment_short_tl,
6711 intro         .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6712 imports       .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6713 loadmodules   .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6714 }
6715 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6716   \str_clear:N \l__document_structure_sfragment_id_str
6717   \str_clear:N \l__document_structure_sfragment_date_str
6718   \clist_clear:N \l__document_structure_sfragment_creators_clist
6719   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6720   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6721   \tl_clear:N \l__document_structure_sfragment_type_tl
6722   \tl_clear:N \l__document_structure_sfragment_short_tl
6723   \tl_clear:N \l__document_structure_sfragment_imports_tl
6724   \tl_clear:N \l__document_structure_sfragment_intro_tl
6725   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6726   \keys_set:nn { document-structure / sfragment } { #1 }
6727 }

```

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.

```

6728 \newif\if@mainmatter\@mainmattertrue
6729 \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.

```

6730 \keys_define:nn { document-structure / sectioning }{
6731   name      .str_set_x:N = \l__document_structure_sect_name_str  ,
6732   ref       .str_set_x:N = \l__document_structure_sect_ref_str   ,
6733   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6734   clear     .default:n   = {true}                                ,
6735   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
6736   num       .default:n   = {true}
6737 }
6738 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6739   \str_clear:N \l__document_structure_sect_name_str
6740   \str_clear:N \l__document_structure_sect_ref_str
6741   \bool_set_false:N \l__document_structure_sect_clear_bool
6742   \bool_set_false:N \l__document_structure_sect_num_bool
6743   \keys_set:nn { document-structure / sectioning } { #1 }
6744 }
6745 \newcommand\omdoc@sectioning[3][]{
6746   \l__document_structure_sect_args:n {#1 }
6747   \let\omdoc@sect@name\l__document_structure_sect_name_str
6748   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6749   \if@mainmatter% numbering not overridden by frontmatter, etc.
6750     \bool_if:NTF \l__document_structure_sect_num_bool {
6751       \sfragment@num{#2}{#3}
6752     }{

```

```

6753     \sfragment@nonum{#2}{#3}
6754   }
6755   \def\current@section@level{\omdoc@sect@name}
6756   \else
6757     \sfragment@nonum{#2}{#3}
6758   \fi
6759 }% 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.

```

6760 \newcommand\sfragment@redefine@addtocontents[1]{%
6761 %\edef\__document_structureimport{#1}%
6762 %\@for\@I:=\__document_structureimport\do{%
6763 %\edef\@path{\csname module@\@I @path\endcsname}%
6764 %\@ifundefined{tf@toc}\relax%
6765 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6766 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6767 %\def\addcontentsline##1##2##3{%
6768 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
6769 %\else% hyperref.sty not loaded
6770 %\def\addcontentsline##1##2##3{%
6771 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
6772 %\fi
6773 }% 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.

```

6774 \newenvironment{sfragment}[2][ ]% keys, title
6775 {
6776   \__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.

```

6777   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6778
6779   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6780     \sfragment@redefine@addtocontents{
6781       %\@ifundefined{module@id}\used@modules%
6782       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6783     }
6784   }

```

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

```

6785
6786   \stex_document_title:n { #2 }
6787
6788   \int_incr:N\l__document_structure_section_level_int
6789   \ifcase\l__document_structure_section_level_int
6790     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6791     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6792     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6793     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

6794 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6795 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
6796 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#1}
6797 \fi
6798 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
6799 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6800   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6801 }
6802 }% for customization
6803 {}

```

and finally, we localize the sections

```

6804 \newcommand\omdoc@part@kw{Part}
6805 \newcommand\omdoc@chapter@kw{Chapter}
6806 \newcommand\omdoc@section@kw{Section}
6807 \newcommand\omdoc@subsection@kw{Subsection}
6808 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6809 \newcommand\omdoc@paragraph@kw{paragraph}
6810 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

36.3 Front and Backmatter

Index markup is provided by the `omtext` package [Kohlhase:smmtf:git], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

6811 \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).

```

6812 \cs_if_exist:NTF\frontmatter{
6813   \let\__document_structure_orig_frontmatter\frontmatter
6814   \let\frontmatter\relax
6815 }{
6816   \tl_set:Nn\__document_structure_orig_frontmatter{
6817     \clearpage
6818     \@mainmatterfalse
6819     \pagenumbering{roman}
6820   }
6821 }
6822 \cs_if_exist:NTF\backmatter{
6823   \let\__document_structure_orig_backmatter\backmatter
6824   \let\backmatter\relax
6825 }{
6826   \tl_set:Nn\__document_structure_orig_backmatter{
6827     \clearpage
6828     \@mainmatterfalse
6829     \pagenumbering{roman}
6830   }

```

```
6831 }
```

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.

```
6832 \newenvironment{frontmatter}{
6833   \_document_structure_orig_frontmatter
6834 }{
6835   \cs_if_exist:NTF\mainmatter{
6836     \mainmatter
6837   }{
6838     \clearpage
6839     \@mainmattertrue
6840     \pagenumbering{arabic}
6841   }
6842 }
```

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

```
6843 \newenvironment{backmatter}{
6844   \_document_structure_orig_backmatter
6845 }{
6846   \cs_if_exist:NTF\mainmatter{
6847     \mainmatter
6848   }{
6849     \clearpage
6850     \@mainmattertrue
6851     \pagenumbering{arabic}
6852   }
6853 }
```

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

```
6854 \@mainmattertrue\pagenumbering{arabic}
```

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

```
6855 \def \c__document_structure_document_str{document}
6856 \newcommand\afterprematurestop{}
6857 \def\prematurestop@endsfragment{
6858   \unless\ifx\@currenvir\c__document_structure_document_str
6859     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6860       \expandafter\prematurestop@endsfragment
6861     }fi
6862 }
6863 \providecommand\prematurestop{
6864   \message{Stopping~sTeX~processing~prematurely}
6865   \prematurestop@endsfragment
6866   \afterprematurestop
6867   \end{document}
6868 }
```

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

36.4 Global Variables

\setSGvar set a global variable

```
6869 \RequirePackage{etoolbox}
6870 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
6871 \newrobustcmd\useSGvar[1]{%
6872   \@ifundefined{sTeX@Gvar@#1}
6873   {\PackageError{document-structure}
6874     {The sTeX Global variable #1 is undefined}
6875     {set it with \protect\setSGvar}}
6876   \@nameuse{sTeX@Gvar@#1}}
```

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

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

```
6877 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6878   \@ifundefined{sTeX@Gvar@#1}
6879   {\PackageError{document-structure}
6880     {The sTeX Global variable #1 is undefined}
6881     {set it with \protect\setSGvar}}
6882   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

Chapter 37

NotesSlides – Implementation

37.1 Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` package. We pass the `nontheorem` option to the `statements` package when we are not in notes mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
6883 \*cls)
6884 \@@=notesslides)
6885 \ProvidesExplClass{notesslides}{2022/05/24}{3.1.0}{notesslides Class}
6886 \RequirePackage{13keys2e}
6887
6888 \keys_define:nn{notesslides / cls}{
6889   class   .str_set_x:N = \c__notesslides_class_str,
6890   notes   .bool_set:N = \c__notesslides_notes_bool ,
6891   slides  .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6892   docopt  .str_set_x:N = \c__notesslides_docopt_str,
6893   unknown .code:n      = {
6894     \PassOptionsToPackage{\CurrentOption}{document-structure}
6895     \PassOptionsToClass{\CurrentOption}{beamer}
6896     \PassOptionsToPackage{\CurrentOption}{notesslides}
6897     \PassOptionsToPackage{\CurrentOption}{stex}
6898   }
6899 }
6900 \ProcessKeysOptions{ notesslides / cls }
6901
6902 \str_if_empty:NF \c__notesslides_class_str {
6903   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6904 }
6905
6906 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6907   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6908 }
6909 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6910   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6911 }
6912
6913 \RequirePackage{stex}
```

```

6914 \stex_html_backend:T {
6915   \bool_set_true:N\c__notesslides_notes_bool
6916 }
6917
6918 \bool_if:NTF \c__notesslides_notes_bool {
6919   \PassOptionsToPackage{notes=true}{notesslides}
6920   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
6921 }{
6922   \PassOptionsToPackage{notes=false}{notesslides}
6923   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
6924 }
6925 </cls>

```

now we do the same for the notesslides package.

```

6926 <*package>
6927 \ProvidesExplPackage{notesslides}{2022/05/24}{3.1.0}{notesslides Package}
6928 \RequirePackage{l3keys2e}
6929
6930 \keys_define:nn{notesslides / pkg}{
6931   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6932   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6933   notes            .bool_set:N = \c__notesslides_notes_bool ,
6934   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6935   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
6936   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
6937   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
6938   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
6939   unknown          .code:n      = {
6940     \PassOptionsToClass{\CurrentOption}{stex}
6941     \PassOptionsToClass{\CurrentOption}{tikzinput}
6942   }
6943 }
6944 \ProcessKeysOptions{ notesslides / pkg }
6945
6946 \RequirePackage{stex}
6947 \stex_html_backend:T {
6948   \bool_set_true:N\c__notesslides_notes_bool
6949 }
6950
6951 \newif\ifnotes
6952 \bool_if:NTF \c__notesslides_notes_bool {
6953   \notesttrue
6954 }{
6955   \notesfalse
6956 }
6957

```

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

```

6958 \str_if_empty:NTF \c__notesslides_topsect_str {
6959   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6960 }{
6961   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6962 }
6963 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
6964 </package>
```

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

```
6965 <*cls>
6966 \bool_if:NTF \c__notesslides_notes_bool {
6967   \str_if_empty:NT \c__notesslides_class_str {
6968     \str_set:Nn \c__notesslides_class_str {article}
6969   }
6970   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
6971     {\c__notesslides_class_str}
6972 }{
6973   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6974   \newcounter{Item}
6975   \newcounter{paragraph}
6976   \newcounter{subparagraph}
6977   \newcounter{Hfootnote}
6978 }
6979 \RequirePackage{document-structure}
```

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

```
6980 \RequirePackage{notesslides}
6981 </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 `STEX` packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the `STEX`-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.

```
6982 <*package>
6983 \bool_if:NT \c__notesslides_notes_bool {
6984   \RequirePackage{a4wide}
6985   \RequirePackage{marginnote}
6986   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6987   \RequirePackage{mdframed}
6988   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6989   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6990 }
6991 \RequirePackage{stex-tikzinput}
6992 \RequirePackage{comment}
6993 \RequirePackage{url}
6994 \RequirePackage{graphicx}
6995 \RequirePackage{pgf}
6996 \RequirePackage{bookmark}
```

37.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class.

```
6997 \bool_if:NT \c__notesslides_notes_bool {
6998   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
6999 }
```

```

7000 \NewDocumentCommand \libusetheme {0{} m} {
7001   \libusepackage[#1]{beamertheme#2}
7002 }
7003

```

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

```

7004 \newcounter{slide}
7005 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7006 \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.

```

7007 \bool_if:NTF \c__notesslides_notes_bool {
7008   \renewenvironment{note}{\ignorespaces}{}
7009 }{
7010   \excludecomment{note}
7011 }

```

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.

```

7012 \bool_if:NT \c__notesslides_notes_bool {
7013   \newlength{\slideframewidth}
7014   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

7015 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7016   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7017     \bool_set_true:N #1
7018   }{
7019     \bool_set_false:N #1
7020   }
7021 }
7022 \keys_define:nn{notesslides / frame}{
7023   label .str_set_x:N = \l__notesslides_frame_label_str,
7024   allowframebreaks .code:n = {
7025     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7026   },
7027   allowdisplaybreaks .code:n = {
7028     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7029   },
7030   fragile .code:n = {
7031     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7032   },
7033   shrink .code:n = {
7034     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7035   },
7036   squeeze .code:n = {
7037     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7038   },
7039   t .code:n = {

```

```

7040     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7041   },
7042   unknown    .code:n      = {}
7043 }
7044 \cs_new_protected:Nn \__notesslides_frame_args:n {
7045   \str_clear:N \l__notesslides_frame_label_str
7046   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7047   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7048   \bool_set_true:N \l__notesslides_frame_fragile_bool
7049   \bool_set_true:N \l__notesslides_frame_shrink_bool
7050   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7051   \bool_set_true:N \l__notesslides_frame_t_bool
7052   \keys_set:nn { notesslides / frame }{ #1 }
7053 }

```

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

```

7054 \renewenvironment{frame}[1][]{
7055   \__notesslides_frame_args:n{#1}
7056   \sffamily
7057   \stepcounter{slide}
7058   \def\@currentlabel{\theslide}
7059   \str_if_empty:NF \l__notesslides_frame_label_str {
7060     \label{\l__notesslides_frame_label_str}
7061   }

```

We redefine the `itemize` environment so that it looks more like the one in `beamer`.

```

7062   \def\itemize@level{outer}
7063   \def\itemize@outer{outer}
7064   \def\itemize@inner{inner}
7065   \renewcommand\newpage{\addtocounter{framenum}{1}}
7066   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7067   \renewenvironment{itemize}{
7068     \ifx\itemize@level\itemize@outer
7069       \def\itemize@label{$\rhd$}
7070     \fi
7071     \ifx\itemize@level\itemize@inner
7072       \def\itemize@label{$\scriptstyle\rhd$}
7073     \fi
7074     \begin{list}
7075       {\itemize@label}
7076       {\setlength{\labelsep}{.3em}
7077        \setlength{\labelwidth}{.5em}
7078        \setlength{\leftmargin}{1.5em}
7079       }
7080     \edef\itemize@level{\itemize@inner}
7081   }{
7082     \end{list}
7083   }

```

We create the box with the `mdframed` environment from the `equinymous` package.

```

7084   \stex_html_backend:TF {
7085     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7086     \mdf@patchamsthm
7087   }{
7088     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7089     }
7090   }{
7091     \stex_html_backend:TF {
7092       \miko@slidelabel\egroup\end{stex_annotate_env}
7093     }\medskip\miko@slidelabel\end{mdframed}}
7094   }

```

Now, we need to redefine the frametitle (we are still in course notes mode).

\frametitle

```

7095   \renewcommand{\frametitle}[1]{
7096     \stex_document_title:n { #1 }
7097     {\Large\bf\sf\color{blue}{#1}}\medskip
7098   }
7099 }

```

(End definition for \frametitle. This function is documented on page ??.)

EdN:10

\pause 10

```

7100 \bool_if:NT \c__notesslides_notes_bool {
7101   \newcommand\pause{}
7102 }

```

(End definition for \pause. This function is documented on page ??.)

nparagraph

```

7103 \bool_if:NTF \c__notesslides_notes_bool {
7104   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7105 }{
7106   \excludecomment{nparagraph}
7107 }

```

nfragment

```

7108 \bool_if:NTF \c__notesslides_notes_bool {
7109   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
7110 }{
7111   \excludecomment{nfragment}
7112 }

```

ndefinition

```

7113 \bool_if:NTF \c__notesslides_notes_bool {
7114   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7115 }{
7116   \excludecomment{ndefinition}
7117 }

```

nassertion

```

7118 \bool_if:NTF \c__notesslides_notes_bool {
7119   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7120 }{
7121   \excludecomment{nassertion}
7122 }

```

¹⁰EDNOTE: MK: fake it in notes mode for now

nsproof

```
7123 \bool_if:NTF \c__notesslides_notes_bool {
7124   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7125 }{
7126   \excludecomment{nproof}
7127 }
```

nexample

```
7128 \bool_if:NTF \c__notesslides_notes_bool {
7129   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7130 }{
7131   \excludecomment{nexample}
7132 }
```

\inputref@*skip We customize the hooks for in \inputref.

```
7133 \def\inputref@preskip{\smallskip}
7134 \def\inputref@postskip{\medskip}
```

(End definition for \inputref@*skip. This function is documented on page ??.)

\inputref*

```
7135 \let\orig@inputref\inputref
7136 \def\inputref{\@ifstar\ninputref\orig@inputref}
7137 \newcommand\ninputref[2] []{
7138   \bool_if:NT \c__notesslides_notes_bool {
7139     \orig@inputref[#1]{#2}
7140   }
7141 }
```

(End definition for \inputref*. This function is documented on page 56.)

37.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

\setslidelogo The default logo is the \TeX logo. Customization can be done by \setslidelogo{<logo name>}.

```
7142 \newlength{\slidelogoheight}
7143
7144 \RequirePackage{graphicx}
7145
7146 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7147 \providecommand\mhgraphics[2] []{
7148   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7149   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7150 }
7151
7152 \bool_if:NTF \c__notesslides_notes_bool {
7153   \setlength{\slidelogoheight}{.4cm}
7154 }{
7155   \setlength{\slidelogoheight}{.25cm}
7156 }
```

```

7157 \ifcsname slidelogo\endcsname\else
7158   \newsavebox{\slidelogo}
7159   \sbox{\slidelogo}{\sTeX}
7160 \fi
7161 \newrobustcmd{\setslidelogo}[2][]{
7162   \tl_if_empty:nTF{#1}{
7163     \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7164   }{
7165     \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7166   }
7167 }

```

(End definition for `\setslidelogo`. This function is documented on page 57.)

\author In notes mode, we redefine the `\author` macro so that it does not disregard the optional argument (as `beamerarticle` does). We want to use it to set the source later.

```

7168 \bool_if:NT \c__notesslides_notes_bool {
7169   \def\author{\@dblarg\ns@author}
7170   \long\def\ns@author[#1]#2{%
7171     \def\c__notesslides_shortauthor{#1}%
7172     \def\@author{#2}
7173   }
7174 }

```

(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.

```

7175 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

(End definition for `\setsource`. This function is documented on page 57.)

\setlicensing Now, we set up the copyright and licensing. By default we use the Creative Commons Attribution-ShareAlike license to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

```

7176 \def\copyrightnotice{%
7177   \footnotesize\copyright : \hspace{.3ex}%
7178   \ifcsname source\endcsname\source\else%
7179   \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7180   \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7181   ?source/author?\fi%
7182 \fi}
7183 \newsavebox{\cclogo}
7184 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7185 \newif\ifcchref\cchreffalse
7186 \AtBeginDocument{
7187   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7188 }
7189 \def\licensing{
7190   \ifcchref
7191     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7192   \else
7193     {\usebox{\cclogo}}

```



```

7194 \fi
7195 }
7196 \newrobustcmd{\setlicensing}[2][]{
7197   \def\@url{#1}
7198   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7199   \ifx\@url\@empty
7200     \def\licensing{\usebox{\cclogo}}
7201   \else
7202     \def\licensing{
7203       \ifcchref
7204         \href{#1}{\usebox{\cclogo}}
7205       \else
7206         {\usebox{\cclogo}}
7207     \fi
7208   }
7209 \fi
7210 }

```

(End definition for \setlicensing. This function is documented on page 57.)

EdN:11

```

\slidelabel Now, we set up the slide label for the article mode.11
7211 \newrobustcmd\miko@slidelabel{
7212   \vbox to \slidelogoheight{
7213     \vss\hbox to \slidewidth
7214       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7215   }
7216 }

```

(End definition for \slidelabel. This function is documented on page ??.)

37.4 Frame Images

\frameimage We have to make sure that the width is overwritten, for that we check the \Gin@ewidth macro from the graphicx package. We also add the label key.

```

7217 \def\Gin@mhrepos{}
7218 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7219 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7220 \newrobustcmd\frameimage[2][]{
7221   \stepcounter{slide}
7222   \bool_if:NT \c__notesslides_frameimages_bool {
7223     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7224     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7225     \begin{center}
7226       \bool_if:NTF \c__notesslides_fiboxed_bool {
7227         \fbox{
7228           \ifx\Gin@ewidth\@empty
7229             \ifx\Gin@mhrepos\@empty
7230               \mhgraphics[width=\slidewidth,#1]{#2}
7231             \else
7232               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7233             \fi
7234           \else% Gin@ewidth empty

```

¹¹EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

7235         \ifx\Gin@mhrepos\@empty
7236         \mhgraphics[#1]{#2}
7237     \else
7238         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7239     \fi
7240 \fi% Gin@ewidth empty
7241 }
7242 }{
7243     \ifx\Gin@ewidth\@empty
7244     \ifx\Gin@mhrepos\@empty
7245         \mhgraphics[width=\slidewidth,#1]{#2}
7246     \else
7247         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7248     \fi
7249     \ifx\Gin@mhrepos\@empty
7250         \mhgraphics[#1]{#2}
7251     \else
7252         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7253     \fi
7254     \fi% Gin@ewidth empty
7255 }
7256 \end{center}
7257 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7258 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7259 }
7260 } % ifmks@sty@frameimages

```

(End definition for `\frameimage`. This function is documented on page 57.)

37.5 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```

7261 \stex_html_backend:F {
7262     \bool_if:NT \c__notesslides_sectocframes_bool {
7263         \str_if_eq:VnTF \__notesslidestopsect{part}{
7264             \newcounter{chapter}\counterwithin*{section}{chapter}
7265         }{
7266             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7267                 \newcounter{chapter}\counterwithin*{section}{chapter}
7268             }
7269         }
7270     }
7271 }

```

`\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
7272 \def\part@prefix{}
7273 \@ifpackageloaded{document-structure}{
7274     \str_case:VnF \__notesslidestopsect {

```

```

7275 {part}{
7276   \int_set:Nn \l_document_structure_section_level_int {0}
7277   \def\thesection{\arabic{chapter}.\arabic{section}}
7278   \def\part@prefix{\arabic{chapter}.}
7279 }
7280 {chapter}{
7281   \int_set:Nn \l_document_structure_section_level_int {1}
7282   \def\thesection{\arabic{chapter}.\arabic{section}}
7283   \def\part@prefix{\arabic{chapter}.}
7284 }
7285 }{
7286   \int_set:Nn \l_document_structure_section_level_int {2}
7287   \def\part@prefix{}
7288 }
7289 }
7290
7291 \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`

```

7292 \renewenvironment{sfragment}[2][]{
7293   \__document_structure_sfragment_args:n { #1 }
7294   \int_incr:N \l_document_structure_section_level_int
7295   \bool_if:NT \c__notesslides_sectocframes_bool {
7296     \stepcounter{slide}
7297     \begin{frame}[noframenumbering]
7298     \vfill\Large\centering
7299     \red{
7300       \ifcase\l_document_structure_section_level_int\or
7301         \stepcounter{part}
7302         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7303         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7304         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7305         \def\currentsectionlevel{\omdoc@part@kw}
7306       \or
7307         \stepcounter{chapter}
7308         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7309         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7310         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7311         \def\currentsectionlevel{\omdoc@chapter@kw}
7312       \or
7313         \stepcounter{section}
7314         \def\__notesslideslabel{\part@prefix\arabic{section}}
7315         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7316         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7317         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7318         \def\currentsectionlevel{\omdoc@section@kw}
7319       \or
7320         \stepcounter{subsection}
7321         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7322         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

7323         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7324         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7325         \def\currentsectionlevel{\omdoc@subsection@kw}
7326     \or
7327         \stepcounter{subsubsection}
7328         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7329         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7330         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7331         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7332         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7333     \or
7334         \stepcounter{paragraph}
7335         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7336         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7337         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7338         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7339         \def\currentsectionlevel{\omdoc@paragraph@kw}
7340     \else
7341         \def\__notesslideslabel{}
7342         \def\currentsectionlevel{\omdoc@paragraph@kw}
7343     \fi% end ifcase
7344     \__notesslideslabel\quad #2%
7345 }%
7346 \vfill%
7347 \end{frame}%
7348 }
7349 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7350     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7351 }
7352 }{}
7353 }

```

We set up a beamer template for theorems like ams style, but without a block environment.

```

7354 \def\inserttheorembodyfont{\normalfont}
7355 %\bool_if:NF \c__notesslides_notes_bool {
7356 % \defbeamertemplate{theorem begin}{miko}
7357 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7358 % \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7359 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7360 % \defbeamertemplate{theorem end}{miko}{\fi}

```

and we set it as the default one.

```

7361 % \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.

```

7362 % \expandafter\def\csname Parent2\endcsname{}
7363 %}
7364
7365 \AddToHook{begindocument}{ % this does not work for some reason
7366     \setbeamertemplate{theorems}[ams style]
7367 }
7368 \bool_if:NT \c__notesslides_notes_bool {
7369     \renewenvironment{columns}[1][\fi]{}{}

```

```

7370     \par\noindent%
7371     \begin{minipage}%
7372     \slidewidth\centering\leavevmode%
7373   }{%
7374     \end{minipage}\par\noindent%
7375   }%
7376   \newsavebox\columnbox%
7377   \renewenvironment<>{column}[2][]{%
7378     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7379   }{%
7380     \end{minipage}\end{lrbox}\usebox\columnbox%
7381   }%
7382 }

7383 \bool_if:NTF \c__notesslides_noproblems_bool {
7384   \newenvironment{problems}{}{}
7385 }{
7386   \excludacomment{problems}
7387 }

```

37.6 Excursions

\excursion The excursion macros are very simple, we define a new internal macro `\excursionref` and use it in `\excursion`, which is just an `\inputref` that checks if the new macro is defined before formatting the file in the argument.

```

7388 \gdef\printexcursions{}
7389 \newcommand\excursionref[2]{% label, text
7390   \bool_if:NT \c__notesslides_notes_bool {
7391     \begin{sparagraph}[title=Excursion]
7392       #2 \sref[fallback=the appendix]{#1}.
7393     \end{sparagraph}
7394   }
7395 }
7396 \newcommand\activate@excursion[2][{}{
7397   \gappto\printexcursions{\inputref{#1}{#2}}
7398 }
7399 \newcommand\excursion[4][{}{ repos, label, path, text
7400   \bool_if:NT \c__notesslides_notes_bool {
7401     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7402   }
7403 }

```

(End definition for `\excursion`. This function is documented on page 58.)

\excursiongroup

```

7404 \keys_define:nn{notesslides / excursiongroup }{
7405   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7406   intro       .tl_set:N    = \l__notesslides_excursion_intro_tl,
7407   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7408 }
7409 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7410   \tl_clear:N \l__notesslides_excursion_intro_tl
7411   \str_clear:N \l__notesslides_excursion_id_str

```

```

7412 \str_clear:N \l__notesslides_excursion_mhrepos_str
7413 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7414 }
7415 \newcommand\excursionsgroup[1][ ]{
7416   \__notesslides_excursion_args:n{ #1 }
7417   \ifdefempty\printexcursions{}% only if there are excursions
7418   {\begin{note}
7419     \begin{sfragment}[#1]{Excursions}%
7420     \ifdefempty\l__notesslides_excursion_intro_tl}{
7421       \inputref[\l__notesslides_excursion_mhrepos_str]{
7422         \l__notesslides_excursion_intro_tl
7423       }
7424     }
7425     \printexcursions%
7426     \end{sfragment}
7427   \end{note}}
7428 }
7429 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7430 \end{package}

```

(End definition for \excursionsgroup. This function is documented on page 58.)

Chapter 38

The Implementation

38.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
7431 <*package>
7432 <@@=problems>
7433 \ProvidesExplPackage{problem}{2022/05/24}{3.1.0}{Semantic Markup for Problems}
7434 \RequirePackage{13keys2e}
7435 \RequirePackage{amssymb}% for \Box
7436
7437 \keys_define:nn { problem / pkg }{
7438   notes      .default:n    = { true },
7439   notes      .bool_set:N   = \c__problems_notes_bool,
7440   gnotes     .default:n    = { true },
7441   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7442   hints      .default:n    = { true },
7443   hints      .bool_set:N   = \c__problems_hints_bool,
7444   solutions  .default:n    = { true },
7445   solutions  .bool_set:N   = \c__problems_solutions_bool,
7446   pts        .default:n    = { true },
7447   pts        .bool_set:N   = \c__problems_pts_bool,
7448   min        .default:n    = { true },
7449   min        .bool_set:N   = \c__problems_min_bool,
7450   boxed      .default:n    = { true },
7451   boxed      .bool_set:N   = \c__problems_boxed_bool,
7452   unknown    .code:n       = {
7453     \PassOptionsToPackage{\CurrentOption}{stex}
7454   }
7455 }
7456 \newif\ifsolutions
7457
7458 \ProcessKeysOptions{ problem / pkg }
7459 \bool_if:NTF \c__problems_solutions_bool {
7460   \solutionstrue
7461 }{
7462   \solutionsfalse
```

```

7463 }
7464 \RequirePackage{stex}

```

Then we make sure that the necessary packages are loaded (in the right versions).

```

7465 \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.

```

7466 \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.

```

7467 \def\prob@problem@kw{Problem}
7468 \def\prob@solution@kw{Solution}
7469 \def\prob@hint@kw{Hint}
7470 \def\prob@note@kw{Note}
7471 \def\prob@gnote@kw{Grading}
7472 \def\prob@pt@kw{pt}
7473 \def\prob@min@kw{min}
7474 \def\prob@correct@kw{Correct}
7475 \def\prob@wrong@kw{Wrong}

```

(End definition for `\prob@*@kw`. This function is documented on page ??.)

For the other languages, we set up triggers

```

7476 \AddToHook{begindocument}{
7477   \ltx@ifpackageloaded{babel}{
7478     \makeatletter
7479     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7480     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7481       \input{problem-ngerman.ldf}
7482     }
7483     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7484       \input{problem-finnish.ldf}
7485     }
7486     \clist_if_in:NnT \l_tmpa_clist {french}{
7487       \input{problem-french.ldf}
7488     }
7489     \clist_if_in:NnT \l_tmpa_clist {russian}{
7490       \input{problem-russian.ldf}
7491     }
7492     \makeatother
7493   }{}
7494 }

```

38.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```

7495 \keys_define:nn{ problem / problem }{
7496   id      .str_set_x:N = \l__problems_prob_id_str,
7497   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7498   min     .tl_set:N    = \l__problems_prob_min_tl,

```



```

7499 title .tl_set:N = \l__problems_prob_title_tl,
7500 type .tl_set:N = \l__problems_prob_type_tl,
7501 imports .tl_set:N = \l__problems_prob_imports_tl,
7502 name .str_set_x:N = \l__problems_prob_name_str,
7503 refnum .int_set:N = \l__problems_prob_refnum_int
7504 }
7505 \cs_new_protected:Nn \l__problems_prob_args:n {
7506 \str_clear:N \l__problems_prob_id_str
7507 \str_clear:N \l__problems_prob_name_str
7508 \tl_clear:N \l__problems_prob_pts_tl
7509 \tl_clear:N \l__problems_prob_min_tl
7510 \tl_clear:N \l__problems_prob_title_tl
7511 \tl_clear:N \l__problems_prob_type_tl
7512 \tl_clear:N \l__problems_prob_imports_tl
7513 \int_zero_new:N \l__problems_prob_refnum_int
7514 \keys_set:nn { problem / problem }{ #1 }
7515 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7516 \let\l__problems_prob_refnum_int\undefined
7517 }
7518 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7519 \newcounter{problem}[section]
7520 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
7521 \def\theplainsproblem{\arabic{problem}}
7522 \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.

```

7523 \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

```

7524 \newcommand\prob@number{
7525 \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7526 \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7527 }{
7528 \int_if_exist:NTF \l__problems_prob_refnum_int {
7529 \prob@label{\int_use:N \l__problems_prob_refnum_int }
7530 }{
7531 \prob@label\theplainsproblem
7532 }
7533 }
7534 }
7535 \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.

```

7536 \newcommand\prob@title[3]{%
7537   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7538     #2 \l__problems_inclprob_title_tl #3
7539   }{
7540     \tl_if_empty:NTF \l__problems_prob_title_tl {
7541       #1
7542     }{
7543       #2 \l__problems_prob_title_tl #3
7544     }
7545   }
7546 }

```

(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.

```

7547 \def\prob@heading{
7548   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7549   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
7550 }

```

(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

```

7551 \newenvironment{sproblem}[1][]{
7552   \__problems_prob_args:n{#1}%\sref@target%
7553   \@in@omtexttrue% we are in a statement (for inline definitions)
7554   \refstepcounter{sproblem}\record@problem
7555   \def\current@section@level{\prob@problem@kw}
7556
7557   \str_if_empty:NT \l__problems_prob_name_str {
7558     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7559     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7560     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7561   }
7562
7563   \stex_if_do_html:T{
7564     \tl_if_empty:NF \l__problems_prob_title_tl {
7565       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7566     }
7567   }
7568
7569   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7570
7571   \stex_reactivate_macro:N \STEXexport
7572   \stex_reactivate_macro:N \importmodule
7573   \stex_reactivate_macro:N \symdecl
7574   \stex_reactivate_macro:N \notation
7575   \stex_reactivate_macro:N \symdef

```

```

7576 \stex_if_do_html:T{
7577   \begin{stex_annotate_env} {problem} {
7578     \l_stex_module_ns_str ? \l_stex_module_name_str
7579   }
7580 }
7581
7582 \stex_annotate_invisible:nnn{header}{} {
7583   \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
7584   \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7585   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7586     \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7587   }
7588 }
7589 }
7590
7591 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7592
7593
7594 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7595   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7596 }{
7597   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7598 }
7599 \str_if_exist:NTF \l__problems_inclprob_id_str {
7600   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7601 }{
7602   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7603 }
7604
7605
7606 \stex_if_smsmode:F {
7607   \clist_set:No \l_tmpa_clist \sproblemtype
7608   \tl_clear:N \l_tmpa_tl
7609   \clist_map_inline:Nn \l_tmpa_clist {
7610     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7611       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7612     }
7613   }
7614   \tl_if_empty:NTF \l_tmpa_tl {
7615     \__problems_sproblem_start:
7616   }{
7617     \l_tmpa_tl
7618   }
7619 }
7620 \stex_ref_new_doc_target:n \sproblemid
7621 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
7622 }{
7623   \__stex_modules_end_module:
7624   \stex_if_smsmode:F{
7625     \clist_set:No \l_tmpa_clist \sproblemtype
7626     \tl_clear:N \l_tmpa_tl
7627     \clist_map_inline:Nn \l_tmpa_clist {
7628       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7629         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}

```

```

7630     }
7631   }
7632   \tl_if_empty:NTF \l_tmpa_tl {
7633     \__problems_sproblem_end:
7634   }{
7635     \l_tmpa_tl
7636   }
7637 }
7638 \stex_if_do_html:T{
7639   \end{stex_annotate_env}
7640 }
7641
7642 \smallskip
7643 }
7644
7645 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7646
7647
7648
7649 \cs_new_protected:Nn \__problems_sproblem_start: {
7650   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7651 }
7652 \cs_new_protected:Nn \__problems_sproblem_end: { \par\smallskip}
7653
7654 \newcommand\stexpatchproblem[3][] {
7655   \str_set:Nx \l_tmpa_str{ #1 }
7656   \str_if_empty:NTF \l_tmpa_str {
7657     \tl_set:Nn \__problems_sproblem_start: { #2 }
7658     \tl_set:Nn \__problems_sproblem_end: { #3 }
7659   }{
7660     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7661     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7662   }
7663 }
7664
7665
7666 \bool_if:NT \c__problems_boxed_bool {
7667   \surroundwithhmdframed{problem}
7668 }

```

\record@problem This macro records information about the problems in the *.aux file.

```

7669 \def\record@problem{
7670   \protected@write\@auxout{}
7671   {
7672     \string\@problem{\prob@number}
7673     {
7674       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7675         \l__problems_inclprob_pts_tl
7676       }{
7677         \l__problems_prob_pts_tl
7678       }
7679     }%
7680     {
7681       \tl_if_exist:NTF \l__problems_inclprob_min_tl {

```

```

7682         \l__problems_inclprob_min_tl
7683     }{
7684         \l__problems_prob_min_tl
7685     }
7686 }
7687 }
7688 }

```

(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).

```

7689 \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.

```

7690 \keys_define:nn { problem / solution }{
7691     id          .str_set_x:N = \l__problems_solution_id_str ,
7692     for         .str_set_x:N = \l__problems_solution_for_str ,
7693     type        .str_set_x:N = \l__problems_solution_type_str ,
7694     title       .tl_set:N    = \l__problems_solution_title_tl
7695 }
7696 \cs_new_protected:Nn \__problems_solution_args:n {
7697     \str_clear:N \l__problems_solution_id_str
7698     \str_clear:N \l__problems_solution_type_str
7699     \str_clear:N \l__problems_solution_for_str
7700     \tl_clear:N \l__problems_solution_title_tl
7701     \keys_set:nn { problem / solution }{ #1 }
7702 }

```

\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.

```

7703 \box_new:N \l__problems_solution_box
7704 \newenvironment{solution}[1][{}]{
7705     \__problems_solution_args:n{#1}
7706     \stex_html_backend:TF{
7707         \stex_if_do_html:T{
7708             \begin{stex_annotate_env}{solution}{}}
7709             \str_if_empty:NF \l__problems_solution_type_str {
7710                 \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
7711             }
7712             \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
7713         }
7714     }{
7715         \setbox\l__problems_solution_box\vbox\bgroup
7716         \par\smallskip\hrule\smallskip
7717         \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
7718     }
7719 }{
7720     \stex_html_backend:TF{
7721         \stex_if_do_html:T{
7722             \end{stex_annotate_env}

```

```

7723     }
7724   }{
7725     \smallskip\hrule
7726     \egroup
7727     \bool_if:NT \c__problems_solutions_bool {
7728       \box\l__problems_solution_box
7729     }
7730   }
7731 }
7732
7733 \newcommand\startsolutions{
7734   \bool_set_true:N \c__problems_solutions_bool
7735   \solutionstrue
7736   % \specialcomment{solution}{\@startsolution}{
7737   %   \bool_if:NF \c__problems_boxed_bool {
7738   %     \hrule\medskip
7739   %   }
7740   %   \end{small}%
7741   % }
7742   % \bool_if:NT \c__problems_boxed_bool {
7743   %   \surroundwithmdframed{solution}
7744   % }
7745 }

```

(End definition for \startsolutions. This function is documented on page 60.)

\stopsolutions

```

7746 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

(End definition for \stopsolutions. This function is documented on page 60.)

exnote

```

7747 \bool_if:NTF \c__problems_notes_bool {
7748   \newenvironment{exnote}[1][]{
7749     \par\smallskip\hrule\smallskip
7750     \noindent\textbf{\prob@note@kw :~ }\small
7751   }{
7752     \smallskip\hrule
7753   }
7754 }{
7755   \excludacomment{exnote}
7756 }

```

hint

```

7757 \bool_if:NTF \c__problems_notes_bool {
7758   \newenvironment{hint}[1][]{
7759     \par\smallskip\hrule\smallskip
7760     \noindent\textbf{\prob@hint@kw :~ }\small
7761   }{
7762     \smallskip\hrule
7763   }
7764 \newenvironment{exhint}[1][]{
7765   \par\smallskip\hrule\smallskip
7766   \noindent\textbf{\prob@hint@kw :~ }\small

```

```

7767 }{
7768     \smallskip\hrule
7769 }
7770 }{
7771     \excludecomment{hint}
7772     \excludecomment{exhint}
7773 }

```

gnote

```

7774 \bool_if:NTF \c__problems_notes_bool {
7775     \newenvironment{gnote}[1][]{
7776         \par\smallskip\hrule\smallskip
7777         \noindent\textbf{\prob@gnote@kw :~ }\small
7778     }{
7779         \smallskip\hrule
7780     }
7781 }{
7782     \excludecomment{gnote}
7783 }

```

38.3 Marup for Added Value Services

38.4 Multiple Choice Blocks

EdN:12

mcb

12

```

7784 \newenvironment{mcb}{
7785     \begin{enumerate}
7786 }{
7787     \end{enumerate}
7788 }

```

we define the keys for the mcb macro

```

7789 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7790     \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7791         \bool_set_true:N #1
7792     }{
7793         \bool_set_false:N #1
7794     }
7795 }
7796 \keys_define:nn { problem / mcb }{
7797     id .str_set_x:N = \l__problems_mcc_id_str ,
7798     feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7799     T .default:n = { false } ,
7800     T .bool_set:N = \l__problems_mcc_t_bool ,
7801     F .default:n = { false } ,
7802     F .bool_set:N = \l__problems_mcc_f_bool ,
7803     Ttext .tl_set:N = \l__problems_mcc_Ttext_tl ,
7804     Ftext .tl_set:N = \l__problems_mcc_Ftext_tl
7805 }
7806 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

¹²EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7807 \str_clear:N \l__problems_mcc_id_str
7808 \tl_clear:N \l__problems_mcc_feedback_tl
7809 \bool_set_false:N \l__problems_mcc_t_bool
7810 \bool_set_false:N \l__problems_mcc_f_bool
7811 \tl_clear:N \l__problems_mcc_Ttext_tl
7812 \tl_clear:N \l__problems_mcc_Ftext_tl
7813 \str_clear:N \l__problems_mcc_id_str
7814 \keys_set:nn { problem / mcc }{ #1 }
7815 }

```

\mcc

```

7816 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
7817 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
7818 \newcommand\mcc[2][{}]{
7819   \l__problems_mcc_args:n{ #1 }
7820   \item[$\Box$] #2
7821   \bool_if:NT \c__problems_solutions_bool{
7822     \\\
7823     \bool_if:NT \l__problems_mcc_t_bool {
7824       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7825     }
7826     \bool_if:NT \l__problems_mcc_f_bool {
7827       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7828     }
7829     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7830       \emph{\l__problems_mcc_feedback_tl}
7831     }
7832   }
7833 } %solutions

```

(End definition for \mcc. This function is documented on page 61.)

38.5 Filling in Concrete Solutions

\includeproblem This is embarrassingly simple, but can grow over time.

```

7834 \newcommand\fillinsol[1]{\quad%
7835   \ifsolutions\textcolor{red}{\#!}\else%
7836   \fbox{\phantom{\huge{\#!}}}%
7837   \fi}

```

(End definition for \includeproblem. This function is documented on page 63.)

38.6 Including Problems

\includeproblem The \includeproblem command is essentially a glorified \input statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the inclprob keys after the input.

```

7838
7839 \keys_define:nn{ problem / inclproblem }{
7840   id      .str_set_x:N = \l__problems_inclprob_id_str,
7841   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7842   min     .tl_set:N    = \l__problems_inclprob_min_tl,

```



```

7843 title .tl_set:N = \l__problems_inclprob_title_tl,
7844 refnum .int_set:N = \l__problems_inclprob_refnum_int,
7845 type .tl_set:N = \l__problems_inclprob_type_tl,
7846 mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
7847 }
7848 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7849 \str_clear:N \l__problems_prob_id_str
7850 \tl_clear:N \l__problems_inclprob_pts_tl
7851 \tl_clear:N \l__problems_inclprob_min_tl
7852 \tl_clear:N \l__problems_inclprob_title_tl
7853 \tl_clear:N \l__problems_inclprob_type_tl
7854 \int_zero_new:N \l__problems_inclprob_refnum_int
7855 \str_clear:N \l__problems_inclprob_mhrepos_str
7856 \keys_set:nn { problem / inclproblem }{ #1 }
7857 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7858 \let\l__problems_inclprob_pts_tl\undefined
7859 }
7860 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7861 \let\l__problems_inclprob_min_tl\undefined
7862 }
7863 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7864 \let\l__problems_inclprob_title_tl\undefined
7865 }
7866 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7867 \let\l__problems_inclprob_type_tl\undefined
7868 }
7869 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7870 \let\l__problems_inclprob_refnum_int\undefined
7871 }
7872 }
7873
7874 \cs_new_protected:Nn \l__problems_inclprob_clear: {
7875 \let\l__problems_inclprob_id_str\undefined
7876 \let\l__problems_inclprob_pts_tl\undefined
7877 \let\l__problems_inclprob_min_tl\undefined
7878 \let\l__problems_inclprob_title_tl\undefined
7879 \let\l__problems_inclprob_type_tl\undefined
7880 \let\l__problems_inclprob_refnum_int\undefined
7881 \let\l__problems_inclprob_mhrepos_str\undefined
7882 }
7883 \l__problems_inclprob_clear:
7884
7885 \newcommand\includeproblem[2][]{
7886 \l__problems_inclprob_args:n{ #1 }
7887 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7888 \stex_html_backend:TF {
7889 \str_clear:N \l_tmpa_str
7890 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7891 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7892 }
7893 \stex_annotate_invisible:nnn{includeproblem}{
7894 \l_tmpa_str / #2
7895 }{}}
7896 }{

```

```

7897     \begingroup
7898     \inputreftrue
7899     \tl_if_empty:nTF{ ##1 }{
7900       \input{#2}
7901     }{
7902       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7903     }
7904   \endgroup
7905 }
7906 }
7907 \__problems_inclprob_clear:
7908 }

```

(End definition for `\includeproblem`. This function is documented on page 63.)

38.7 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```

7909 \AddToHook{enddocument}{
7910   \bool_if:NT \c__problems_pts_bool {
7911     \message{Total:~\arabic{pts}~points}
7912   }
7913   \bool_if:NT \c__problems_min_bool {
7914     \message{Total:~\arabic{min}~minutes}
7915   }
7916 }

```

The margin pars are reader-visible, so we need to translate

```

7917 \def\pts#1{
7918   \bool_if:NT \c__problems_pts_bool {
7919     \marginpar{#1~\prob@pt@kw}
7920   }
7921 }
7922 \def\min#1{
7923   \bool_if:NT \c__problems_min_bool {
7924     \marginpar{#1~\prob@min@kw}
7925   }
7926 }

```

`\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.

```

7927 \newcounter{pts}
7928 \def\show@pts{
7929   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7930     \bool_if:NT \c__problems_pts_bool {
7931       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7932       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7933     }
7934   }{
7935     \tl_if_exist:NT \l__problems_prob_pts_tl {
7936       \bool_if:NT \c__problems_pts_bool {

```

```

7937         \tl_if_empty:NT\l__problems_prob_pts_tl{
7938             \tl_set:Nn \l__problems_prob_pts_tl {0}
7939         }
7940         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7941         \addtocounter{pts}{\l__problems_prob_pts_tl}
7942     }
7943 }
7944 }
7945 }

```

(End definition for \show@pts. This function is documented on page ??.)
and now the same for the minutes

\show@min

```

7946 \newcounter{min}
7947 \def\show@min{
7948     \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7949         \bool_if:NT \c__problems_min_bool {
7950             \marginpar{\l__problems_inclprob_pts_tl\ min}
7951             \addtocounter{min}{\l__problems_inclprob_min_tl}
7952         }
7953     }{
7954         \tl_if_exist:NT \l__problems_prob_min_tl {
7955             \bool_if:NT \c__problems_min_bool {
7956                 \tl_if_empty:NT\l__problems_prob_min_tl{
7957                     \tl_set:Nn \l__problems_prob_min_tl {0}
7958                 }
7959                 \marginpar{\l__problems_prob_min_tl\ min}
7960                 \addtocounter{min}{\l__problems_prob_min_tl}
7961             }
7962         }
7963     }
7964 }
7965 \</package>

```

(End definition for \show@min. This function is documented on page ??.)

Chapter 39

Implementation: The hwexam Package

39.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
7966 \*package>
7967 \ProvidesExplPackage{hwexam}{2022/05/24}{3.1.0}{homework assignments and exams}
7968 \RequirePackage{13keys2e}
7969
7970 \newif\iftest\testfalse
7971 \DeclareOption{test}{\testtrue}
7972 \newif\ifmultiple\multiplefalse
7973 \DeclareOption{multiple}{\multipletrue}
7974 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
7975 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7976 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7977 \RequirePackage{keyval}[1997/11/10]
7978 \RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
7979 \newcommand\hwexam@assignment@kw{Assignment}
7980 \newcommand\hwexam@given@kw{Given}
7981 \newcommand\hwexam@due@kw{Due}
7982 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
7983 \newcommand\hwexam@minutes@kw{minutes}
7984 \newcommand\correction@probs@kw{prob.}
7985 \newcommand\correction@pts@kw{total}
7986 \newcommand\correction@reached@kw{reached}
7987 \newcommand\correction@sum@kw{Sum}
7988 \newcommand\correction@grade@kw{grade}
7989 \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

```

7990 \AddToHook{begindocument}{
7991 \ltx@ifpackageloaded{babel}{
7992 \makeatletter
7993 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7994 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7995 \input{hwexam-ngerman.ldf}
7996 }
7997 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7998 \input{hwexam-finnish.ldf}
7999 }
8000 \clist_if_in:NnT \l_tmpa_clist {french}{
8001 \input{hwexam-french.ldf}
8002 }
8003 \clist_if_in:NnT \l_tmpa_clist {russian}{
8004 \input{hwexam-russian.ldf}
8005 }
8006 \makeatother
8007 }{}
8008 }
8009

```

39.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```

8010 \newcounter{assignment}
8011 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8012 \keys_define:nn { hwexam / assignment } {
8013 id .str_set:N = \l_@@_assign_id_str,
8014 number .int_set:N = \l_@@_assign_number_int,
8015 title .tl_set:N = \l_@@_assign_title_tl,
8016 type .tl_set:N = \l_@@_assign_type_tl,
8017 given .tl_set:N = \l_@@_assign_given_tl,
8018 due .tl_set:N = \l_@@_assign_due_tl,
8019 loadmodules .code:n = {
8020 \bool_set_true:N \l_@@_assign_loadmodules_bool
8021 }
8022 }
8023 \cs_new_protected:Nn \_@@_assignment_args:n {
8024 \str_clear:N \l_@@_assign_id_str
8025 \int_set:Nn \l_@@_assign_number_int {-1}
8026 \tl_clear:N \l_@@_assign_title_tl
8027 \tl_clear:N \l_@@_assign_type_tl
8028 \tl_clear:N \l_@@_assign_given_tl
8029 \tl_clear:N \l_@@_assign_due_tl
8030 \bool_set_false:N \l_@@_assign_loadmodules_bool
8031 \keys_set:nn { hwexam / assignment }{ #1 }
8032 }

```

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.

```

8033 \newcommand\given@due[2]{
8034 \bool_lazy_all:nF {
8035 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8036 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8037 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8038 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8039 }{ #1 }
8040
8041 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8042 \tl_if_empty:NF \l_@@_assign_given_tl {
8043 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8044 }
8045 }{
8046 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8047 }
8048
8049 \bool_lazy_or:nnF {
8050 \bool_lazy_and_p:nn {
8051 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8052 }{
8053 \tl_if_empty_p:V \l_@@_assign_due_tl
8054 }
8055 }{
8056 \bool_lazy_and_p:nn {
8057 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8058 }{
8059 \tl_if_empty_p:V \l_@@_assign_due_tl
8060 }
8061 }{ ,~ }
8062
8063 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8064 \tl_if_empty:NF \l_@@_assign_due_tl {
8065 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8066 }
8067 }{
8068 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8069 }
8070
8071 \bool_lazy_all:nF {
8072 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8073 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8074 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8075 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8076 }{ #2 }
8077 }

```

`\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.

```

8078 \newcommand\assignment@title[3]{
8079 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8080 \tl_if_empty:NTF \l_@@_assign_title_tl {
8081 #1
8082 }{
8083 #2\l_@@_assign_title_tl#3
8084 }
8085 }{
8086 #2\l_@@_inclasssign_title_tl#3
8087 }
8088 }

```

(End definition for \assignment@title. This function is documented on page ??.)

\assignment@number Like \assignment@title only for the number, and no around part.

```

8089 \newcommand\assignment@number{
8090 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8091 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8092 \arabic{assignment}
8093 } {
8094 \int_use:N \l_@@_assign_number_int
8095 }
8096 }{
8097 \int_use:N \l_@@_inclasssign_number_int
8098 }
8099 }

```

(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.

```

8100 \newenvironment{assignment}[1][]{
8101 \_@@_assignment_args:n { #1 }
8102 %\sref@target
8103 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8104 \global\stepcounter{assignment}
8105 }{
8106 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8107 }
8108 \setcounter{sproblem}{0}
8109 \renewcommand\prob@label[1]{\assignment@number.##1}
8110 \def\current@section@level{\document@hwexamtype}
8111 %\sref@label{id}{\document@hwexamtype \thesection}
8112 \begin{@assignment}
8113 }{
8114 \end{@assignment}
8115 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

8116 \def\ass@title{
8117 {\protect\document@hwexamtype}\arabic{assignment}
8118 \assignment@title{}\;{}{}\;} -- \given@due{}\}
8119 }
8120 \ifmultiple
8121 \newenvironment{@assignment}{
8122 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8123 \begin{sfragment}[loadmodules]{\ass@title}
8124 }{
8125 \begin{sfragment}{\ass@title}
8126 }
8127 }{
8128 \end{sfragment}
8129 }

```

for the single-page case we make a title block from the same components.

```

8130 \else
8131 \newenvironment{@assignment}{
8132 \begin{center}\bf
8133 \Large@title\strut\
8134 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;{}{}\;{}\\}
8135 \large\given@due{--\;}\;{}{--}
8136 \end{center}
8137 }{}
8138 \fi% multiple

```

39.3 Including Assignments

\in*assignment This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points. Importantly, it resets the `inclassig` keys after the input.

```

8139 \keys_define:nn { hwexam / inclassignment } {
8140 %id .str_set_x:N = \l_@@_assign_id_str,
8141 number .int_set:N = \l_@@_inclassign_number_int,
8142 title .tl_set:N = \l_@@_inclassign_title_tl,
8143 type .tl_set:N = \l_@@_inclassign_type_tl,
8144 given .tl_set:N = \l_@@_inclassign_given_tl,
8145 due .tl_set:N = \l_@@_inclassign_due_tl,
8146 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8147 }
8148 \cs_new_protected:Nn \_@@_inclassignment_args:n {
8149 \int_set:Nn \l_@@_inclassign_number_int {-1}
8150 \tl_clear:N \l_@@_inclassign_title_tl
8151 \tl_clear:N \l_@@_inclassign_type_tl
8152 \tl_clear:N \l_@@_inclassign_given_tl
8153 \tl_clear:N \l_@@_inclassign_due_tl
8154 \str_clear:N \l_@@_inclassign_mhrepos_str
8155 \keys_set:nn { hwexam / inclassignment }{ #1 }
8156 }
8157 \_@@_inclassignment_args:n {}
8158
8159 \newcommand\inputassignment[2][{}]{

```



```

8160 \_@@_inclassignment_args:n { #1 }
8161 \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8162   \input{#2}
8163 }{
8164   \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8165     \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
8166   }
8167 }
8168 \_@@_inclassignment_args:n {}
8169 }
8170 \newcommand\includeassignment[2][]{
8171   \newpage
8172   \inputassignment[#1]{#2}
8173 }

```

(End definition for \in*assignment. This function is documented on page ??.)

39.4 Typesetting Exams

\quizheading

```

8174 \ExplSyntaxOff
8175 \newcommand\quizheading[1]{%
8176   \def\@tas{#1}%
8177   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8178   \ifx\@tas\@empty\else%
8179     \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8180   \fi%
8181 }
8182 \ExplSyntaxOn

```

(End definition for \quizheading. This function is documented on page ??.)

\testheading

```

8183
8184 \def\hwexamheader{\input{hwexam-default.header}}
8185
8186 \def\hwexamminutes{
8187   \tl_if_empty:NTF \testheading@duration {
8188     {\testheading@min}~\hwexam@minutes@kw
8189   }{
8190     \testheading@duration
8191   }
8192 }
8193
8194 \keys_define:nn { hwexam / testheading } {
8195   min .tl_set:N = \testheading@min,
8196   duration .tl_set:N = \testheading@duration,
8197   reqpts .tl_set:N = \testheading@reqpts,
8198   tools .tl_set:N = \testheading@tools
8199 }
8200 \cs_new_protected:Nn \_@@_testheading_args:n {
8201   \tl_clear:N \testheading@min
8202   \tl_clear:N \testheading@duration

```

```

8203 \tl_clear:N \testheading@reqpts
8204 \tl_clear:N \testheading@tools
8205 \keys_set:nn { hwexam / testheading }{ #1 }
8206 }
8207 \newenvironment{testheading}[1][]{
8208 \_@@_testheading_args:n{ #1 }
8209 \newcount\check@time\check@time=\testheading@min
8210 \advance\check@time by -\theassignment@totalmin
8211 \newif\if@bonuspoints
8212 \tl_if_empty:NTF \testheading@reqpts {
8213 \@bonuspointsfalse
8214 }{
8215 \newcount\bonus@pts
8216 \bonus@pts=\theassignment@totalpts
8217 \advance\bonus@pts by -\testheading@reqpts
8218 \edef\bonus@pts{\the\bonus@pts}
8219 \@bonuspointstrue
8220 }
8221 \edef\check@time{\the\check@time}
8222
8223 \makeatletter\hwexamheader\makeatother
8224 }{
8225 \newpage
8226 }

```

(End definition for \testheading. This function is documented on page ??.)

\testspace

```

8227 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

(End definition for \testspace. This function is documented on page ??.)

\testnewpage

```

8228 \newcommand\testnewpage{\iftest\newpage\fi}

```

(End definition for \testnewpage. This function is documented on page ??.)

\testemptypage

```

8229 \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.

```

8230 <@@=problems>
8231 \renewcommand\@problem[3]{
8232 \stepcounter{assignment@probs}
8233 \def\__problemspts{#2}
8234 \ifx\__problemspts\@empty\else
8235 \addtocounter{assignment@totalpts}{#2}
8236 \fi
8237 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8238 \xdef\correction@probs{\correction@probs & #1}%
8239 \xdef\correction@pts{\correction@pts & #2}
8240 \xdef\correction@reached{\correction@reached &}

```

```

8241 }
8242 <@@=hwexam>

```

(End definition for \@problem. This function is documented on page ??.)

`\correction@table` This macro generates the correction table

```

8243 \newcounter{assignment@probs}
8244 \newcounter{assignment@totalpts}
8245 \newcounter{assignment@totalmin}
8246 \def\correction@probs{\correction@probs@kw}
8247 \def\correction@pts{\correction@pts@kw}
8248 \def\correction@reached{\correction@reached@kw}
8249 \stepcounter{assignment@probs}
8250 \newcommand\correction@table{
8251 \resizebox{\textwidth}{!}{%
8252 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8253 &\multicolumn{\theassignment@probs}{c|}|%|
8254 {\footnotesize\correction@forgrading@kw} &\\ \hline
8255 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8256 \correction@pts & \theassignment@totalpts & \\ \hline
8257 \correction@reached & & \[.7cm]\hline
8258 \end{tabular}}
8259 </package>

```

(End definition for \correction@table. This function is documented on page ??.)

39.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```

Chapter 40

References

EdN:13

13

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T_EX distribution. The Comprehensive T_EX Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphicx.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

¹³EdNOTE: we need an un-numbered version sfragment*

- [SIa] *sLaTeX/sTeX-IDE*. URL: <https://github.com/slatex/sTeX-IDE> (visited on 04/22/2022).
- [SIb] *sLaTeX/stexls-vscode-plugin*. URL: <https://github.com/slatex/stexls-vscode-plugin> (visited on 04/22/2022).
- [SLS] *sLaTeX/stexls*. URL: <https://github.com/slatex/stexls> (visited on 04/22/2022).
- [ST] *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: <https://ctan.org/pkg/stex> (visited on 04/22/2022).
- [sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: <https://github.com/sLaTeX/sTeX> (visited on 05/11/2020).
- [Tana] Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: <http://ctan.org/pkg/beamer> (visited on 01/07/2014).
- [Tanb] Till Tantau. *User Guide to the Beamer Class*. URL: <http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.
- [TL] *TeX Live*. URL: <http://www.tug.org/texlive/> (visited on 12/11/2012).