

The sTeX3 Package Collection *

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

2022-09-09

Abstract

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

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

sTeX augments L^AT_EX with

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

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

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

*Version 3.2 (last revised 2022-09-09)

Contents

I	Manual	1
1	What is sTeX?	2
2	Quickstart	3
2.1	Setup	3
2.1.1	Minimal Setup for the PDF-only Workflow	3
2.1.2	GIT-based Setup for the sTeX Development Version	3
2.1.3	sTeX Archives (Manual Setup)	4
2.1.4	The sTeX IDE	4
2.1.5	Manual Setup for Active Documents and Knowledge Management Services	4
2.2	A First sTeX Document	5
2.2.1	OMDoc/xhtml Conversion	8
2.2.2	MMT/OMDoc Conversion	9
3	Creating sTeX Content	10
3.1	How Knowledge is Organized in sTeX	10
3.2	sTeX Archives	11
3.2.1	The Local MathHub-Directory	11
3.2.2	The Structure of sTeX Archives	12
3.2.3	MANIFEST.MF-Files	12
3.2.4	Using Files in sTeX Archives Directly	13
3.3	Module, Symbol and Notation Declarations	14
3.3.1	The smodule-Environment	14
3.3.2	Declaring New Symbols and Notations	16
3.3.3	Operator Notations	20
3.3.3	Argument Modes	20
3.3.3	Mode-b Arguments	20
3.3.3	Mode-a Arguments	21
3.3.3	Mode-B Arguments	22
3.3.4	Type and Definiens Components	23
3.3.5	Precedences and Automated Bracketing	24
3.3.6	Variables	26
3.3.7	Variable Sequences	27
3.4	Module Inheritance and Structures	29
3.4.1	Multilinguality and Translations	29
3.4.2	Simple Inheritance and Namespaces	30
3.4.3	The mathstructure Environment	32
3.4.4	The copymodule Environment	35
3.4.5	The interpretmodule Environment	36
3.5	Primitive Symbols (The sTeX Metatheory)	37
4	Using sTeX Symbols	38
4.1	\symref and its variants	38
4.2	Marking Up Text and On-the-Fly Notations	39

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

9	sTeX-MathHub	74
9.1	Macros and Environments	74
9.1.1	Files, Paths, URIs	74
9.1.2	MathHub Archives	75
9.1.3	Using Content in Archives	76
10	sTeX-References	77
10.1	Macros and Environments	77
10.1.1	Setting Reference Targets	77
10.1.2	Using References	78
11	sTeX-Modules	79
11.1	Macros and Environments	79
11.1.1	The <code>smodule</code> environment	81
12	sTeX-Module Inheritance	83
12.1	Macros and Environments	83
12.1.1	SMS Mode	83
12.1.2	Imports and Inheritance	84
13	sTeX-Symbols	86
13.1	Macros and Environments	86
14	sTeX-Terms	88
14.1	Macros and Environments	88
15	sTeX-Structural Features	90
15.1	Macros and Environments	90
15.1.1	Structures	90
16	sTeX-Statements	91
16.1	Macros and Environments	91
17	sTeX-Proofs: Structural Markup for Proofs	92
18	sTeX-Metatheory	93
18.1	Symbols	93
III	Extensions	94
19	Tikzinput: Treating TIKZ code as images	95
19.1	Macros and Environments	95
20	document-structure: Semantic Markup for Open Mathematical Documents in L^AT_EX	96
21	NotesSlides – Slides and Course Notes	97
22	problem.sty: An Infrastructure for formatting Problems	98

23	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	99
IV	Implementation	100
24	STEX-Basics Implementation	101
24.1	The STEXDocument Class	101
24.2	Preliminaries	102
24.3	Messages and logging	102
24.4	HTML Annotations	103
24.5	Babel Languages	105
24.6	Persistence	107
24.7	Auxiliary Methods	107
25	STEX-MathHub Implementation	111
25.1	Generic Path Handling	111
25.2	PWD and kpsewhich	113
25.3	File Hooks and Tracking	114
25.4	MathHub Repositories	115
25.5	Using Content in Archives	120
26	STEX-References Implementation	125
26.1	Document URIs and URLs	125
26.2	Setting Reference Targets	127
26.3	Using References	130
27	STEX-Modules Implementation	137
27.1	The smodule environment	141
27.2	Invoking modules	147
28	STEX-Module Inheritance Implementation	149
28.1	SMS Mode	149
28.2	Inheritance	153
29	STEX-Symbols Implementation	159
29.1	Symbol Declarations	159
29.2	Notations	167
29.3	Variables	177
30	STEX-Terms Implementation	186
30.1	Symbol Invocations	186
30.2	Terms	194
30.3	Notation Components	199
30.4	Variables	201
30.5	Sequences	204
31	STEX-Structural Features Implementation	205
31.1	Imports with modification	206
31.2	The feature environment	214
31.3	Structure	214

32	sTeX-Statements Implementation	225
32.1	Definitions	225
32.2	Assertions	231
32.3	Examples	234
32.4	Logical Paragraphs	237
33	The Implementation	242
33.1	Proofs	242
34	sTeX-Others Implementation	251
35	sTeX-Metatheory Implementation	253
36	Tikzinput Implementation	256
37	document-structure.sty Implementation	259
37.1	Package Options	259
37.2	Document Structure	260
37.3	Front and Backmatter	264
37.4	Global Variables	266
38	NotesSlides – Implementation	267
38.1	Class and Package Options	267
38.2	Notes and Slides	269
38.3	Header and Footer Lines	273
38.4	Frame Images	275
38.5	Sectioning	276
38.6	Excursions	279
39	The Implementation	281
39.1	Package Options	281
39.2	Problems and Solutions	282
39.3	Markup for Added Value Services	289
39.4	Multiple Choice Blocks	289
39.5	Filling in Concrete Solutions	290
39.6	Including Problems	291
39.7	Reporting Metadata	292
39.8	Testing and Spacing	293
40	Implementation: The hwexam Package	295
40.1	Package Options	295
40.2	Assignments	296
40.3	Including Assignments	299
40.4	Typesetting Exams	300
40.5	Leftovers	302
41	References	303

Part I

Manual



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



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

Chapter 1

What is sTeX?

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

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

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

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

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

- The sTeX package collection to use semantic annotations in L^AT_EX documents,
- RuS_{TE}X [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_{TE}X system already.

Chapter 2

Quickstart

2.1 Setup

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

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

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

2.1.1 Minimal Setup for the PDF-only Workflow

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

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

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

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

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

and keep it updated by pulling updates via `git pull` in the cloned $\text{\texttt{sTeX}}$ directory. Make sure to either clone the $\text{\texttt{sTeX}}$ repository into a local texmf-tree or to 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 (env.) \begin{smodule}{GeometricSeries}
...
\end{smodule}
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

\[\defeq{\geometricSeries}{\definien{
  \infinisum{\svar{n}}{1}{
    \realdivide[frac]{1}{
      \realpower{2}{\svar{n}}
    }
  }}
}\].\]

```

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

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

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

2.2.1 OMDoc/xhtml Conversion

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

TODO VSCode Plugin

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

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

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

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

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

```

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

```

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

Remark 2.2.2:

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

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

2.2.2 Mmt/OMDoc Conversion

Another way to convert our document to *actual* MMT/OMDOC is to put it in an `TEX` archive (see [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].

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

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

3.2.1 The Local MathHub-Directory

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

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

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

3.2.2 The Structure of \TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

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

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

3.2.3 MANIFEST.MF-Files

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

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

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

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

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

source-base or

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

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

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

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

3.2.4 Using Files in \TeX Archives Directly

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

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

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

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

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

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

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

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

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

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

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

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

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

Remark 3.2.1:

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

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

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

Example 1

Input:

```

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

```

Output:

Hello World

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

For example:

Example 2

Input:

```

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

```

Output:

```

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

```

3.3.2 Declaring New Symbols and Notations

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

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

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

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

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

Example 3

Input:

```

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

```

Output:

```

Given a foo, we can...

```

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

Example 4

Input:

```

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

```

Output:

```

this is a symbol taking two arguments.

```

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

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

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

Example 5

Input:

```

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

```

Output:

```

First: a; Second: b

```

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

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

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

Example 6

Input:

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

Output:

First: a ; Second: b



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



Note that it is required that

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

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

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

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

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

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

Example 7

Input:

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

Output:

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

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

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

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

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

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

Operator Notations

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

We can do so by supplying the `\notation` (or `\symdef`) with an *operator notation*, indicated with the optional argument `op=`. We can then invoke the operator notation using `\symbolname![notation-identifier]`. Since operator notations never take arguments, we do not need to use `\comp` in it, the whole notation is wrapped in a `\comp` automatically:

Example 8

Input:

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

Output:

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

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

3.3.3 Argument Modes

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

Mode-b Arguments

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

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

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

Example 9

Input:

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

Output:

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

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

Mode-a Arguments

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

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

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

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

Example 10

Input:

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

Output:

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

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

Example 11

Input:

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

Output:

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

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

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

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

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

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

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

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

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

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

Mode-B Arguments

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

Example 12

Input:

```

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

```

Output:

$\forall x,y,z.P$

3.3.4 Type and Definiens Components

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

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

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

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

Example 13

Input:

```

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

```

Output:

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

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

Example 14

Input:

```

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

```

Output:

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

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

```

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

```

Output:

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

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

Example 16

Input:

```

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

```

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

We will investigate the precise meaning of `<opprec>` and the `<argprec>`s shortly – in the vast majority of cases, it is perfectly sufficient to think of `prec=` taking a single number and having that be *the* precedence of the notation, where lower precedences (somewhat counterintuitively) bind stronger than higher precedences. So fixing our notations for `\addition` and `\multiplication`, we get:

Example 18

Input:

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

Output:

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

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

`\infprec`
`\neginfprec`

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

More precisely, each notation takes

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

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

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

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

In the example above:



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

3.3.6 Variables

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

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

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

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

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

Example 19

Input:

```

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

```

Output:

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

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

TODO: bind=forall/exists

3.3.7 Variable Sequences

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

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

This is best shown by example:

Example 20

Input:

```

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

```

Output:

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

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$a_1 + \dots + a_n$

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

Example 22

Input:

```

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

```

Output:

a_1^1, \dots, a_n^m and $a_1^1 + \dots + a_n^m$

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

Example 23

Input:

```

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

```

Output:

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

3.4 Module Inheritance and Structures

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

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

3.4.1 Multilinguality and Translations

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

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

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

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

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

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

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

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

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

3.4.2 Simple Inheritance and Namespaces

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

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

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



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

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

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

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



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

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



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

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

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

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

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



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



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

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

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

3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

```

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

Output:

A *monoid* is...

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

Example 25

Input:

```

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

Output:

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

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

Example 26

Input:

```

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

Output:

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

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

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

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

- `mathstructure{<name>}` simply creates a nested theory with name $\hookrightarrow M \rightarrow$ `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)`
- $\hookrightarrow M \rightarrow$ – a *dependent record type with manifest fields*, the fields of which are generated
- $\hookrightarrow T \rightarrow$ 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.

usestructure (*env.*) The `usestructure{<struct>}` environment is used in multilingual settings as a parallel to the `mathstructure`. It opens a group and then issues a `\usemodule{.../<struct>-structure}` that gives the body access to all the semantic macros in the referenced structure.

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     \renamedekl[name=universe]{universe}{runiverse}
4     \renamedekl[name=plus]{operation}{rplus}
5     \renamedekl[name=zero]{unit}{rzero}
6     \renamedekl[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    \renamedekl[name=times]{operation}{rtimes}
14    \renamedekl[name=one]{unit}{rone}
15  \end{copymodule}
16  \notation*{rtimes}[cdot,op=\cdot,prec=50]{#1 \comp\cdot #2}
17  \notation*{rone}[one]{\comp1}
18  Test: $\rtimes a\{rplus c\{rtimes d\}}$
19 \end{smodule}

```

Output:

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

TODO: explain donotclone

3.4.5 The interpretmodule Environment

TODO: explain

Example 31

Input:

```

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

```

Output:

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

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

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

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

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

Chapter 4

Using \TeX Symbols

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

4.1 `\symref` and its variants

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

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

Example 32

Input:

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

Output:

A natural number is...

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

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

Example 33

Input:

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

Output:

Natural numbers are...



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

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

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

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

4.2 Marking Up Text and On-the-Fly Notations

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

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

Example 34

Input:

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

Output:

The sum of n and m is...

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

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



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

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

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

Example 35

Input:

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

Output:

```
Addition is...
```

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

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

Example 36

Input:

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

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

Output:

adding k yields...

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

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

Example 37

Input:

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

Output:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

For a further example, the following:

Part III

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

```
\sref[file=../stex-doc]{part:extends}[in=../stex-doc,title={the full \stex}3 document]
```

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

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

Chapter 5

sTeX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

The *presentation* of these environments can be customized to use e.g. predefined `theorem-environments`, see [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 [chapter 6](#)), `type=` for customization (see [section 5.3](#)) and additional information (e.g. definition principles, “difficulty” etc), as well as `title=` (for giving the paragraph a title), and finally `for=`.

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

Example 38

Input:

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

Output:

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

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

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

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

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

Example 39

Input:

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

```

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

```

Output:

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

Axiom 5.1.2 (Associativity). \circ is associative

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

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

EdN:4

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

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

```

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

```

⁴EdNOTE: MK: reference

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

5.2 Proofs

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

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{spfblock}\end{subproof}
37 \conclude{
38   We have considered all the cases, so we have proven the assertion.
39 }
40 \end{spproof}

```

This yields the following result:

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

For the induction we have to consider three cases:

1. $n = 1$

then we compute $1 = 1^2$

2. $n = 2$

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

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

3. $n > 1$

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

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

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

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

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

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

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

\yield See above

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

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

\justarg

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

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

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

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

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

5.3 Highlighting and Presentation Customizations

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

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

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

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

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

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

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

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

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

```

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

```

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

```

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

```

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

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

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

Chapter 6

Cross References

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

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

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

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

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

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

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

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

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

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

For example, the reference to the `sfragment`-environment above will appear as “subsection 7.2.1 (Introduction) in the `gTeX3` manual” if you are reading this in the package documentation for `stex-references` directly, but as a linked “subsection 7.2.1” in the full documentation or manual. This is achieved using

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

For a further example, the following:

Part III

will say “Part III” (and link accordingly) in the full documentation, and “Part III (Extensions) in the full `gTeX3` documentation” everywhere else. This is achieved using

```
\sref[file=./stex-doc]{part:extends}[in=./stex-doc,title={the full \sTeX}3 document]
```

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

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

Chapter 7

Additional Packages

7.1 Tikzinput: Treating TIKZ code as images

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

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

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

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

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

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

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

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

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

7.2 Modular Document Structuring

7.2.1 Introduction

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

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

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

7.2.2 Package Options

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

<code>class=<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>

7.2.3 Document Fragments

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

```

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

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

7.2.4 Ending Documents Prematurely

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

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

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

7.2.5 Global Document Variables

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

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

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

7.3 Slides and Course Notes

7.3.1 Introduction

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

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

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

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

7.3.2 Package Options

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

<code>slides</code>	The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see subsection 7.3.3).
<code>notes</code>	
<code>sectocframes</code>	If the option <code>sectocframes</code> is given, then for the <code>sfragments</code> , special frames with the <code>sfragment</code> title (and number) are generated.
<code>frameimages</code>	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.
<code>fiboxed</code>	

7.3.3 Notes and Slides

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



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

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

```

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

```

```

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

```

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



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



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

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

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

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

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

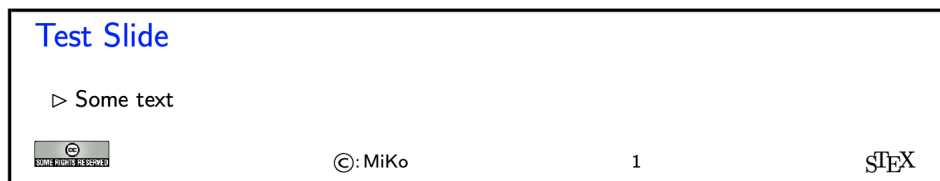
`nexample` (*env.*) level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,

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

`nassertion` (*env.*)

7.3.4 Customizing Header and Footer Lines

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



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

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

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

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

7.3.5 Frame Images

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

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

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

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


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

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

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

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

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

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

7.3.6 Excursions

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

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

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

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

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

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

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

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

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



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

7.4 Representing Problems and Solutions

7.4.1 Introduction

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

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

7.4.2 Problems and Solutions

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

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

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

Example 40

Input:

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

```

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

```

Output:

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

Hint: Think positively, this is simple!

Note: Justify your answer

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

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

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

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

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

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

7.4.3 Markup for Added-Value Services

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

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

Multiple Choice Blocks

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

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

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

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

Example 41

Input:

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

Output:

Problem 7.4.2 (Functions)

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

☐ def

Correct!

☐ function

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

☐ fun

Wrong! *that is for Standard ML*

☐ public static void

Wrong! *that is for Java*

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

Example 42

Input:

```

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

```

Output:

Problem 7.4.3 (Functions)

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

☐ def

☐ function

☐ fun

☐ public static void

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

Filling-In Concrete Solutions

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

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

Example 43

Input:

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

Output:

Problem 7.4.4 (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle?
and the actual solution in solutions mode:

Example 44

Input:

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

Output:

Problem 7.4.5 (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle? !

If we do not want to leak information about the solution by the size of the blank we can also give `\fillinsol` an optional argument with a size: `\fillinsol[3cm]{12}` makes a box three cm wide.

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

7.4.4 Including Problems

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

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

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

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

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

7.4.5 Testing and Spacing

The `problem` package is often used by the `hwexam` package, which is used to create homework assignments and exams. Both of these have a “test mode” (invoked by the package option `test`), where certain information –master solutions or feedback – is not shown in the presentation.

`\testspace` `\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. Specific instances exist: `\testsmallspace`, `\testsmallspace`, `\testsmallspace` `\testsmallspace` give small (1cm), medium (2cm), and big (3cm) vertical space.
`\testsmallspace` `\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.
`\testnewpage`
`\testemptypage`

7.5 Homeworks, Quizzes and Exams

7.5.1 Introduction

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

7.5.2 Package Options

`solutions` The `hwexam` package and class take the options `solutions`, `notes`, `hints`, `gnotes`, `pts`, `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation for a description of the intended behavior).
`notes`
`hints`
`gnotes`
`pts`
`min`

`multiple` Furthermore, the `hwexam` package takes the option `multiple` 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.).

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the \LaTeX source.

7.5.3 Assignments

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

7.5.4 Including Assignments

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

7.5.5 Typesetting Exams

testheading (*env.*) 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.

```
reqpts 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)

2022-09-09

You have one hour (sharp) for the test;

Write the solutions to the sheet.

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

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

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

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

	To be used for grading, do not write here													
prob.	7.4.1	7.4.2	7.4.3	7.4.4	7.4.5	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total	10					4	4	6	6	4	4	2	40	
reached														

good luck

EdN:8

8

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

Part II

Documentation

Chapter 8

STEX-Basics

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

8.1 Macros and Environments

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

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

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

8.1.1 HTML Annotations

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

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

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

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

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

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

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 Babel Languages

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

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

8.1.3 Auxiliary Methods

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

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

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

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

Chapter 9

ST_EX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

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

9.1.2 MathHub Archives

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

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

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

<code>\l_stex_current_repository_prop</code>

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

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

<code>\stex_set_current_repository:n</code>

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

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

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

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

9.1.3 Using Content in Archives

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

Chapter 10

STEX-References

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

10.1 Macros and Environments

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

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

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

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

10.1.1 Setting Reference Targets

<code>\stex_ref_new_doc_target:n</code>	<code>\stex_ref_new_doc_target:n{<id>}</code> Sets a new reference target with id <code><id></code> .
-----------------------------------------	----------------------------------------------------------------------------------------------------------------------

<code>\stex_ref_new_sym_target:n</code>	<code>\stex_ref_new_sym_target:n{<uri>}</code> Sets a new reference target for the symbol <code><uri></code> .
-----------------------------------------	-------------------------------------------------------------------------------------------------------------------------------

10.1.2 Using References

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

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

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

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

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

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

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

Chapter 11

sTeX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

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

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

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

<hr/> <u>$\backslash\text{c_stex_module_}\langle URI \rangle_constants$</u> <hr/>	The names of all constants declared in the module
--------------------------------------------------------------------------------------------------	---------------------------------------------------

<hr/> <u>$\backslash\text{c_stex_module_}\langle URI \rangle_constants$</u> <hr/>	The full URIs of all modules imported in this module
--------------------------------------------------------------------------------------------------	------------------------------------------------------

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

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

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

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

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

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

11.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Chapter 12

STEX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

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

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

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

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

12.1.2 Imports and Inheritance

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

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

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

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

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

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 $\langle name \rangle . \langle lang \rangle . \text{tex}$ 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 $\langle name \rangle . \langle lang \rangle . \text{tex}$ 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.

<code>\l_stex_import_name_str</code>	stores the result in these four variables.
<code>\l_stex_import_archive_str</code>	
<code>\l_stex_import_path_str</code>	
<code>\l_stex_import_ns_str</code>	

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

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 13

$\text{\texttt{S}\text{\textsf{T}\text{\textsf{E}\text{\textsf{X}}}}$ -Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

$\text{\texttt{\textbackslash symdecl}}$	$\text{\texttt{\textbackslash symdecl}}\{\langle\textit{macroname}\rangle\}[\langle\textit{args}\rangle]$
------------------------------------------	-----------------------------------------------------------------------------------------------------------

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

- **name**: An (OMDOC) name. By default equal to $\langle\textit{macroname}\rangle$.
- **type**: An (ideally semantic) term, representing a *type*. Not used by $\text{\texttt{S}\text{\textsf{T}\text{\textsf{E}\text{\textsf{X}}}}$, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by $\text{\texttt{S}\text{\textsf{T}\text{\textsf{E}\text{\textsf{X}}}}$, but passed on to MMT for semantic services.
- **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. $\text{\texttt{\textbackslash symdecl}}\{\texttt{plus}}\}[\texttt{args=ii}]$ allows for $\text{\texttt{\textbackslash plus}}\{2\}\{2\}$.
 - a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. $\text{\texttt{\textbackslash symdecl}}\{\texttt{plus}}\}[\texttt{args=a}]$ allows for $\text{\texttt{\textbackslash plus}}\{2,2,2\}$.
 - b** a *variable* argument. Is treated by $\text{\texttt{S}\text{\textsf{T}\text{\textsf{E}\text{\textsf{X}}}}$ like an **i**-argument, but an application is turned into an $\text{\texttt{OMBind}}$ in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g. $\text{\texttt{\textbackslash symdecl}}\{\texttt{forall}}\}[\texttt{args=bi}]$ allows for $\text{\texttt{\textbackslash forall}}\{x\}\text{\texttt{\textbackslash in}}\{\texttt{Nat}}\}\{x\geq 0\}$.

<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>type</code> (token list), • <code>args</code> (string of is, as and bs), • <code>arity</code> (integer string), • <code>assocs</code> (integer string; number of associative arguments),
<hr/> <hr/> <code>\stex_all_symbols:n</code>	<p>Iterates over all currently available symbols. Requires two <code>\seq_map_break:</code> to break fully.</p>
<hr/> <hr/> <code>\stex_get_symbol:n</code>	<p>Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...</p>
<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> (sequence of integer strings)
<hr/> <hr/> <code>\symdef</code>	<p><code>\symdef[<args>]{<symbol>}{<notations+>}</code> Combines <code>\symdecl</code> and <code>\notation</code> by introducing a new symbol and assigning a new notation for it.</p>

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

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

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

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

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

<code>\STEXInternalTermMathOMSiiii</code>	<code><URI><fragment><precedence><body></code>
<code>\STEXInternalTermMathOMAi</code>	
<code>\STEXInternalTermMathOMBiiii</code>	

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

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

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

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

Chapter 15

TeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` (*env.*) TODO

Chapter 16

TeX-Statements

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

16.1 Macros and Environments

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

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

Chapter 17

sTEX-Proofs: Structural Markup for Proofs

Chapter 18

sT_EX-Metatheory

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput: Treating TIKZ code as images

19.1 Macros and Environments

Chapter 20

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

Chapter 21

NotesSlides – Slides and Course Notes

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

Chapter 23

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

Part IV
Implementation

Chapter 24

sTeX -Basics Implementation

24.1 The sTeXDocument Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/08/08}{3.2.0}{sTeX document class}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \bool_set_true:N \c_stex_document_class_bool
12
13 \RequirePackage{stex}
14
15 \stex_html_backend:TF {
16   \LoadClass{article}
17 }{
18   \LoadClass[border=1px,varwidth,crop=false]{standalone}
19   \setlength\textwidth{15cm}
20 }
21 \RequirePackage{standalone}
22
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```

```

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

```

24.2 Preliminaries

```

44 <*package>
45
46 %%%%%%%%% basics.dtx %%%%%%%%%
47
48 \RequirePackage{expl3,l3keys2e,ltxcmds}
49 \ProvidesExplPackage{stex}{2022/08/08}{3.2.0}{sTeX package}
50
51 \bool_if_exist:NF \c_stex_document_class_bool {
52   \bool_set_false:N \c_stex_document_class_bool
53   \RequirePackage{standalone}
54 }
55
56 \message{^^J*~This~is~sTeX~version~3.2.0~*^^J}
57
58 %\RequirePackage{morewrites}
59 %\RequirePackage{amsmath}
60
61 Package options:
62 \keys_define:nn { stex } {
63   debug      .clist_set:N = \c_stex_debug_clist ,
64   lang       .clist_set:N = \c_stex_languages_clist ,
65   mathhub    .tl_set_x:N  = \mathhub ,
66   usesms     .bool_set:N  = \c_stex_persist_mode_bool ,
67   writesms   .bool_set:N  = \c_stex_persist_write_mode_bool ,
68   image      .bool_set:N  = \c_tikzinput_image_bool ,
69   unknown    .code:n      = {}
70 }
71 \ProcessKeysOptions { stex }

```

\stex The sTeX logo:

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

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

24.3 Messages and logging

```

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

```

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

```

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

```

(End definition for \stex_debug:nn. This function is documented on page 72.)

Redirecting messages:

```

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

```

24.4 HTML Annotations

```

107 <@@=stex_annotate>

```

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

```

108 \tl_new:N \l_stex_html_arg_tl

```

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

`_stex_html_checkempty:n`

```

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

```

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

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

`\stex_if_do_html:TF`

```

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

```

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

`\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 72.)

`\stex_annotate_html:nnn`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

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

```

```

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

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 73.)

24.5 Babel Languages

```

160 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

```

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

```


24.6 Persistence

```

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

```

24.7 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\ignorespacesandpars`

```

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

```

```

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

```

```

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

```

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

\MMTrule

```

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

```

```

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

```

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

Chapter 25

STEX -MathHub Implementation

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

25.1 Generic Path Handling

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

`\stex_path_from_string:Nn`

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

```

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

```

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

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

```

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

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 74.)

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

```

438 \str_const:Nn \c__stex_path_dot_str {.}
439 \str_const:Nn \c__stex_path_up_str {..}

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

(End definition for `\c_stex_pwd_seq` and `\c_stex_pwd_str`. These variables are documented on page 74.)

25.3 File Hooks and Tracking

516 `<@@=stex_files>`

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

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

`\g_stex_files_stack` keeps track of file changes

517 `\seq_gclear_new:N\g_stex_files_stack`

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

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

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 74.)

`\g_stex_currentfile_seq`

521 `\seq_gclear_new:N\g_stex_currentfile_seq`

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 75.)

`\stex_filestack_push:n`

```

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

```



```

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

```

(End definition for `\stex_filestack_push:n`. This function is documented on page 75.)

`\stex_filestack_pop:`

```

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

```

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

Hooks for the current file:

```

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

```

25.4 MathHub Repositories

```

555 <@=stex_mathhub>

```

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

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

```

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

```

```

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

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 75.)

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

```

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

```

```

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

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

621 \seq_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for \l_stex_mathhub_manifest_file_seq.)

__stex_mathhub_find_manifest:N

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

```

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

```

(End definition for __stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior

File variable used for MANIFEST-files

```

657 \ior_new:N \c__stex_mathhub_manifest_ior

```

(End definition for `\c__stex_mathhub_manifest_ior.`)

`__stex_mathhub_parse_manifest:n` Stores the entries in manifest file in the corresponding property list:

```

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

```

(End definition for `__stex_mathhub_parse_manifest:n.`)

`\stex_set_current_repository:n`

```

704 \cs_new_protected:Nn \stex_set_current_repository:n {

```

```

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

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 75.)

`\stex_require_repository:n`

```

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

```

(End definition for `\stex_require_repository:n`. This function is documented on page 75.)

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 75.)

`\stex_in_repository:nn` Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```

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

```

```

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

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 75.)

25.5 Using Content in Archives

`\mhpath`

```

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

```

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

`\inputref`

`\mhinput`

```

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

```

```

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

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 76.)

`\addmhbibresource`

```

838 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {

```

```

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

```

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

\libinput

```

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

```

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

\libusepackage

```

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

```



```

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

```

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

`\mhgraphics`
`\cmhgraphics`

```

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

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 76.)

`\lstinputmhlisting`
`\clstinputmhlisting`

```

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

```

933

934 `\end{package}`

(End definition for `\lstinputmhlisting` and `\clstinputmhlisting`. These functions are documented on page 76.)

Chapter 26

STEX -References Implementation

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

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

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
956 \str_new:N \l_stex_current_docns_str
```

(End definition for `\l_stex_current_docns_str`. This variable is documented on page 77.)

`\stex_get_document_uri:`

```
957 \cs_new_protected:Nn \stex_get_document_uri: {
```

```

958 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
959 \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
960 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
961 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
962 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
963
964 \str_clear:N \l_tmpa_str
965 \prop_if_exist:NT \l_stex_current_repository_prop {
966   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
967     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
968   }
969 }
970
971 \str_if_empty:NTF \l_tmpa_str {
972   \str_set:Nx \l_stex_current_docns_str {
973     file:/\stex_path_to_string:N \l_tmpa_seq
974   }
975 }{
976   \bool_set_true:N \l_tmpa_bool
977   \bool_while_do:Nn \l_tmpa_bool {
978     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
979     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
980       {source} { \bool_set_false:N \l_tmpa_bool }
981     }{}{
982       \seq_if_empty:NT \l_tmpa_seq {
983         \bool_set_false:N \l_tmpa_bool
984       }
985     }
986   }
987
988   \seq_if_empty:NTF \l_tmpa_seq {
989     \str_gset_eq:NN \l_stex_current_docns_str \l_tmpa_str
990   }{
991     \str_gset:Nx \l_stex_current_docns_str {
992       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
993     }
994   }
995 }
996 %\stex_get_document_url:
997 }

```

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

`\l_stex_current_docurl_str`

```

998 \str_new:N \l_stex_current_docurl_str

```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 77.)

`\stex_get_document_url:`

```

999 \cs_new_protected:Nn \stex_get_document_url: {
1000   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1001   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1002   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1003   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1004   \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

1005
1006 \str_clear:N \l_tmpa_str
1007 \prop_if_exist:NT \l_stex_current_repository_prop {
1008   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
1009     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
1010       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
1011     }
1012   }
1013 }
1014
1015 \str_if_empty:NTF \l_tmpa_str {
1016   \str_set:Nx \l_stex_current_docurl_str {
1017     file:/\stex_path_to_string:N \l_tmpa_seq
1018   }
1019 }{
1020   \bool_set_true:N \l_tmpa_bool
1021   \bool_while_do:Nn \l_tmpa_bool {
1022     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1023     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1024       {source} { \bool_set_false:N \l_tmpa_bool }
1025     }{}{
1026       \seq_if_empty:NT \l_tmpa_seq {
1027         \bool_set_false:N \l_tmpa_bool
1028       }
1029     }
1030   }
1031
1032   \seq_if_empty:NTF \l_tmpa_seq {
1033     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1034   }{
1035     \str_set:Nx \l_stex_current_docurl_str {
1036       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1037     }
1038   }
1039 }
1040 }

```

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

26.2 Setting Reference Targets

```

1041 \str_const:Nn \c__stex_refs_url_str{URL}
1042 \str_const:Nn \c__stex_refs_ref_str{REF}
1043 \str_new:N \l__stex_refs_curr_label_str
1044 % @currentlabel -> number
1045 % @currentlabelname -> title
1046 % @currentHref -> name.number <- id of some kind
1047 % @currentcounter <- name/id
1048 % \#autorefname <- "Section"
1049 % \theH# -> \arabic{section}
1050 % \the# -> number
1051 % \hyper@makecurrent{#}
1052 \int_new:N \l__stex_refs_unnamed_counter_int

```

Restoring references from .sref-files

\STEXInternalSrefRestoreTarget

1053 \cs_new_protected:Npn \STEXInternalSrefRestoreTarget #1#2#3#4#5 {}

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

\stex_ref_new_doc_target:n

```

1054 \seq_new:N \g_stex_ref_files_seq
1055
1056 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1057   %\stex_get_document_uri:
1058   \str_clear:N \l__stex_refs_curr_label_str
1059   \str_set:Nx \l_tmpa_str { #1 }
1060   \str_if_empty:NT \l_tmpa_str {
1061     \int_gincr:N \l__stex_refs_unnamed_counter_int
1062     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1063   }
1064   \str_set:Nx \l__stex_refs_curr_label_str {
1065     \l_stex_current_docns_str?\l_tmpa_str
1066   }
1067
1068   \exp_args:Noo \STEXInternalAuxAddDocRef\l_stex_current_docns_str\l_tmpa_str
1069
1070   %\seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1071   %   \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1072   %}
1073   %\seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1074   %   \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1075   %}
1076
1077
1078   \stex_if_smsmode:TF {
1079     %\stex_get_document_url:
1080     %\str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1081     %\str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1082   }{
1083     \iow_now:Nx \c__stex_refs_refs_iow {
1084       \STEXInternalSrefRestoreTarget
1085       {\l_stex_current_docns_str}
1086       {\l_tmpa_str}
1087       {\@currentcounter}
1088       {\@currentlabel}
1089       {\tl_if_exist:NT\@currentlabelname{\exp_args:No\unexpanded\@currentlabelname}}
1090     }
1091     %\iow_now:Nx \c__stex_refs_refs_iow {
1092     %   {\l_stex_current_docns_str?\l_tmpa_str}~=={\use:c{\@currentcounter autorefname}~\@currentcounter}
1093     %   \stex_debug:nn{sref}{New~label~\l__stex_refs_curr_label_str~at~\use:c{\use:c{\@currentcounter}~\@currentcounter}}
1094     %   \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1095     %   \immediate\write\auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_str}}
1096     %   \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1097   }
1098 }
1099 \NewDocumentCommand \slabel {m} {\stex_ref_new_doc_target:n {#1}}

```

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

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

```

1100 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1101   \exp_args:NNx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1102     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1103       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1104     }
1105   }{
1106     \exp_args:NNx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1107     %\seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1108       \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1109     %}
1110     \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1111   }
1112
1113   %\str_set:Nn \l_tmpa_str {#1?#2}
1114   %\str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1115   %\seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1116     % \seq_new:c {g__stex_refs_labels_#2_seq}
1117     %}
1118   %\seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1119     % \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1120     %}
1121 }

```

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

```

1122 \AtEndDocument{
1123   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1124 }

```

`\stex_ref_new_sym_target:n`

```

1125 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1126
1127   % \stex_if_smsmode:TF {
1128   %   \str_if_exist:cF{sref_sym_#1_type}{
1129   %     \stex_get_document_url:
1130   %     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
1131   %     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1132   %   }
1133   % }{
1134   %   \str_if_empty:NF \l__stex_refs_curr_label_str {
1135   %     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1136   %     \immediate\write\@auxout{
1137   %       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label
1138   %       \l__stex_refs_curr_label_str
1139   %     }
1140   %   }
1141   % }
1142 % }
1143 }

```

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

26.3 Using References

`\sref` Optional arguments:

```

1144
1145 \keys_define:nn { stex / sref / 1 } {
1146   archive .str_set_x:N = \l__stex_refs_repo_str,
1147   file     .str_set_x:N = \l__stex_refs_file_str,
1148   % TODO get rid of this
1149   fallback .code:n = {},
1150   pre      .code:n = {},
1151   post     .code:n = {}
1152 }
1153 \cs_new_protected:Nn \__stex_refs_args_i:n {
1154   \str_clear:N \l__stex_refs_repo_str
1155   \str_clear:N \l__stex_refs_file_str
1156   \keys_set:nn { stex / sref / 1 } { #1 }
1157 }
1158 \keys_define:nn { stex / sref / 2 } {
1159   in .str_set_x:N = \l__stex_refs_in_str,
1160   archive .str_set_x:N = \l__stex_refs_repob_str,
1161   title .tl_set:N = \l__stex_refs_title_tl
1162 }
1163 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1164   \str_clear:N \l__stex_refs_in_str
1165   \tl_clear:N \l__stex_refs_title_tl
1166   \str_clear:N \l__stex_refs_repob_str
1167   \keys_set:nn { stex / sref / 2 } { #1 }
1168 }

```

The actual macro:

```

1169 \NewDocumentCommand \sref { 0{} m 0{} }{
1170   \__stex_refs_args_i:n{#1}
1171   \__stex_refs_args_ii:n{#3}
1172   \str_clear:N \l__stex_refs_uri_str
1173   \__stex_refs_find_uri:n{#2}
1174   \__stex_refs_do_sref:n{#2}
1175 }
1176 \NewDocumentCommand \extref { 0{} m m }{
1177   \__stex_refs_args_i:n{#1}
1178   \__stex_refs_args_ii:n{#3}
1179   \str_if_empty:NT \l__stex_refs_in_str {
1180     \msg_error:nn{stex}{error/extrefmissing}
1181   }
1182   \str_clear:N \l__stex_refs_uri_str
1183   \__stex_refs_find_uri:n{#2}
1184   \__stex_refs_do_sref_in:n{#2}
1185 }
1186
1187 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1188   \stex_debug:nn{sref}{File:~\l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1189   \str_if_empty:NTF \l__stex_refs_file_str {
1190     \stex_debug:nn{sref}{Empty.~Checking~current~file~for~#1}
1191     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str_seq}{
1192       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str_seq}{

```



```

1193     \str_if_eq:nnT{#1}{##1}{
1194       \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1195       \stex_debug:nn{sref}{Found.}
1196       \seq_map_break:
1197     }
1198   }
1199 }
1200 \str_if_empty:NT \l__stex_refs_uri_str {
1201   \stex_debug:nn{sref}{Checking~other~files}
1202   \seq_map_inline:Nn \g_stex_ref_files_seq {
1203     \stex_debug:nn{sref}{##1...}
1204     \seq_map_inline:cn{g_stex_ref_##1_seq}{
1205       \str_if_eq:nnT{#1}{####1}{
1206         \stex_debug:nn{sref}{Found~##1}
1207         \str_set:Nn \l__stex_refs_uri_str {##1}
1208         \seq_map_break:n{\seq_map_break:}
1209       }
1210     }
1211   }
1212 }
1213 }{
1214   \str_if_empty:NTF \l__stex_refs_repo_str {
1215     \prop_if_exist:NTF \l_stex_current_repository_prop {
1216       \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1217       \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1218       \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1219       \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1220     }{
1221       \stex_path_from_string:Nn \l_tmpb_seq {
1222         \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1223       }
1224       \str_set:Nx \l__stex_refs_uri_str {file:/\stex_path_to_string:N \l_tmpb_seq}
1225     }
1226   }{
1227     \stex_require_repository:n \l__stex_refs_repo_str
1228     \prop_get:cnN { c_stex_mathhub_\l__stex_refs_repo_str_manifest_prop } { ns } \l__stex_refs_uri_str
1229     \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1230     \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1231     \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1232   }
1233 }
1234 }
1235
1236 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1237   \cs_if_exist:cTF{autoref}{
1238     \exp_args:Nx\autoref{sref_#1}
1239   }{
1240     \exp_args:Nx\ref{sref_#1}
1241   }
1242 }
1243
1244 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1245   \str_if_empty:NTF \l__stex_refs_uri_str {
1246     \str_if_empty:NTF \l__stex_refs_in_str {

```

```

1247     \stex_debug:nn{sref}{autoref-on~#1}
1248     \__stex_refs_do_autoref:n{#1}
1249   }{
1250     \stex_debug:nn{sref}{srefin-on~#1}
1251     \__stex_refs_do_sref_in:n{#1}
1252   }
1253 }{
1254   \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1255     \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref\_l__stex_refs_uri_str_seq}{\detokenize{#1}}{
1256       \stex_debug:nn{sref}{Reference~found~in~ref~files;~autoref-on~#1}
1257       \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1258     }{
1259       \str_if_empty:NTF \l__stex_refs_in_str {
1260         \stex_debug:nn{sref}{in~empty;~autoref-on~#1}
1261         \__stex_refs_do_autoref:n{#1}
1262       }{
1263         \stex_debug:nn{sref}{in~non-empty;~srefin-on~#1}
1264         \__stex_refs_do_sref_in:n{#1}
1265       }
1266     }
1267   }{
1268     \str_if_empty:NTF \l__stex_refs_in_str {
1269       \stex_debug:nn{sref}{in~empty;~autoref-on~#1}
1270       \__stex_refs_do_autoref:n{#1}
1271     }{
1272       \stex_debug:nn{sref}{in~non-empty;~srefin-on~#1}
1273       \__stex_refs_do_sref_in:n{#1}
1274     }
1275   }
1276 }
1277 }
1278
1279 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1280   \str_if_empty:NTF \l__stex_refs_uri_str {
1281     \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1282       \tl_set:Nn \l__stex_refs_return_tl {
1283         \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1284         \tl_if_empty:nTF\l__stex_refs_title_tl{
1285           ???
1286         }\l__stex_refs_title_tl
1287       }
1288     }
1289   }{
1290     \stex_debug:nn{sref}{\l__stex_refs_uri_str{~ == ~ #1 ~ ?}
1291     \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1292       \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1293       \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1294         \stex_debug:nn{sref}{success!}
1295         \tl_set:Nn \l__stex_refs_return_tl {
1296           \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1297           \tl_if_empty:nTF\l__stex_refs_title_tl{
1298             ???
1299           }\l__stex_refs_title_tl
1300         }

```

```

1301         \endinput
1302     }
1303 }
1304 }
1305 }
1306
1307 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1308   \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1309   \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1310   %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1311   \begingroup\catcode13=9\relax\catcode10=9\relax
1312   \str_if_empty:NTF \l__stex_refs_repob_str {
1313     \prop_if_exist:NTF \l_stex_current_repository_prop {
1314       \str_set:Nx \l_tmpa_str {
1315         \c_stex_mathhub_str /
1316         \prop_item:Nn \l_stex_current_repository_prop { id }
1317         / source / \l__stex_refs_in_str .sref
1318       }
1319     }{
1320       \str_set:Nx \l_tmpa_str {
1321         \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str . sref
1322       }
1323     }
1324   }{
1325     \str_set:Nx \l_tmpa_str {
1326       \c_stex_mathhub_str / \l__stex_refs_repob_str
1327       / source / \l__stex_refs_in_str . sref
1328     }
1329   }
1330   \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1331   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1332   \stex_debug:nn{sref}{File: \l_tmpa_str}
1333   \exp_args:No \IfFileExists \l_tmpa_str {
1334     \tl_clear:N \l__stex_refs_return_tl
1335     \str_set:Nn \l__stex_refs_id_str {#1}
1336     \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1337     \use:c{@ @ input}{\l_tmpa_str}
1338     \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1339       \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1340       \__stex_refs_do_autoref:n{
1341         \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1342       }
1343     }{
1344       \l__stex_refs_return_tl
1345     }
1346   }{
1347     \exp_args:Nnno \msg_warning:nnn{stex}{warning/smsmissing}\l_tmpa_str
1348     \__stex_refs_do_autoref:n{
1349       \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1350     }
1351   }
1352   \endgroup
1353 }
1354

```

```

1355 % \__stex_refs_args:n { #1 }
1356 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1357 %   \str_set:Nx \l_tmpa_str { #2 }
1358 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1359 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1360 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1361 %       \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1362 %         \str_clear:N \l_tmpa_str
1363 %       }
1364 %     }{
1365 %       \str_clear:N \l_tmpa_str
1366 %     }
1367 %   }{
1368 %     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1369 %     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1370 %     \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1371 %     \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1372 %       \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1373 %       \str_clear:N \l_tmpa_str
1374 %       \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str _seq} {
1375 %         \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1376 %           \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1377 %         }{
1378 %           \seq_map_break:n {
1379 %             \str_set:Nn \l_tmpa_str { ##1 }
1380 %           }
1381 %         }
1382 %       }
1383 %     }{
1384 %       \str_clear:N \l_tmpa_str
1385 %     }
1386 %   }
1387 %   \str_if_empty:NTF \l_tmpa_str {
1388 %     \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_li
1389 %   }{
1390 %     \str_if_eq:cNTF {sref_\l_tmpa_str _type} \c__stex_refs_ref_str {
1391 %       \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1392 %         \cs_if_exist:cTF{autoref}{
1393 %           \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1394 %         }{
1395 %           \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1396 %         }
1397 %       }{
1398 %         \ltx@ifpackageloaded{hyperref}{
1399 %           \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1400 %         }{
1401 %           \l__stex_refs_linktext_tl
1402 %         }
1403 %       }
1404 %     }{
1405 %       \ltx@ifpackageloaded{hyperref}{
1406 %         \href{\use:c{sref_url_\l_tmpa_str _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_
1407 %       }{
1408 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_ref

```

```

1409 %      }
1410 %      }
1411 %    }
1412 %  }{
1413 %    % TODO
1414 %  }
1415 %}

```

(End definition for `\sref`. This function is documented on page 78.)

`\srefsym`

```

1416 \NewDocumentCommand \srefsym { 0{} m}{
1417   \stex_get_symbol:n { #2 }
1418   \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1419 }
1420
1421 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1422
1423 % \str_if_exist:cTF {sref_sym_#2 _label_str }{
1424 %   \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1425 % }{
1426 %   \__stex_refs_args:n { #1 }
1427 %   \str_if_empty:NTF \l__stex_refs_indocument_str {
1428 %     \tl_if_exist:cTF{sref_sym_#2 _type}{
1429 %       % doc uri in \l_tmpb_str
1430 %       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1431 %       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1432 %         % reference
1433 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1434 %           \cs_if_exist:cTF{autoref}{
1435 %             \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1436 %           }{
1437 %             \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1438 %           }
1439 %         }{
1440 %           \ltx@ifpackageloaded{hyperref}{
1441 %             \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1442 %           }{
1443 %             \l__stex_refs_linktext_tl
1444 %           }
1445 %         }
1446 %       }{
1447 %         % URL
1448 %         \ltx@ifpackageloaded{hyperref}{
1449 %           \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1450 %           }{
1451 %             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_r
1452 %           }
1453 %         }
1454 %       }{
1455 %         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1456 %       }
1457 %     }{
1458 %       % TODO

```

```

1459 %   }
1460 %   }
1461 }

```

(End definition for \srefsym. This function is documented on page 78.)

\srefsymuri

```

1462 \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1463   #2%\__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1464 }

```

(End definition for \srefsymuri. This function is documented on page 78.)

```

1465 \</package>

```

Chapter 27

STEX -Modules Implementation

```
1466 <*package>
1467
1468 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1469
1470 <@@=stex_modules>
1471
1472   Warnings and error messages
1473   \msg_new:nnn{stex}{error/unknownmodule}{
1474     No~module~#1~found
1475   }
1476   \msg_new:nnn{stex}{error/syntax}{
1477     Syntax~error:~#1
1478   }
1479   \msg_new:nnn{stex}{error/siglanguage}{
1480     Module~#1~declares~signature~#2,~but~does~not~
1481     declare~its~language
1482   }
1483   \msg_new:nnn{stex}{warning/deprecated}{
1484     #1~is~deprecated;~please~use~#2~instead!
1485   }
1486   \msg_new:nnn{stex}{error/conflictingmodules}{
1487     Conflicting~imports~for~module~#1
1488   }
```

`\l_stex_current_module_str` The current module:

```
1488 \str_new:N \l_stex_current_module_str
```

(End definition for \l_stex_current_module_str. This variable is documented on page 80.)

`\l_stex_all_modules_seq` Stores all available modules

```
1489 \seq_new:N \l_stex_all_modules_seq
```

(End definition for \l_stex_all_modules_seq. This variable is documented on page 80.)

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1490 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1491   \str_if_empty:NTF \l_stex_current_module_str
1492   \prg_return_false: \prg_return_true:
1493 }

(End definition for \stex_if_in_module:TF. This function is documented on page 80.)

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1494 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1495   \prop_if_exist:cTF { c_stex_module_#1_prop }
1496   \prg_return_true: \prg_return_false:
1497 }

(End definition for \stex_if_module_exists:nTF. This function is documented on page 80.)

```

```

\stex_add_to_current_module:n
\STEXexport
1498 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1499   \stex_add_to_current_module:n { #1 }
1500   \stex_do_up_to_module:n { #1 }
1501 }}
1502 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1503
1504 \cs_new_protected:Nn \stex_add_to_current_module:n {
1505   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1506 }
1507 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1508 \cs_new_protected:Npn \STEXexport {
1509   \ExplSyntaxOn
1510   \__stex_modules_export:n
1511 }
1512 \cs_new_protected:Nn \__stex_modules_export:n {
1513   \ignorespacesandpars#1\ExplSyntaxOff
1514   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1515   \stex_smsmode_do:
1516 }
1517 \let \stex_module_export_helper:n \use:n
1518 \stex_deactivate_macro:Nn \STEXexport {module~environments}

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented
on page 80.)

```

```

\stex_add_constant_to_current_module:n
1519 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1520   \str_set:Nx \l_tmpa_str { #1 }
1521   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1522 }

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page
80.)

```

```

\stex_add_import_to_current_module:n
1523 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1524   \str_set:Nx \l_tmpa_str { #1 }
1525   \exp_args:Nno

```



```

1526 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1527 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1528 }
1529 }

```

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

`\stex_collect_imports:n`

```

1530 \cs_new_protected:Nn \stex_collect_imports:n {
1531 \seq_clear:N \l_stex_collect_imports_seq
1532 \__stex_modules_collect_imports:n {#1}
1533 }
1534 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1535 \seq_map_inline:cn {c_stex_module_#1_imports} {
1536 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1537 \__stex_modules_collect_imports:n { ##1 }
1538 }
1539 }
1540 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1541 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1542 }
1543 }

```

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

`\stex_do_up_to_module:n`

```

1544 \int_new:N \l__stex_modules_group_depth_int
1545 \cs_new_protected:Nn \stex_do_up_to_module:n {
1546 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1547 #1
1548 }{
1549 #1
1550 \expandafter \tl_gset:Nn
1551 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1552 \expandafter\expandafter\expandafter\endcsname
1553 \expandafter\expandafter\expandafter { \csname
1554 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1555 \aftergroup\__stex_modules_aftergroup_do:
1556 }
1557 }
1558 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1559 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1560 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1561 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1562 }}}
1563 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1564 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1565 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1566 }{
1567 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1568 \aftergroup\__stex_modules_aftergroup_do:
1569 }
1570 }
1571 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1572 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1573 }

(End definition for \stex_do_up_to_module:n. This function is documented on page 80.)

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

1574

(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.

```

1575 \str_new:N \l_stex_module_ns_str
1576 \str_new:N \l_stex_module_subpath_str
1577 \cs_new_protected:Nn __stex_modules_compute_namespace:nN {
1578   \seq_set_eq:NN \l_tmpa_seq #2
1579   % split off file extension
1580   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1581   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1582   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1583   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1584
1585   \bool_set_true:N \l_tmpa_bool
1586   \bool_while_do:Nn \l_tmpa_bool {
1587     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1588     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1589       {source} { \bool_set_false:N \l_tmpa_bool }
1590     }{}{
1591       \seq_if_empty:NT \l_tmpa_seq {
1592         \bool_set_false:N \l_tmpa_bool
1593       }
1594     }
1595   }
1596
1597   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1598   % \l_tmpa_seq <- sub-path relative to archive
1599   \str_if_empty:NTF \l_stex_module_subpath_str {
1600     \str_set:Nx \l_stex_module_ns_str {#1}
1601   }{
1602     \str_set:Nx \l_stex_module_ns_str {
1603       #1/\l_stex_module_subpath_str
1604     }
1605   }
1606 }
1607
1608 \cs_new_protected:Nn \stex_modules_current_namespace: {
1609   \str_clear:N \l_stex_module_subpath_str
1610   \prop_if_exist:NTF \l_stex_current_repository_prop {
1611     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1612     __stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1613   }{
1614     % split off file extension
1615     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1616     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str

```

```

1617 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1618 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1619 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1620 \str_set:Nx \l_stex_module_ns_str {
1621   file:/\stex_path_to_string:N \l_tmpa_seq
1622 }
1623 }
1624 }

```

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

27.1 The smodule environment

smodule arguments:

```

1625 \keys_define:nn { stex / module } {
1626   title      .tl_set:N      = \smodulename ,
1627   type       .str_set_x:N   = \smodulename ,
1628   id         .str_set_x:N   = \smoduleid ,
1629   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1630   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1631   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1632   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1633   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1634   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1635   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1636   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1637 }
1638
1639 \cs_new_protected:Nn \__stex_modules_args:n {
1640   \str_clear:N \smodulename
1641   \str_clear:N \smodulename
1642   \str_clear:N \smoduleid
1643   \str_clear:N \l_stex_module_ns_str
1644   \str_clear:N \l_stex_module_deprecate_str
1645   \str_clear:N \l_stex_module_lang_str
1646   \str_clear:N \l_stex_module_sig_str
1647   \str_clear:N \l_stex_module_creators_str
1648   \str_clear:N \l_stex_module_contributors_str
1649   \str_clear:N \l_stex_module_meta_str
1650   \str_clear:N \l_stex_module_srccite_str
1651   \keys_set:nn { stex / module } { #1 }
1652 }
1653
1654 % module parameters here? In the body?
1655

```

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

```

1656 \cs_new_protected:Nn \stex_module_setup:nn {
1657   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1658   \str_set:Nx \l_stex_module_name_str { #2 }
1659   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1660 \stex_if_in_module:TF {
1661   % Nested module
1662   \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1663   { ns } \l_stex_module_ns_str
1664   \str_set:Nx \l_stex_module_name_str {
1665     \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1666     { name } / \l_stex_module_name_str
1667   }
1668   \str_if_empty:NT \l_stex_module_lang_str {
1669     \str_set:Nx \l_stex_module_lang_str {
1670       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1671       { lang }
1672     }
1673   }
1674 }{
1675   % not nested:
1676   \str_if_empty:NT \l_stex_module_ns_str {
1677     \stex_modules_current_namespace:
1678     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1679     / {\l_stex_module_ns_str}
1680     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1681     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1682       \str_set:Nx \l_stex_module_ns_str {
1683         \stex_path_to_string:N \l_tmpa_seq
1684       }
1685     }
1686   }
1687 }

```

Next, we determine the language of the module:

```

1688 \str_if_empty:NT \l_stex_module_lang_str {
1689   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1690   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1691   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1692   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1693     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1694       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1695     }
1696   }
1697   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1698   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1699     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1700     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1701       inferred~from~file~name}
1702   }
1703 }
1704
1705 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1706   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1707 }}

```

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

```

1708 \str_if_empty:NTF \l_stex_module_sig_str {
1709   \exp_args:Nnx \prop_gset_from_keyval:cn {
1710     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1711   } {
1712     name      = \l_stex_module_name_str ,
1713     ns        = \l_stex_module_ns_str ,
1714     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1715     lang      = \l_stex_module_lang_str ,
1716     sig       = \l_stex_module_sig_str ,
1717     deprecate = \l_stex_module_deprecate_str ,
1718     meta      = \l_stex_module_meta_str
1719   }
1720   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1721   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1722   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1723   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1724   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1725 \str_if_empty:NT \l_stex_module_meta_str {
1726   \str_set:Nx \l_stex_module_meta_str {
1727     \c_stex_metatheory_ns_str ? Metatheory
1728   }
1729 }
1730 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1731   \bool_set_true:N \l_stex_in_meta_bool
1732   \exp_args:Nx \stex_add_to_current_module:n {
1733     \bool_set_true:N \l_stex_in_meta_bool
1734     \stex_activate_module:n {\l_stex_module_meta_str}
1735     \bool_set_false:N \l_stex_in_meta_bool
1736   }
1737   \stex_activate_module:n {\l_stex_module_meta_str}
1738   \bool_set_false:N \l_stex_in_meta_bool
1739 }
1740 }{
1741   \str_if_empty:NT \l_stex_module_lang_str {
1742     \msg_error:nnxx{stex}{error/siglanguage}{
1743       \l_stex_module_ns_str?\l_stex_module_name_str
1744     }{\l_stex_module_sig_str}
1745   }
1746   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1747   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1748     \stex_debug:nn{modules}{(already exists)}
1749   }{
1750     \stex_debug:nn{modules}{(needs loading)}
1751     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1752     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1753     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1754     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1755     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1756     \str_set:Nx \l_tmpa_str {
1757       \stex_path_to_string:N \l_tmpa_seq /

```

```

1758     \l_tmpa_str . \l_stex_module_sig_str .tex
1759   }
1760   \IfFileExists \l_tmpa_str {
1761     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1762       \str_clear:N \l_stex_current_module_str
1763       \seq_clear:N \l_stex_all_modules_seq
1764       \stex_debug:nn{modules}{Loading~signature}
1765     }
1766   }{
1767     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1768   }
1769 }
1770 \stex_if_smsmode:F {
1771   \stex_activate_module:n {
1772     \l_stex_module_ns_str ? \l_stex_module_name_str
1773   }
1774 }
1775 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1776 }
1777 \str_if_empty:NF \l_stex_module_deprecate_str {
1778   \msg_warning:nnxx{stex}{warning/deprecated}{
1779     Module~\l_stex_current_module_str
1780   }{
1781     \l_stex_module_deprecate_str
1782   }
1783 }
1784 \seq_put_right:Nx \l_stex_all_modules_seq {
1785   \l_stex_module_ns_str ? \l_stex_module_name_str
1786 }
1787 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1788 }

```

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

smodule (*env.*) The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}

1789 \cs_new_protected:Nn \__stex_modules_begin_module: {
1790   \stex_reactivate_macro:N \STEXexport
1791   \stex_reactivate_macro:N \importmodule
1792   \stex_reactivate_macro:N \symdecl
1793   \stex_reactivate_macro:N \notation
1794   \stex_reactivate_macro:N \symdef
1795
1796   \stex_debug:nn{modules}{
1797     New~module:\\
1798     Namespace:~\l_stex_module_ns_str\\
1799     Name:~\l_stex_module_name_str\\
1800     Language:~\l_stex_module_lang_str\\
1801     Signature:~\l_stex_module_sig_str\\
1802     Metatheory:~\l_stex_module_meta_str\\
1803     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1804   }
1805 }

```

```

1806 \stex_if_do_html:T{
1807   \begin{stex_annotate_env} {theory} {
1808     \l_stex_module_ns_str ? \l_stex_module_name_str
1809   }
1810
1811   \stex_annotate_invisible:nnn{header}{} {
1812     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1813     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1814     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1815       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1816     }
1817     \str_if_empty:NF \smoduletype {
1818       \stex_annotate:nnn{type}{\smoduletype}{}
1819     }
1820   }
1821 }
1822 % TODO: Inherit metatheory for nested modules?
1823 }
1824 \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}`

```

1825 \cs_new_protected:Nn \_stex_modules_end_module: {
1826   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module}
1827   \stex_reset_up_to_module:n \l_stex_current_module_str
1828   \stex_if_smsmode:T {
1829     \stex_persist:x {
1830       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1831         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1832       }
1833       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1834         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1835       }
1836       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1837         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1838       }
1839       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1840     }
1841     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module
1842     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1843   }
1844 }

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1845 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1846 \NewDocumentEnvironment { smodule } { 0 } { m } {
1847   \stex_module_setup:nn{#1}{#2}
1848   %\par
1849   \stex_if_smsmode:F{
1850     \tl_if_empty:NF \smoduletitle {
1851       \exp_args:No \stex_document_title:n \smoduletitle
1852     }

```

```

1853 \tl_clear:N \l_tmpa_tl
1854 \clist_map_inline:Nn \smodulotype {
1855   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1856     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1857   }
1858 }
1859 \tl_if_empty:NTF \l_tmpa_tl {
1860   \__stex_modules_smodule_start:
1861 }{
1862   \l_tmpa_tl
1863 }
1864 }
1865 \__stex_modules_begin_module:
1866 \str_if_empty:NF \smoduleid {
1867   \stex_ref_new_doc_target:n \smoduleid
1868 }
1869 \stex_smsmode_do:
1870 } {
1871   \__stex_modules_end_module:
1872   \stex_if_smsmode:F {
1873     \end{stex_annotate_env}
1874     \clist_set:Nn \l_tmpa_clist \smodulotype
1875     \tl_clear:N \l_tmpa_tl
1876     \clist_map_inline:Nn \l_tmpa_clist {
1877       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1878         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1879       }
1880     }
1881     \tl_if_empty:NTF \l_tmpa_tl {
1882       \__stex_modules_smodule_end:
1883     }{
1884       \l_tmpa_tl
1885     }
1886   }
1887 }

```

\stexpatchmodule

```

1888 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1889 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1890
1891 \newcommand\stexpatchmodule[3] [] {
1892   \str_set:Nx \l_tmpa_str{ #1 }
1893   \str_if_empty:NTF \l_tmpa_str {
1894     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1895     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1896   }{
1897     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1898     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1899   }
1900 }

```

(End definition for \stexpatchmodule. This function is documented on page [81](#).)

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1901 \NewDocumentCommand \STEXModule { m } {
1902   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1903   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1904   \tl_set:Nn \l_tmpa_tl {
1905     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1906   }
1907   \seq_map_inline:Nn \l_stex_all_modules_seq {
1908     \str_set:Nn \l_tmpb_str { ##1 }
1909     \str_if_eq:eeT { \l_tmpa_str } {
1910       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1911     } {
1912       \seq_map_break:n {
1913         \tl_set:Nn \l_tmpa_tl {
1914           \stex_invoke_module:n { ##1 }
1915         }
1916       }
1917     }
1918   }
1919   \l_tmpa_tl
1920 }
1921
1922 \cs_new_protected:Nn \stex_invoke_module:n {
1923   \stex_debug:nn{modules}{Invoking~module~#1}
1924   \peek_charcode_remove:NTF ! {
1925     \__stex_modules_invoke_uri:nN { #1 }
1926   } {
1927     \peek_charcode_remove:NTF ? {
1928       \__stex_modules_invoke_symbol:nn { #1 }
1929     } {
1930       \msg_error:nnx{stex}{error/syntax}{
1931         ?~or~!~expected~after~
1932         \c_backslash_str STEXModule{#1}
1933       }
1934     }
1935   }
1936 }
1937
1938 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1939   \str_set:Nn #2 { #1 }
1940 }
1941
1942 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1943   \stex_invoke_symbol:n{#1?#2}
1944 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 81.)

```

\stex_activate_module:n
1945 \bool_new:N \l_stex_in_meta_bool
1946 \bool_set_false:N \l_stex_in_meta_bool

```

```

1947 \cs_new_protected:Nn \stex_activate_module:n {
1948   \stex_debug:nn{modules}{Activating~module~#1}
1949   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1950     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1951     \use:c{ c_stex_module_#1_code }
1952   }
1953 }

```

(End definition for `\stex_activate_module:n`. This function is documented on page [82](#).)

`mmtinterface` (*env.*)

```

1954 \NewDocumentEnvironment { mmtinterface } { 0{} m m } {
1955   \begin{smodule}[#1]{#3}
1956   \str_set:Nx \l_stex_module_mmtfor_str {#2}
1957   \MMTinclude{#2}
1958   \stex_reactivate_macro:N \mmtdecl
1959   \stex_reactivate_macro:N \mmtdef
1960 }{
1961   \end{smodule}
1962 }

1963 \</package>

```

Chapter 28

sTeX -Module Inheritance Implementation

```
1964 <*package>
1965
1966 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1967
```

28.1 SMS Mode

```
1968 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1969 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1970 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1971 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1972
1973 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1974   \makeatletter
1975   \makeatother
1976   \ExplSyntaxOn
1977   \ExplSyntaxOff
1978   \rustexBREAK
1979 }
1980
1981 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1982   \symdef
1983   \importmodule
1984   \notation
1985   \symdecl
1986   \STEXexport
1987   \inlineass
1988   \inlinedef
1989   \inlineex
1990   \endinput
1991   \setnotation
```

```

1992 \copynotation
1993 \assign
1994 \renamedekl
1995 \donotcopy
1996 \instantiate
1997 \textsymdecl
1998 }
1999
2000 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
2001   \tl_to_str:n {
2002     smodule,
2003     copymodule,
2004     interpretmodule,
2005     realization,
2006     sdefinition,
2007     sexample,
2008     sassertion,
2009     sparagraph,
2010     mathstructure,
2011     extstructure,
2012     extstructure*
2013   }
2014 }

```

(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 83.)

`\stex_if_smsmode_p:`

`\stex_if_smsmode:TF`

```

2015 \bool_new:N \g__stex_smsmode_bool
2016 \bool_set_false:N \g__stex_smsmode_bool
2017 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2018   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2019 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

2020 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
2021   \vbox_set:Nn \l_tmpa_box {
2022     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
2023     \bool_gset_true:N \g__stex_smsmode_bool
2024     #2
2025     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2026   }
2027   \box_clear:N \l_tmpa_box
2028 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

2029 \quark_new:N \q__stex_smsmode_break
2030
2031 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
2032   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2033   \stex_smsmode_do:

```

```

2034 }
2035
2036 \cs_new_protected:Nn \__stex_smsmode_module:nn {
2037   \__stex_modules_args:n{#1}
2038   \stex_if_in_module:F {
2039     \str_if_empty:NF \l_stex_module_sig_str {
2040       \stex_modules_current_namespace:
2041       \str_set:Nx \l_stex_module_name_str { #2 }
2042       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2043         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2044         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2045         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2046         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2047         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2048         \str_set:Nx \l_tmpa_str {
2049           \stex_path_to_string:N \l_tmpa_seq /
2050           \l_tmpa_str . \l_stex_module_sig_str .tex
2051         }
2052         \IfFileExists \l_tmpa_str {
2053           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2054         }{
2055           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2056         }
2057       }
2058     }
2059   }
2060 }
2061
2062 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2063   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
2064   \tl_if_empty:nTF{#1}{
2065     \prop_if_exist:NTF \l_stex_current_repository_prop
2066     {
2067       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2068       \prg_return_true:
2069     } {
2070       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2071       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2072       \tl_if_empty:NT \l_tmpa_tl {
2073         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2074       }
2075       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
2076       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2077       \prg_return_true: \prg_return_false:
2078     }
2079   }\prg_return_true:
2080 }
2081
2082 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2083   \stex_filestack_push:n{#1}
2084   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2085   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2086   % ----- new -----
2087   \__stex_smsmode_in_smsmode:nn{#1}{

```

```

2088 \let\importmodule\__stex_smsmode_importmodule:
2089 \let\stex_module_setup:nn\__stex_smsmode_module:nn
2090 \let\__stex_modules_begin_module:\relax
2091 \let\__stex_modules_end_module:\relax
2092 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2093 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2094 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2095 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2096 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2097 \everyeof{\q__stex_smsmode_break\noexpand}
2098 \expandafter\expandafter\expandafter
2099 \stex_smsmode_do:
2100 \csname @ @ input\endcsname "#1"\relax
2101
2102 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2103   \stex_filestack_push:n{##1}
2104   \expandafter\expandafter\expandafter
2105   \stex_smsmode_do:
2106   \csname @ @ input\endcsname "##1"\relax
2107   \stex_filestack_pop:
2108 }
2109 }
2110 % ----- new -----
2111 \__stex_smsmode_in_smsmode:nn{#1} {
2112   #2
2113   % ----- new -----
2114   \begingroup
2115   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2116   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2117     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
2118       \stex_import_module_uri:nn ##1
2119       \stex_import_require_module:nnnn
2120       \l_stex_import_ns_str
2121       \l_stex_import_archive_str
2122       \l_stex_import_path_str
2123       \l_stex_import_name_str \endgroup
2124     }
2125   }
2126   \endgroup
2127   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
2128   % ----- new -----
2129   \everyeof{\q__stex_smsmode_break\noexpand}
2130   \expandafter\expandafter\expandafter
2131   \stex_smsmode_do:
2132   \csname @ @ input\endcsname "#1"\relax
2133 }
2134 \stex_filestack_pop:
2135 }

```

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

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

```

2136 \cs_new_protected:Npn \stex_smsmode_do: {

```

```

2137 \stex_if_smsmode:T {
2138   \__stex_smsmode_do:w
2139 }
2140 }
2141 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2142   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2143     \expandafter\if\expandafter\relax\noexpand#1
2144     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2145   \else\expandafter\__stex_smsmode_do:w\fi
2146 }{
2147   \__stex_smsmode_do:w % #1
2148 }
2149 }
2150 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2151   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
2152     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2153       #1\__stex_smsmode_do:w
2154     }{
2155       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2156         #1
2157       }{
2158         \cs_if_eq:NNTF \begin #1 {
2159           \__stex_smsmode_check_begin:n
2160         }{
2161           \cs_if_eq:NNTF \end #1 {
2162             \__stex_smsmode_check_end:n
2163           }{
2164             \__stex_smsmode_do:w
2165           }
2166         }
2167       }
2168     }
2169   }
2170 }
2171
2172 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2173   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2174     \begin{#1}
2175   }{
2176     \__stex_smsmode_do:w
2177   }
2178 }
2179 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2180   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2181     \end{#1}\__stex_smsmode_do:w
2182   }{
2183     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2184   }
2185 }

```

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

28.2 Inheritance

2186 <@@=stex_importmodule>

\stex_import_module_uri:nn

```
2187 \cs_new_protected:Nn \stex_import_module_uri:nn {
2188   \str_set:Nx \l_stex_import_archive_str { #1 }
2189   \str_set:Nn \l_stex_import_path_str { #2 }
2190
2191   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2192   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2193   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2194
2195   \stex_modules_current_namespace:
2196   \bool_lazy_all:nTF {
2197     {\str_if_empty_p:N \l_stex_import_archive_str}
2198     {\str_if_empty_p:N \l_stex_import_path_str}
2199     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2200   }{
2201     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2202     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2203   }{
2204     \str_if_empty:NT \l_stex_import_archive_str {
2205       \prop_if_exist:NT \l_stex_current_repository_prop {
2206         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2207       }
2208     }
2209     \str_if_empty:NTF \l_stex_import_archive_str {
2210       \str_if_empty:NF \l_stex_import_path_str {
2211         \stex_path_from_string:Nn \l_tmpb_seq {
2212           \l_stex_module_ns_str / .. / \l_stex_import_path_str
2213         }
2214         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2215         \str_replace_once:Nnn \l_stex_import_ns_str {file:} {file://}
2216       }
2217     }{
2218       \stex_require_repository:n \l_stex_import_archive_str
2219       \prop_get:cnN { c_stex_mathhub \l_stex_import_archive_str _manifest_prop } { ns }
2220       \l_stex_import_ns_str
2221       \str_if_empty:NF \l_stex_import_path_str {
2222         \str_set:Nx \l_stex_import_ns_str {
2223           \l_stex_import_ns_str / \l_stex_import_path_str
2224         }
2225       }
2226     }
2227   }
2228 }
```

(End definition for \stex_import_module_uri:nn. This function is documented on page 85.)

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str 2229 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 2230 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 2231 \str_new:N \l_stex_import_path_str
2232 \str_new:N \l_stex_import_ns_str

(End definition for \l_stex_import_name_str and others. These variables are documented on page 85.)


```

\stex_import_require_module:nnnn {{\ns}} {{\archive-ID}} {{\path}} {{\name}}
2233 \cs_new_protected:Nn \stex_import_require_module:nnnn {
2234 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2235
2236 \stex_debug:nn{requiremodule}{Here:\\~1:~#1\\~2:~#2\\~3:~#3\\~4:~#4}
2237
2238 \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2239 \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2240
2241 %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
2242
2243 % archive
2244 \str_set:Nx \l_tmpa_str { #2 }
2245 \str_if_empty:NTF \l_tmpa_str {
2246 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2247 \seq_put_right:Nn \l_tmpa_seq {...}
2248 } {
2249 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2250 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2251 \seq_put_right:Nn \l_tmpa_seq { source }
2252 }
2253
2254 % path
2255 \str_set:Nx \l_tmpb_str { #3 }
2256 \str_if_empty:NTF \l_tmpb_str {
2257 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2258
2259 \ltx@ifpackageloaded{babel} {
2260 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2261 { \language } \l_tmpb_str {
2262 \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2263 }
2264 } {
2265 \str_clear:N \l_tmpb_str
2266 }
2267
2268 \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2269 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2270 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2271 }{
2272 \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2273 \IfFileExists{ \l_tmpa_str.tex }{
2274 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2275 }{
2276 % try english as default
2277 \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2278 \IfFileExists{ \l_tmpa_str.en.tex }{
2279 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2280 }{
2281 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2282 }
2283 }
2284 }
2285

```

```

2286 } {
2287   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2288   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2289
2290   \ltx@ifpackageloaded{babel} {
2291     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2292       { \language } \l_tmpb_str {
2293       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2294     }
2295   } {
2296     \str_clear:N \l_tmpb_str
2297   }
2298
2299   \stex_path_canonicalize:N \l_tmpb_seq
2300   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2301
2302   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2303   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2304     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2305   }{
2306     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2307     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2308       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2309     }{
2310       % try english as default
2311       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2312       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2313         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2314       }{
2315         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2316         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2317           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2318         }{
2319           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2320           \IfFileExists{ \l_tmpa_str.tex }{
2321             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2322           }{
2323             % try english as default
2324             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2325             \IfFileExists{ \l_tmpa_str.en.tex }{
2326               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2327             }{
2328               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2329             }
2330           }
2331         }
2332       }
2333     }
2334   }
2335 }
2336
2337 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2338   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2339     \seq_clear:N \l_stex_all_modules_seq

```

```

2340     \str_clear:N \l_stex_current_module_str
2341     \str_set:Nx \l_tmpb_str { #2 }
2342     \str_if_empty:NF \l_tmpb_str {
2343       \stex_set_current_repository:n { #2 }
2344     }
2345     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2346   }
2347
2348   \stex_if_module_exists:nF { #1 ? #4 } {
2349     \msg_error:nnx{stex}{error/unknownmodule}{
2350       #1?#4~(in~file~\g__stex_importmodule_file_str)
2351     }
2352   }
2353 }
2354
2355 }
2356 \stex_activate_module:n { #1 ? #4 }
2357 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 85.)

`\importmodule`

```

2358 \NewDocumentCommand \importmodule { 0{} m } {
2359   \stex_import_module_uri:nn { #1 } { #2 }
2360   \stex_debug:nn{modules}{Importing~module:~
2361     \l_stex_import_ns_str ? \l_stex_import_name_str
2362   }
2363   \stex_import_require_module:nnnn
2364   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2365   { \l_stex_import_path_str } { \l_stex_import_name_str }
2366   \stex_if_smsmode:F {
2367     \stex_annotate_invisible:nnn
2368     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2369   }
2370   \exp_args:Nx \stex_add_to_current_module:n {
2371     \stex_import_require_module:nnnn
2372     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2373     { \l_stex_import_path_str } { \l_stex_import_name_str }
2374   }
2375   \exp_args:Nx \stex_add_import_to_current_module:n {
2376     \l_stex_import_ns_str ? \l_stex_import_name_str
2377   }
2378   \stex_smsmode_do:
2379   \ignorespacesandpars
2380 }
2381 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

2382 \NewDocumentCommand \usemodule { 0{} m } {
2383   \stex_if_smsmode:F {
2384     \stex_import_module_uri:nn { #1 } { #2 }
2385     \stex_import_require_module:nnnn
2386     { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```

```

2387     { \l_stex_import_path_str } { \l_stex_import_name_str }
2388     \stex_annotate_invisible:nnn
2389     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2390   }
2391   \stex_smsmode_do:
2392   \ignorespacesandpars
2393 }

```

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

```

2394 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2395   \tl_if_empty:nF{#2}{
2396     \clist_set:Nn \l_tmpa_clist {#2}
2397     \clist_map_inline:Nn \l_tmpa_clist {
2398       \tl_if_head_eq_charcode:nNTF {##1} [{
2399         #1 ##1
2400       }{
2401         #1{##1}
2402       }
2403     }
2404   }
2405 }
2406 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2407
2408
2409 \end{package}

```

Chapter 29

STEX -Symbols Implementation

```
2410 <*package>
2411
2412 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
2413
2414 Warnings and error messages
2415 \msg_new:nnn{stex}{error/wrongargs}{
2416   args~value~in~symbol~declaration~for~#1~
2417   needs~to~be~i,~a,~b~or~B,~but~#2~given
2418 }
2419 \msg_new:nnn{stex}{error/unknownsymbol}{
2420   No~symbol~#1~found!
2421 }
2422 \msg_new:nnn{stex}{error/seqlength}{
2423   Expected~#1~arguments;~got~#2!
2424 }
2425 \msg_new:nnn{stex}{error/unknownnotation}{
2426   Unknown~notation~#1~for~#2!
2427 }
```

29.1 Symbol Declarations

```
2427 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2428 \cs_new_protected:Nn \stex_all_symbols:n {
2429   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2430   \seq_map_inline:Nn \l_stex_all_modules_seq {
2431     \seq_map_inline:cn{c_stex_module_##1_constants}{
2432       \__stex_symdecl_all_symbols_cs{##1?####1}
2433     }
2434   }
2435 }
```

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

`\STEXsymbol`

```
2436 \NewDocumentCommand \STEXsymbol { m } {
2437   \stex_get_symbol:n { #1 }
2438   \exp_args:No
2439   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2440 }
```

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

`symdecl` arguments:

```
2441 \keys_define:nn { stex / symdecl } {
2442   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2443   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2444   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2445   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2446   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
2447   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
2448   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2449   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2450   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
2451   assoc     .choices:nn =
2452             {bin,binl,binr,pre,conj,pwconj}
2453             {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2454 }
2455
2456 \bool_new:N \l_stex_symdecl_make_macro_bool
2457
2458 \cs_new_protected:Nn \__stex_symdecl_args:n {
2459   \str_clear:N \l_stex_symdecl_name_str
2460   \str_clear:N \l_stex_symdecl_args_str
2461   \str_clear:N \l_stex_symdecl_deprecate_str
2462   \str_clear:N \l_stex_symdecl_reorder_str
2463   \str_clear:N \l_stex_symdecl_assoctype_str
2464   \bool_set_false:N \l_stex_symdecl_local_bool
2465   \tl_clear:N \l_stex_symdecl_type_tl
2466   \tl_clear:N \l_stex_symdecl_definiens_tl
2467   \clist_clear:N \l_stex_symdecl_argnames_clist
2468
2469   \keys_set:nn { stex / symdecl } { #1 }
2470 }
```

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

```
2471
2472 \NewDocumentCommand \symdecl { s m O{} } {
2473   \__stex_symdecl_args:n { #3 }
2474   \IfBooleanTF #1 {
2475     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2476   } {
2477     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2478   }
2479   \stex_symdecl_do:n { #2 }
2480   \stex_smsmode_do:
2481 }
```

```

2482
2483 \cs_new_protected:Nn \stex_symdecl_do:nn {
2484   \__stex_symdecl_args:n{#1}
2485   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2486   \stex_symdecl_do:n{#2}
2487 }
2488
2489 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2490 \cs_new_protected:Nn \stex_symdecl_do:n {
2491   \stex_if_in_module:F {
2492     % TODO throw error? some default namespace?
2493   }
2494
2495   \str_if_empty:NT \l_stex_symdecl_name_str {
2496     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2497   }
2498
2499   \prop_if_exist:cT { l_stex_symdecl_
2500     \l_stex_current_module_str ?
2501     \l_stex_symdecl_name_str
2502     _prop
2503   }{
2504     % TODO throw error (beware of circular dependencies)
2505   }
2506
2507   \prop_clear:N \l_tmpa_prop
2508   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2509   \seq_clear:N \l_tmpa_seq
2510   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2511   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2512
2513   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2514     \str_if_empty:NF \l_stex_module_deprecate_str {
2515       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2516     }
2517   }
2518   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2519
2520   \exp_args:No \stex_add_constant_to_current_module:n {
2521     \l_stex_symdecl_name_str
2522   }
2523
2524   % arity/args
2525   \int_zero:N \l_tmpb_int
2526
2527   \bool_set_true:N \l_tmpa_bool
2528   \str_map_inline:Nn \l_stex_symdecl_args_str {
2529     \token_case_meaning:NnF ##1 {
2530       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2531       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2532     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2533     {\tl_to_str:n a} {
2534         \bool_set_false:N \l_tmpa_bool
2535         \int_incr:N \l_tmpb_int
2536     }
2537     {\tl_to_str:n B} {
2538         \bool_set_false:N \l_tmpa_bool
2539         \int_incr:N \l_tmpb_int
2540     }
2541 }{
2542     \msg_error:nnxx{stex}{error/wrongargs}{
2543         \l_stex_current_module_str ?
2544         \l_stex_symdecl_name_str
2545     }{##1}
2546 }
2547 }
2548
2549 \bool_if:NTF \l_tmpa_bool {
2550     % possibly numeric
2551     \str_if_empty:NTF \l_stex_symdecl_args_str {
2552         \prop_put:Nnn \l_tmpa_prop { args } {}
2553         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2554     }{
2555         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2556         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2557         \str_clear:N \l_tmpa_str
2558         \int_step_inline:nn \l_tmpa_int {
2559             \str_put_right:Nn \l_tmpa_str i
2560         }
2561         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2562     }
2563 } {
2564     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2565     \prop_put:Nnx \l_tmpa_prop { arity }
2566     { \str_count:N \l_stex_symdecl_args_str }
2567 }
2568 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2569
2570 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2571     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2572 }{
2573     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2574 }
2575
2576 % argnames
2577
2578 \clist_clear:N \l_tmpa_clist
2579 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
2580     \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
2581         \clist_put_right:Nn \l_tmpa_clist {##1}
2582     }{
2583         \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl
2584         \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
2585     }

```



```

2586 }
2587 \prop_put:Nn \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
2588
2589 % semantic macro
2590
2591 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2592   \exp_args:Nx \stex_do_up_to_module:n {
2593     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2594       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2595     }}
2596   }
2597 }
2598
2599 \stex_debug:nn{symbols}{New~symbol:~
2600   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2601   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2602   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2603   Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2604 }
2605
2606 % circular dependencies require this:
2607 \stex_if_do_html:T {
2608   \stex_annotate_invisible:nnn {symdecl} {
2609     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2610   } {
2611     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2612       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2613     }
2614     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2615     \stex_annotate_invisible:nnn{macroname}{#1}{}
2616     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2617       \stex_annotate_invisible:nnn{definiens}{}
2618       {\l_stex_symdecl_definiens_tl$}
2619     }
2620     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2621       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2622     }
2623     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2624       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2625     }
2626   }
2627 }
2628 \prop_if_exist:cF {
2629   \l_stex_symdecl_
2630   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2631   _prop
2632 } {
2633   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2634     \__stex_symdecl_restore_symbol:nnnnnnnn
2635       {\l_stex_symdecl_name_str}
2636       { \prop_item:Nn \l_tmpa_prop {args} }
2637       { \prop_item:Nn \l_tmpa_prop {arity} }
2638       { \prop_item:Nn \l_tmpa_prop {assoc} }
2639       { \prop_item:Nn \l_tmpa_prop {defined} }

```

```

2640         {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2641         {\l_stex_current_module_str}
2642         { \prop_item:Nn \l_tmpa_prop {argnames} }
2643     }
2644 }
2645 }
2646 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2647     \prop_clear:N \l_tmpa_prop
2648     \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2649     \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2650     \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2651     \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2652     \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2653     \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2654     \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2655     \tl_if_empty:nF{#6}{
2656         \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2657     }
2658     \prop_set_eq:cN{\l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2659     \seq_clear:c{\l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2660 }

```

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

`\textsymdecl`

```

2661
2662 \keys_define:nn { stex / textsymdecl } {
2663     name      .str_set_x:N = \l__stex_symdecl_name_str ,
2664     type      .tl_set:N    = \l__stex_symdecl_type_tl
2665 }
2666
2667 \cs_new_protected:Nn \stex_textsymdecl_args:n {
2668     \str_clear:N \l__stex_symdecl_name_str
2669     \tl_clear:N \l__stex_symdecl_type_tl
2670     \clist_clear:N \l_stex_symdecl_argnames_clist
2671     \keys_set:nn { stex / textsymdecl } { #1 }
2672 }
2673
2674 \NewDocumentCommand \textsymdecl {m O{} m} {
2675     \stex_textsymdecl_args:n { #2 }
2676     \str_if_empty:NTF \l__stex_symdecl_name_str {
2677         \__stex_symdecl_args:n{name=#1,#2}
2678     }{
2679         \__stex_symdecl_args:n{#2}
2680     }
2681     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2682     \stex_symdecl_do:n{#1-sym}
2683     \stex_execute_in_module:n{
2684         \cs_set_nopar:cpn{#1name}{
2685             \ifvmode\hbox_unpack:N\c_empty_box\fi
2686             \ifmmode\hbox{#3}\else#3\fi\hspace
2687         }
2688         \cs_set_nopar:cpn{#1}{
2689             \ifmmode\csname#1-sym\expandafter\endcsname\else

```

```

2690     \ifvmode\hbox_unpack:N\c_empty_box\fi
2691     \symref{#1-sym}{#3}\expandafter\xspace
2692     \fi
2693   }
2694 }
2695 \stex_execute_in_module:x{
2696   \__stex_notation_restore_notation:nnnnn
2697   {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl_name_str{#1}}{0}
2698   {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{\neginfprec}}{0}
2699   \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2700   }}}
2701   {}
2702   {}
2703 }
2704 \stex_smsmode_do:
2705 }

```

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

`\stex_get_symbol:n`

```

2706 \str_new:N \l_stex_get_symbol_uri_str
2707
2708 \cs_new_protected:Nn \stex_get_symbol:n {
2709   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2710     \tl_set:Nn \l_tmpa_tl { #1 }
2711     \__stex_symdecl_get_symbol_from_cs:
2712   }{
2713     % argument is a string
2714     % is it a command name?
2715     \cs_if_exist:cTF { #1 }{
2716       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2717       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2718       \str_if_empty:NTF \l_tmpa_str {
2719         \exp_args:Nx \cs_if_eq:NNTF {
2720           \tl_head:N \l_tmpa_tl
2721         } \stex_invoke_symbol:n {
2722           \__stex_symdecl_get_symbol_from_cs:
2723         }{
2724           \__stex_symdecl_get_symbol_from_string:n { #1 }
2725         }
2726       } {
2727         \__stex_symdecl_get_symbol_from_string:n { #1 }
2728       }
2729     }{
2730       % argument is not a command name
2731       \__stex_symdecl_get_symbol_from_string:n { #1 }
2732       % \l_stex_all_symbols_seq
2733     }
2734   }
2735   \str_if_eq:eeF {
2736     \prop_item:cn {
2737       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2738     }{ deprecate }
2739   }{}{

```

```

2740     \msg_warning:nnxx{stex}{warning/deprecated}{
2741       Symbol~\l_stex_get_symbol_uri_str
2742     }{
2743       \prop_item:cn {l_stex_symdecl_l\l_stex_get_symbol_uri_str _prop}{ deprecate }
2744     }
2745   }
2746 }
2747
2748 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2749   \tl_set:Nn \l_tmpa_tl {
2750     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2751   }
2752   \str_set:Nn \l_tmpa_str { #1 }
2753
2754   %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2755
2756   \str_if_in:NnTF \l_tmpa_str ? {
2757     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2758     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2759     \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2760   }{
2761     \str_clear:N \l_tmpb_str
2762   }
2763   \str_if_empty:NnTF \l_tmpb_str {
2764     \seq_map_inline:Nn \l_stex_all_modules_seq {
2765       \seq_map_inline:cn{c_stex_module_##1_constants}{
2766         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2767           \seq_map_break:n{\seq_map_break:n{
2768             \tl_set:Nn \l_tmpa_tl {
2769               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2770             }
2771           }}
2772         }
2773       }
2774     }
2775   }{
2776     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2777     \seq_map_inline:Nn \l_stex_all_modules_seq {
2778       \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2779         \seq_map_inline:cn{c_stex_module_##1_constants}{
2780           \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2781             \seq_map_break:n{\seq_map_break:n{
2782               \tl_set:Nn \l_tmpa_tl {
2783                 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2784               }
2785             }}
2786           }
2787         }
2788       }
2789     }
2790   }
2791
2792   \l_tmpa_tl
2793 }

```

```

2794
2795 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2796   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2797     { \tl_tail:N \l_tmpa_tl }
2798   \tl_if_single:NTF \l_tmpa_tl {
2799     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2800       \exp_after:wN \str_set:Nn \exp_after:wN
2801         \l_stex_get_symbol_uri_str \l_tmpa_tl
2802     }{
2803       % TODO
2804       % tail is not a single group
2805     }
2806   }{
2807     % TODO
2808     % tail is not a single group
2809   }
2810 }

```

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

29.2 Notations

```

2811 <@@=stex_notation>
      notation arguments:
2812 \keys_define:nn { stex / notation } {
2813   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2814   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2815   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2816   op       .tl_set:N = \l__stex_notation_op_tl ,
2817   primary .bool_set:N = \l__stex_notation_primary_bool ,
2818   primary .default:n = {true} ,
2819   hints    .str_set_x:N = \l__stex_notation_hints_str,
2820   unknown .code:n = \str_set:Nx
2821     \l__stex_notation_variant_str \l_keys_key_str
2822 }
2823
2824 \cs_new_protected:Nn \_stex_notation_args:n {
2825   % \str_clear:N \l__stex_notation_lang_str
2826   \str_clear:N \l__stex_notation_variant_str
2827   \str_clear:N \l__stex_notation_prec_str
2828   \str_clear:N \l__stex_notation_hints_str
2829   \tl_clear:N \l__stex_notation_op_tl
2830   \bool_set_false:N \l__stex_notation_primary_bool
2831
2832   \keys_set:nn { stex / notation } { #1 }
2833 }

```

\notation

```

2834 \NewDocumentCommand \notation { s m O{}} {
2835   \_stex_notation_args:n { #3 }
2836   \tl_clear:N \l_stex_symdecl_definiens_tl
2837   \stex_get_symbol:n { #2 }
2838   \tl_set:Nn \l_stex_notation_after_do_tl {

```

```

2839 \__stex_notation_final:
2840 \IfBooleanTF#1{
2841   \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2842 }{}
2843 \stex_smsmode_do:\ignorespacesandpars
2844 }
2845 \stex_notation_do:nnnnn
2846 { \prop_item:cn {\l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2847 { \prop_item:cn { \l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2848 { \l__stex_notation_variant_str }
2849 { \l__stex_notation_prec_str }
2850 }
2851 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nnnnn

```

2852 \seq_new:N \l__stex_notation_precedences_seq
2853 \tl_new:N \l__stex_notation_opprec_tl
2854 \int_new:N \l__stex_notation_currarg_int
2855 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2856
2857 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2858   \let\STEXInternalCurrentSymbolStr\relax
2859   \seq_clear:N \l__stex_notation_precedences_seq
2860   \tl_clear:N \l__stex_notation_opprec_tl
2861   \str_set:Nx \l__stex_notation_args_str { #1 }
2862   \str_set:Nx \l__stex_notation_arity_str { #2 }
2863   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2864   \str_set:Nx \l__stex_notation_prec_str { #4 }
2865
2866   % precedences
2867   \str_if_empty:NTF \l__stex_notation_prec_str {
2868     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2869       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2870     }{
2871       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2872     }
2873   } {
2874     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2875       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2876       \int_step_inline:nn { \l__stex_notation_arity_str } {
2877         \exp_args:NNo
2878         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2879       }
2880     }{
2881       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2882       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2883         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2884         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2885           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2886             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2887           \seq_map_inline:Nn \l_tmpa_seq {
2888             \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }

```

```

2889     }
2890   }
2891   }{
2892     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2893       \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2894     }{
2895       \tl_set:No \l__stex_notation_opprec_tl { 0 }
2896     }
2897   }
2898 }
2899 }
2900
2901 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2902 \int_step_inline:nn { \l__stex_notation_arity_str } {
2903   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2904     \exp_args:NNo
2905     \seq_put_right:No \l__stex_notation_precedences_seq {
2906       \l__stex_notation_opprec_tl
2907     }
2908   }
2909 }
2910 \tl_clear:N \l_stex_notation_dummyargs_tl
2911
2912 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2913   \exp_args:NNe
2914   \cs_set:Npn \l_stex_notation_macrocode_cs {
2915     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2916     { \l__stex_notation_suffix_str }
2917     { \l__stex_notation_opprec_tl }
2918     { \exp_not:n { #5 } }
2919   }
2920   \l_stex_notation_after_do_tl
2921 }{
2922   \str_if_in:NnTF \l__stex_notation_args_str b {
2923     \exp_args:Nne \use:nn
2924     {
2925       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2926       \cs_set:Npn \l__stex_notation_arity_str } { {
2927         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2928         { \l__stex_notation_suffix_str }
2929         { \l__stex_notation_opprec_tl }
2930         { \exp_not:n { #5 } }
2931       } }
2932   }{
2933     \str_if_in:NnTF \l__stex_notation_args_str B {
2934       \exp_args:Nne \use:nn
2935       {
2936         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2937         \cs_set:Npn \l__stex_notation_arity_str } { {
2938           \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2939           { \l__stex_notation_suffix_str }
2940           { \l__stex_notation_opprec_tl }
2941           { \exp_not:n { #5 } }
2942         } }

```

```

2943   }{
2944     \exp_args:Nne \use:nn
2945     {
2946       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2947       \cs_set:Npn \l__stex_notation_arity_str } { {
2948         \STEXInternalTermMathOMAiiai { \STEXInternalCurrentSymbolStr }
2949         { \l__stex_notation_suffix_str }
2950         { \l__stex_notation_opprec_tl }
2951         { \exp_not:n { #5 } }
2952       } }
2953     }
2954   }
2955
2956   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2957   \int_zero:N \l__stex_notation_currarg_int
2958   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2959   \__stex_notation_arguments:
2960 }
2961 }

```

(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

```

2962 \cs_new_protected:Nn \__stex_notation_arguments: {
2963   \int_incr:N \l__stex_notation_currarg_int
2964   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2965     \l_stex_notation_after_do_tl
2966   }{
2967     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2968     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2969     \str_if_eq:VnTF \l_tmpa_str a {
2970       \__stex_notation_argument_assoc:nn{a}
2971     }{
2972       \str_if_eq:VnTF \l_tmpa_str B {
2973         \__stex_notation_argument_assoc:nn{B}
2974       }{
2975         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2976         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2977           { \STEXInternalTermMathArgiii
2978             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2979             { \l_tmpb_str }
2980             { ###\int_use:N \l__stex_notation_currarg_int }
2981           }
2982         }
2983         \__stex_notation_arguments:
2984       }
2985     }
2986   }
2987 }

```

(End definition for __stex_notation_arguments:.)

_stex_notation_argument_assoc:nn

```

2988 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {

```



```

2989
2990 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2991   {\l__stex_notation_arity_str}{
2992     #2
2993   }
2994 \int_zero:N \l_tmpa_int
2995 \tl_clear:N \l_tmpa_tl
2996 \str_map_inline:Nn \l__stex_notation_args_str {
2997   \int_incr:N \l_tmpa_int
2998   \tl_put_right:Nx \l_tmpa_tl {
2999     \str_if_eq:nnTF {##1}{a}{ } {} }{
3000     \str_if_eq:nnTF {##1}{B}{ } {} }{
3001     {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
3002     }
3003   }
3004 }
3005 }
3006 \exp_after:wN\exp_after:wN\exp_after:wN \def
3007 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
3008 \exp_after:wN\exp_after:wN\exp_after:wN ##
3009 \exp_after:wN\exp_after:wN\exp_after:wN 1
3010 \exp_after:wN\exp_after:wN\exp_after:wN ##
3011 \exp_after:wN\exp_after:wN\exp_after:wN 2
3012 \exp_after:wN\exp_after:wN\exp_after:wN {
3013   \exp_after:wN \exp_after:wN \exp_after:wN
3014   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
3015     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3016   }
3017 }
3018
3019 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
3020 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
3021   \STEXInternalTermMathAssocArgiiii
3022   { \int_use:N \l__stex_notation_currarg_int }
3023   { \l_tmpa_str }
3024   { ####\int_use:N \l__stex_notation_currarg_int }
3025   { \l_tmpa_cs {####1} {####2} }
3026   {#1}
3027 } }
3028 \__stex_notation_arguments:
3029 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

3030 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
3031   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
3032   \cs_set_nopar:Npn {#3}{#4}
3033   \tl_if_empty:nF {#5}{
3034     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
3035     }
3036   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
3037     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
3038   }

```

```

3039 }
3040
3041 \cs_new_protected:Nn \__stex_notation_final: {
3042
3043   \stex_execute_in_module:x {
3044     \__stex_notation_restore_notation:nnnnn
3045     {\l_stex_get_symbol_uri_str}
3046     {\l__stex_notation_suffix_str}
3047     {\l__stex_notation_arity_str}
3048     {
3049       \exp_after:wN \exp_after:wN \exp_after:wN
3050       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3051       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3052     }
3053     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3054   }
3055
3056   \stex_debug:nn{symbols}{
3057     Notation~\l__stex_notation_suffix_str
3058     ~for~\l_stex_get_symbol_uri_str^^J
3059     Operator~precedence:~\l__stex_notation_opprec_tl^^J
3060     Argument~precedences:~
3061     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3062     Notation: \cs_meaning:c {
3063       stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3064       \l__stex_notation_suffix_str
3065       _cs
3066     }
3067   }
3068   % HTML annotations
3069   \stex_if_do_html:T {
3070     \stex_annotate_invisible:nnn { notation }
3071     { \l_stex_get_symbol_uri_str } {
3072       \stex_annotate_invisible:nnn { notationfragment }
3073       { \l__stex_notation_suffix_str }{}
3074       \stex_annotate_invisible:nnn { precedence }
3075       { \l__stex_notation_prec_str }{}
3076
3077       \int_zero:N \l_tmpa_int
3078       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3079       \tl_clear:N \l_tmpa_tl
3080       \int_step_inline:nn { \l__stex_notation_arity_str }{
3081         \int_incr:N \l_tmpa_int
3082         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3083         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
3084         \str_if_eq:VnTF \l_tmpb_str a {
3085           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3086             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3087             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3088           } }
3089         }{
3090           \str_if_eq:VnTF \l_tmpb_str B {
3091             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3092               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,

```

```

3093         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{\}
3094     } }
3095     }{
3096         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3097             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{\}
3098         } }
3099     }
3100 }
3101 }
3102 \stex_annotate_invisible:nnn { notationcomp }{ }{
3103     \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3104     $ \exp_args:Nno \use:nn { \use:c {
3105         stex_notation_ \STEXInternalCurrentSymbolStr
3106         \c_hash_str \l__stex_notation_suffix_str_cs
3107     } } { \l_tmpa_tl } $
3108 }
3109 \tl_if_empty:NF \l__stex_notation_op_tl {
3110     \stex_annotate_invisible:nnn { notationopcomp }{ }{
3111         $\l__stex_notation_op_tl$
3112     }
3113 }
3114 }
3115 }
3116 }

```

(End definition for `_stex_notation_final:`.)

`\setnotation`

```

3117 \keys_define:nn { stex / setnotation } {
3118 % lang .tl_set_x:N = \l__stex_notation_lang_str ,
3119 variant .tl_set_x:N = \l__stex_notation_variant_str ,
3120 unknown .code:n = \str_set:Nx
3121     \l__stex_notation_variant_str \l_keys_key_str
3122 }
3123
3124 \cs_new_protected:Nn \_stex_setnotation_args:n {
3125 % \str_clear:N \l__stex_notation_lang_str
3126 \str_clear:N \l__stex_notation_variant_str
3127 \keys_set:nn { stex / setnotation } { #1 }
3128 }
3129
3130 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
3131 \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
3132     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
3133     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
3134 }
3135 }
3136
3137 \cs_new_protected:Nn \stex_setnotation:n {
3138 \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
3139 { \l__stex_notation_variant_str }{
3140     \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
3141     \stex_debug:nn {notations}{
3142         Setting~default~notation~

```

```

3143         {\l__stex_notation_variant_str }~for~
3144         #1 \\\
3145         \expandafter\meaning\csname
3146         l_stex_symdecl_#1 _notations\endcsname
3147     }
3148 }{
3149     \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
3150 }
3151 }
3152
3153 \NewDocumentCommand \setnotation {m m} {
3154     \stex_get_symbol:n { #1 }
3155     \_stex_setnotation_args:n { #2 }
3156     \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3157     \stex_smsmode_do:\ignorespacesandpars
3158 }
3159
3160 \cs_new_protected:Nn \stex_copy_notations:nn {
3161     \stex_debug:nn {notations}{
3162         Copying~notations~from~#2~to~#1\\
3163         \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3164     }
3165     \tl_clear:N \l_tmpa_tl
3166     \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3167         \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
3168     }
3169     \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3170         \stex_debug:nn{Here}{Here:~##1}
3171         \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3172         \edef \l_tmpa_tl {
3173             \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3174             \exp_after:wN\exp_after:wN\exp_after:wN {
3175                 \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3176             }
3177         }
3178
3179         \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3180         \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
3181         \exp_after:wN { \l_tmpa_tl }
3182
3183         \edef \l_tmpa_tl {
3184             \exp_after:wN \exp_not:n \exp_after:wN {
3185                 \l_tmpa_tl {##### 1}{##### 2}
3186             }
3187         }
3188
3189         \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3190
3191     \stex_execute_in_module:x {
3192         \_stex_notation_restore_notation:nnnnn
3193         {#1}{##1}
3194         { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
3195         { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
3196         {

```

```

3197         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3198             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3199             }
3200         }
3201     }\endgroup
3202 }
3203 }
3204
3205 \NewDocumentCommand \copynotation {m m} {
3206     \stex_get_symbol:n { #1 }
3207     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3208     \stex_get_symbol:n { #2 }
3209     \exp_args:Noo
3210     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3211     \stex_smsmode_do:\ignorespacesandpars
3212 }
3213

```

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

\symdef

```

3214 \keys_define:nn { stex / symdef } {
3215     name .str_set_x:N = \l_stex_symdecl_name_str ,
3216     args .str_set_x:N = \l_stex_symdecl_args_str ,
3217     type .tl_set:N = \l_stex_symdecl_type_tl ,
3218     def .tl_set:N = \l_stex_symdecl_definiens_tl ,
3219     reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3220     op .tl_set:N = \l__stex_notation_op_tl ,
3221     % lang .str_set_x:N = \l__stex_notation_lang_str ,
3222     variant .str_set_x:N = \l__stex_notation_variant_str ,
3223     prec .str_set_x:N = \l__stex_notation_prec_str ,
3224     argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3225     assoc .choices:nn =
3226         {bin,binl,binr,pre,conj,pwconj}
3227         {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3228     unknown .code:n = \str_set:Nx
3229         \l__stex_notation_variant_str \l_keys_key_str
3230 }
3231
3232 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3233     \str_clear:N \l_stex_symdecl_name_str
3234     \str_clear:N \l_stex_symdecl_args_str
3235     \str_clear:N \l_stex_symdecl_assoctype_str
3236     \str_clear:N \l_stex_symdecl_reorder_str
3237     \bool_set_false:N \l_stex_symdecl_local_bool
3238     \tl_clear:N \l_stex_symdecl_type_tl
3239     \tl_clear:N \l_stex_symdecl_definiens_tl
3240     \clist_clear:N \l_stex_symdecl_argnames_clist
3241     % \str_clear:N \l__stex_notation_lang_str
3242     \str_clear:N \l__stex_notation_variant_str
3243     \str_clear:N \l__stex_notation_prec_str
3244     \tl_clear:N \l__stex_notation_op_tl
3245
3246     \keys_set:nn { stex / symdef } { #1 }

```

```

3247 }
3248
3249 \NewDocumentCommand \symdef { m 0{} } {
3250   \_stex_notation_symdef_args:n { #2 }
3251   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3252   \stex_symdecl_do:n { #1 }
3253   \tl_set:Nn \l_stex_notation_after_do_tl {
3254     \_stex_notation_final:
3255     \stex_smsmode_do:\ignorespacesandpars
3256   }
3257   \str_set:Nx \l_stex_get_symbol_uri_str {
3258     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3259   }
3260   \exp_args:Nx \stex_notation_do:nnnnn
3261     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { args } }
3262     { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
3263     { \l__stex_notation_variant_str }
3264     { \l__stex_notation_prec_str}
3265 }
3266 \stex_deactivate_macro:Nn \symdef {module~environments}
3267
3268 \keys_define:nn { stex / mmtdef } {
3269   name .str_set_x:N = \l_stex_symdecl_name_str ,
3270   args .str_set_x:N = \l_stex_symdecl_args_str ,
3271   reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3272   op .tl_set:N = \l__stex_notation_op_tl ,
3273   % lang .str_set_x:N = \l__stex_notation_lang_str ,
3274   variant .str_set_x:N = \l__stex_notation_variant_str ,
3275   prec .str_set_x:N = \l__stex_notation_prec_str ,
3276   argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3277   assoc .choices:nn =
3278     {bin,binl,binr,pre,conj,pwconj}
3279     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3280   unknown .code:n = \str_set:Nx
3281     \l__stex_notation_variant_str \l_keys_key_str
3282 }
3283 \cs_new_protected:Nn \_stex_mmtdef_args:n {
3284   \str_clear:N \l_stex_symdecl_name_str
3285   \str_clear:N \l_stex_symdecl_args_str
3286   \str_clear:N \l_stex_symdecl_assoctype_str
3287   \str_clear:N \l_stex_symdecl_reorder_str
3288   \clist_clear:N \l_stex_symdecl_argnames_clist
3289   % \str_clear:N \l__stex_notation_lang_str
3290   \str_clear:N \l__stex_notation_variant_str
3291   \str_clear:N \l__stex_notation_prec_str
3292   \tl_clear:N \l__stex_notation_op_tl
3293
3294   \keys_set:nn { stex / mmtdef } { #1 }
3295 }
3296
3297 \NewDocumentCommand \mmtdef {m 0{} }{
3298   \_stex_mmtdef_args:n{ #2 }
3299   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3300   \str_if_empty:NT \l_stex_symdecl_name_str {

```

```

3301   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3302 }
3303 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3304 %   \stex_annotate:nnn{ OMID }{
3305 %     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3306 %   }{}
3307 %}
3308 \stex_symdecl_do:n { #1 }
3309 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3310   \stex_annotate:nnn{ OMID }{
3311     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3312   }{},
3313   \stex_annotate:nnn{ OMID }{
3314     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3315   }{}
3316 }
3317 \tl_set:Nn \l_stex_notation_after_do_tl {
3318   \__stex_notation_final:
3319   \stex_smsmode_do:\ignorespacesandpars
3320 }
3321 \str_set:Nx \l_stex_get_symbol_uri_str {
3322   \l_stex_current_module_str ? \l_stex_symdecl_name_str
3323 }
3324 \exp_args:Nx \stex_notation_do:nnnnn
3325 { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { args } }
3326 { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { arity } }
3327 { \l_stex_notation_variant_str }
3328 { \l__stex_notation_prec_str}
3329 }

```

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

29.3 Variables

```

3330 <@@=stex_variables>
3331
3332 \keys_define:nn { stex / vardef } {
3333   name      .str_set_x:N = \l__stex_variables_name_str ,
3334   args      .str_set_x:N = \l__stex_variables_args_str ,
3335   type      .tl_set:N    = \l__stex_variables_type_tl ,
3336   def       .tl_set:N    = \l__stex_variables_def_tl ,
3337   op        .tl_set:N    = \l__stex_variables_op_tl ,
3338   prec      .str_set_x:N = \l__stex_variables_prec_str ,
3339   reorder   .str_set_x:N = \l__stex_variables_reorder_str ,
3340   argnames  .clist_set:N = \l__stex_variables_argnames_clist ,
3341   assoc     .choices:nn  =
3342     {bin,binl,binr,pre,conj,pwconj}
3343     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3344   bind      .choices:nn  =
3345     {forall,exists}
3346     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3347 }
3348
3349 \cs_new_protected:Nn \__stex_variables_args:n {

```

```

3350 \str_clear:N \l__stex_variables_name_str
3351 \str_clear:N \l__stex_variables_args_str
3352 \str_clear:N \l__stex_variables_prec_str
3353 \str_clear:N \l__stex_variables_assoctype_str
3354 \str_clear:N \l__stex_variables_reorder_str
3355 \str_clear:N \l__stex_variables_bind_str
3356 \tl_clear:N \l__stex_variables_type_tl
3357 \tl_clear:N \l__stex_variables_def_tl
3358 \tl_clear:N \l__stex_variables_op_tl
3359 \clist_clear:N \l__stex_variables_argnames_clist
3360
3361 \keys_set:nn { stex / vardef } { #1 }
3362 }
3363
3364 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
3365   \__stex_variables_args:n {#2}
3366   \str_if_empty:NT \l__stex_variables_name_str {
3367     \str_set:Nx \l__stex_variables_name_str { #1 }
3368   }
3369   \prop_clear:N \l_tmpa_prop
3370   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3371
3372   \int_zero:N \l_tmpb_int
3373   \bool_set_true:N \l_tmpa_bool
3374   \str_map_inline:Nn \l__stex_variables_args_str {
3375     \token_case_meaning:NnF ##1 {
3376       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3377       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3378       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3379       {\tl_to_str:n a} {
3380         \bool_set_false:N \l_tmpa_bool
3381         \int_incr:N \l_tmpb_int
3382       }
3383       {\tl_to_str:n B} {
3384         \bool_set_false:N \l_tmpa_bool
3385         \int_incr:N \l_tmpb_int
3386       }
3387     }{
3388       \msg_error:nxxx{stex}{error/wrongargs}{
3389         variable~\l__stex_variables_name_str
3390       }{##1}
3391     }
3392   }
3393   \bool_if:NTF \l_tmpa_bool {
3394     % possibly numeric
3395     \str_if_empty:NTF \l__stex_variables_args_str {
3396       \prop_put:Nnn \l_tmpa_prop { args } {}
3397       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3398     }{
3399       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3400       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3401       \str_clear:N \l_tmpa_str
3402       \int_step_inline:nn \l_tmpa_int {
3403         \str_put_right:Nn \l_tmpa_str i

```



```

3404     }
3405     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3406     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3407   }
3408 } {
3409   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3410   \prop_put:Nnx \l_tmpa_prop { arity }
3411   { \str_count:N \l__stex_variables_args_str }
3412 }
3413 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3414 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
3415
3416 % argnames
3417
3418 \clist_clear:N \l_tmpa_clist
3419 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {
3420   \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3421     \clist_put_right:Nn \l_tmpa_clist { ##1 }
3422   } {
3423     \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3424     \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist { \c_dollar_str \l_tmpa_tl }
3425   }
3426 }
3427 \prop_put:Nnx \l_tmpa_prop { argnames } { \clist_use:Nn \l_tmpa_clist , }
3428
3429
3430 \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop } \l_tmpa_prop
3431
3432 \tl_if_empty:NF \l__stex_variables_op_tl {
3433   \cs_set:cpx {
3434     stex_var_op_notation_ \l__stex_variables_name_str _cs
3435   } { \exp_not:N \comp { \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3436 }
3437
3438 \tl_set:Nn \l_stex_notation_after_do_tl {
3439   \exp_args:Nne \use:nn {
3440     \cs_generate_from_arg_count:cNnn { stex_var_notation_ \l__stex_variables_name_str _cs }
3441     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3442   } {{
3443     \exp_after:wN \exp_after:wN \exp_after:wN
3444     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3445     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3446   }}
3447 \stex_if_do_html:T {
3448   \stex_annotate_invisible:nnn { vardecl } { \l__stex_variables_name_str } {
3449     \stex_annotate_invisible:nnn { precedence }
3450     { \l__stex_variables_prec_str } {}
3451     \tl_if_empty:NF \l__stex_variables_type_tl { \stex_annotate_invisible:nnn { type } { } { $ \l
3452     \stex_annotate_invisible:nnn { args } { \l__stex_variables_args_str } {}
3453     \stex_annotate_invisible:nnn { macroname } { #1 } {}
3454     \tl_if_empty:NF \l__stex_variables_def_tl {
3455       \stex_annotate_invisible:nnn { definiens } {}
3456       { $ \l__stex_variables_def_tl $ }
3457   }

```

```

3458 \str_if_empty:NF \l__stex_variables_assoctype_str {
3459 \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}}
3460 }
3461 \str_if_empty:NF \l__stex_variables_reorder_str {
3462 \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}}
3463 }
3464 \int_zero:N \l_tmpa_int
3465 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3466 \tl_clear:N \l_tmpa_tl
3467 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3468 \int_incr:N \l_tmpa_int
3469 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3470 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3471 \str_if_eq:VnTF \l_tmpb_str a {
3472 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3473 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3474 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3475 } }
3476 }{
3477 \str_if_eq:VnTF \l_tmpb_str B {
3478 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3479 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3480 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3481 } }
3482 }{
3483 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3484 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3485 } }
3486 }
3487 }
3488 }
3489 \stex_annotate_invisible:nnn { notationcomp }{}{
3490 \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3491 $ \exp_args:Nno \use:nn { \use:c {
3492 stex_var_notation_\l__stex_variables_name_str _cs
3493 } } { \l_tmpa_tl } $
3494 }
3495 \tl_if_empty:NF \l__stex_variables_op_tl {
3496 \stex_annotate_invisible:nnn { notationopcomp }{}{
3497 $\l__stex_variables_op_tl$
3498 }
3499 }
3500 }
3501 \str_if_empty:NF \l__stex_variables_bind_str {
3502 \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3503 }
3504 }\ignorespacesandpars
3505 }
3506
3507 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3508 }
3509
3510 \cs_new:Nn \stex_reset:N {
3511 \tl_if_exist:NTF #1 {

```

```

3512     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3513   }{
3514     \let \exp_not:N #1 \exp_not:N \undefined
3515   }
3516 }
3517
3518 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3519   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3520   \exp_args:Nnx \use:nn {
3521     % TODO
3522     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3523       #2
3524     }
3525   }{
3526     \_stex_reset:N \varnot
3527     \_stex_reset:N \vartype
3528     \_stex_reset:N \vardefi
3529   }
3530 }
3531
3532 \NewDocumentCommand \vardef { s } {
3533   \IfBooleanTF#1 {
3534     \__stex_variables_do_complex:nn
3535   }{
3536     \__stex_variables_do_simple:nnn
3537   }
3538 }
3539
3540 \NewDocumentCommand \svar { 0{} m }{
3541   \tl_if_empty:nTF {#1}{
3542     \str_set:Nn \l_tmpa_str { #2 }
3543   }{
3544     \str_set:Nn \l_tmpa_str { #1 }
3545   }
3546   \_stex_term_omv:nn {
3547     var://\l_tmpa_str
3548   }{
3549     \exp_args:Nnx \use:nn {
3550       \def\comp{\_varcomp}
3551       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3552       \comp{ #2 }
3553     }{
3554       \_stex_reset:N \comp
3555       \_stex_reset:N \STEXInternalCurrentSymbolStr
3556     }
3557   }
3558 }
3559
3560
3561
3562 \keys_define:nn { stex / varseq } {
3563   name .str_set_x:N = \l__stex_variables_name_str ,
3564   args .int_set:N   = \l__stex_variables_args_int ,
3565   type .tl_set:N    = \l__stex_variables_type_tl ,

```

```

3566 mid .tl_set:N = \l__stex_variables_mid_tl ,
3567 bind .choices:nn =
3568 {forall,exists}
3569 {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3570 }
3571
3572 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3573 \str_clear:N \l__stex_variables_name_str
3574 \int_set:Nn \l__stex_variables_args_int 1
3575 \tl_clear:N \l__stex_variables_type_tl
3576 \str_clear:N \l__stex_variables_bind_str
3577
3578 \keys_set:nn { stex / varseq } { #1 }
3579 }
3580
3581 \NewDocumentCommand \varseq {m O{ } m m m}{
3582 \__stex_variables_seq_args:n { #2 }
3583 \str_if_empty:NT \l__stex_variables_name_str {
3584 \str_set:Nx \l__stex_variables_name_str { #1 }
3585 }
3586 \prop_clear:N \l_tmpa_prop
3587 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3588
3589 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3590 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3591 \msg_error:nnxx{stex}{error/seqlength}
3592 {\int_use:N \l__stex_variables_args_int}
3593 {\seq_count:N \l_tmpa_seq}
3594 }
3595 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3596 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3597 \msg_error:nnxx{stex}{error/seqlength}
3598 {\int_use:N \l__stex_variables_args_int}
3599 {\seq_count:N \l_tmpb_seq}
3600 }
3601 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3602 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3603
3604 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3605 \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3606
3607 % argnames
3608
3609 \clist_clear:N \l_tmpa_clist
3610 \int_step_inline:nn {\l__stex_variables_args_int} {
3611 \clist_put_right:Nn \l_tmpa_clist {##1}
3612 }
3613 \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3614
3615
3616
3617
3618 \exp_args:NNo \tl_set:N \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3619 \int_step_inline:nn \l__stex_variables_args_int {

```

```

3620 \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3621 }
3622 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3623 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3624 \tl_if_empty:NF \l__stex_variables_mid_tl {
3625 \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3626 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3627 }
3628 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3629 \int_step_inline:nn \l__stex_variables_args_int {
3630 \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3631 }
3632 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3633 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3634
3635
3636 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3637
3638 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3639
3640 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3641
3642 \int_step_inline:nn \l__stex_variables_args_int {
3643 \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3644 \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3645 }}
3646 }
3647
3648 \tl_set:Nx \l_tmpa_tl {
3649 \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3650 \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3651 }
3652 }
3653
3654 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3655
3656 \exp_args:Nno \use:nn {
3657 \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3658 \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3659
3660 \stex_debug:nn{sequences}{New~Sequence:~
3661 \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str _cs\endcsname\\~\\
3662 \prop_to_keyval:N \l_tmpa_prop
3663 }
3664 \prop_set_eq:cN {l_stex_symdecl_varseq://\l__stex_variables_name_str _prop}\l_tmpa_prop
3665
3666 \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3667 \tl_if_empty:NF \l__stex_variables_type_tl {
3668 \stex_annotate:nnn {type}{\l__stex_variables_type_tl$}
3669 }
3670 \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3671 \str_if_empty:NF \l__stex_variables_bind_str {
3672 \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3673 }

```

```

3674 \stex_annotate:nnn{startindex}{#3$}
3675 \stex_annotate:nnn{endindex}{#4$}
3676
3677 \tl_clear:N \l_tmpa_tl
3678 \int_step_inline:nn \l__stex_variables_args_int {
3679   \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3680     \stex_annotate:nnn{argmarker}{#1}{}
3681   } }
3682 }
3683 \stex_annotate_invisible:nnn { notationcomp }{}{
3684   \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3685   $ \exp_args:Nno \use:nn { \use:c {
3686     stex_varseq_\l__stex_variables_name_str _cs
3687   } } { \l_tmpa_tl } $
3688 }
3689 \stex_annotate_invisible:nnn { notationopcomp }{}{
3690   $ \prop_item:Nn \l_tmpa_prop { notation } $
3691 }
3692
3693 }}
3694
3695 \ignorespacesandpars
3696 }
3697
3698
3699 \keys_define:nn { stex / mmtdecl } {
3700   name      .str_set_x:N = \l_stex_symdecl_name_str ,
3701   args      .str_set_x:N = \l_stex_symdecl_args_str ,
3702   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
3703   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
3704   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
3705   assoc     .choices:nn =
3706     {bin,binl,binr,pre,conj,pwconj}
3707     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
3708 }
3709
3710 \cs_new_protected:Nn \_stex_mmtdecl_args:n {
3711   \str_clear:N \l_stex_symdecl_name_str
3712   \str_clear:N \l_stex_symdecl_args_str
3713   \str_clear:N \l_stex_symdecl_deprecate_str
3714   \str_clear:N \l_stex_symdecl_reorder_str
3715   \str_clear:N \l_stex_symdecl_assoctype_str
3716   \bool_set_false:N \l_stex_symdecl_local_bool
3717   \clist_clear:N \l_stex_symdecl_argnames_clist
3718
3719   \keys_set:nn { stex / symdecl } { #1 }
3720 }
3721
3722 \NewDocumentCommand \mmtdecl { s m O{} } {
3723   \_stex_mmtdecl_args:n{#3}
3724   \IfBooleanTF #1 {
3725     \bool_set_false:N \l_stex_symdecl_make_macro_bool
3726   } {
3727     \bool_set_true:N \l_stex_symdecl_make_macro_bool

```

```

3728 }
3729 \str_if_empty:NT \l_stex_symdecl_name_str {
3730   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3731 }
3732 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3733 %   \stex_annotate:nnn{ OMID }{
3734 %     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3735 %   }{}
3736 %}
3737 \stex_symdecl_do:n{#2}
3738 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3739   \stex_annotate:nnn{ OMID }{
3740     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3741   }{},
3742   \stex_annotate:nnn{ OMID }{
3743     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3744   }{}
3745 }
3746 \stex_smsmode_do:
3747 }
3748
3749 \stex_deactivate_macro:Nn \mmtdecl {mmtinterface~environments}
3750 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
3751
3752 </package>

```

Chapter 30

STEX -Terms Implementation

```
3753 <*package>
3754
3755 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3756
3757 <@@=stex_terms>
3758
3759 Warnings and error messages
3760 \msg_new:nnn{stex}{error/nonotation}{
3761   Symbol~#1~invoked,~but~has~no~notation~#2!
3762 }
3763 \msg_new:nnn{stex}{error/notationarg}{
3764   Error~in~parsing~notation~#1
3765 }
3766 \msg_new:nnn{stex}{error/noop}{
3767   Symbol~#1~has~no~operator~notation~for~notation~#2
3768 }
3769 \msg_new:nnn{stex}{error/notallowed}{
3770   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3771 }
3772 \msg_new:nnn{stex}{error/doubleargument}{
3773   Argument~#1~of~symbol~#2~already~assigned
3774 }
3775 \msg_new:nnn{stex}{error/overarity}{
3776   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3777 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3777
3778
3779 \bool_new:N \l_stex_allow_semantic_bool
3780 \bool_set_true:N \l_stex_allow_semantic_bool
3781
```



```

3782 \cs_new_protected:Nn \stex_invoke_symbol:n {
3783   \ifvmode\indent\fi
3784   \bool_if:NTF \l_stex_allow_semantic_bool {
3785     \str_if_eq:eeF {
3786       \prop_item:cn {
3787         l_stex_symdecl_#1_prop
3788       }{ deprecate }
3789     }{}{
3790       \msg_warning:nxxx{stex}{warning/deprecated}{
3791         Symbol~#1
3792       }{
3793         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3794       }
3795     }
3796     \if_mode_math:
3797       \exp_after:wN \__stex_terms_invoke_math:n
3798     \else:
3799       \exp_after:wN \__stex_terms_invoke_text:n
3800     \fi: { #1 }
3801   }{
3802     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3803   }
3804 }
3805
3806 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3807   \peek_charcode_remove:NTF ! {
3808     \__stex_terms_invoke_op_custom:nn {#1}
3809   }{
3810     \__stex_terms_invoke_custom:nn {#1}
3811   }
3812 }
3813
3814 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3815   \peek_charcode_remove:NTF ! {
3816     % operator
3817     \peek_charcode_remove:NTF * {
3818       % custom op
3819       \__stex_terms_invoke_op_custom:nn {#1}
3820     }{
3821       % op notation
3822       \peek_charcode:NTF [ {
3823         \__stex_terms_invoke_op_notation:nw {#1}
3824       }{
3825         \__stex_terms_invoke_op_notation:nw {#1}[]
3826       }
3827     }
3828   }{
3829     \peek_charcode_remove:NTF * {
3830       \__stex_terms_invoke_custom:nn {#1}
3831       % custom
3832     }{
3833       % normal
3834       \peek_charcode:NTF [ {
3835         \__stex_terms_invoke_notation:nw {#1}

```

```

3836     }{
3837       \__stex_terms_invoke_notation:nw {#1}[]
3838     }
3839   }
3840 }
3841 }
3842
3843
3844 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3845   \exp_args:Nnx \use:nn {
3846     \def\comp{\_comp}
3847     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3848     \bool_set_false:N \l_stex_allow_semantic_bool
3849     \stex_mathml_intent:nn{#1}{
3850       \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3851         \comp{ #2 }
3852       }
3853     }
3854   }{
3855     \stex_reset:N \comp
3856     \stex_reset:N \STEXInternalCurrentSymbolStr
3857     \bool_set_true:N \l_stex_allow_semantic_bool
3858   }
3859 }
3860
3861 \keys_define:nn { stex / terms } {
3862   % lang      .tl_set_x:N = \l_stex_notation_lang_str ,
3863   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3864   unknown .code:n      = \str_set:Nx
3865     \l_stex_notation_variant_str \l_keys_key_str
3866 }
3867
3868 \cs_new_protected:Nn \__stex_terms_args:n {
3869   % \str_clear:N \l_stex_notation_lang_str
3870   \str_clear:N \l_stex_notation_variant_str
3871
3872   \keys_set:nn { stex / terms } { #1 }
3873 }
3874
3875 \cs_new_protected:Nn \stex_find_notation:nn {
3876   \__stex_terms_args:n { #2 }
3877   \seq_if_empty:cTF {
3878     \l_stex_symdecl_ #1 _notations
3879   } {
3880     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3881   } {
3882     \str_if_empty:NTF \l_stex_notation_variant_str {
3883       \seq_get_left:cN { \l_stex_symdecl_ #1 _notations } \l_stex_notation_variant_str
3884     } {
3885       \seq_if_in:cxTF { \l_stex_symdecl_ #1 _notations } {
3886         \l_stex_notation_variant_str
3887       } {
3888         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3889       } {

```

```

3890         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3891         ~\l_stex_notation_variant_str
3892         }
3893     }
3894 }
3895 }
3896 }
3897
3898 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3899     \exp_args:Nnx \use:nn {
3900         \def\comp{\_comp}
3901         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3902         \stex_find_notation:nn { #1 }{ #2 }
3903         \bool_set_false:N \l_stex_allow_semantic_bool
3904         \cs_if_exist:cTF {
3905             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3906         }{
3907             \_stex_term_oms:nnn { #1 }{
3908                 #1 \c_hash_str \l_stex_notation_variant_str
3909             }{
3910                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3911             }
3912         }{
3913             \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3914                 \cs_if_exist:cTF {
3915                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3916                 }{
3917                     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3918                         \_stex_reset:N \comp
3919                         \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3920                         \_stex_reset:N \STEXInternalCurrentSymbolStr
3921                         \bool_set_true:N \l_stex_allow_semantic_bool
3922                     }
3923                     \def\comp{\_comp}
3924                     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3925                     \bool_set_false:N \l_stex_allow_semantic_bool
3926                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3927                 }{
3928                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3929                     ~\l_stex_notation_variant_str
3930                     }
3931                 }
3932             }{
3933                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3934             }
3935         }
3936     }{
3937         \_stex_reset:N \comp
3938         \_stex_reset:N \STEXInternalCurrentSymbolStr
3939         \bool_set_true:N \l_stex_allow_semantic_bool
3940     }
3941 }
3942
3943 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```

```

3944 \stex_find_notation:nn { #1 }{ #2 }
3945 \cs_if_exist:cTF {
3946   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3947 }{
3948   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3949     \_stex_reset:N \comp
3950     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3951     \_stex_reset:N \STEXInternalCurrentSymbolStr
3952     \bool_set_true:N \l_stex_allow_semantic_bool
3953   }
3954   \def\comp{\_comp}
3955   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3956   \bool_set_false:N \l_stex_allow_semantic_bool
3957   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3958 }{
3959   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3960     ~\l_stex_notation_variant_str
3961   }
3962 }
3963 }
3964
3965 \prop_new:N \l__stex_terms_custom_args_prop
3966 \clist_new:N \l_stex_argnames_seq
3967 \seq_new:N \l__stex_terms_tmp_seq
3968
3969 \cs_new_protected:Nn \__stex_terms_custom_comp:n{ \bool_set_false:N \l_stex_allow_semantic_bo
3970
3971 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3972   \exp_args:Nnx \use:nn {
3973     \def\comp{\__stex_terms_custom_comp:n}
3974     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3975     \prop_clear:N \l__stex_terms_custom_args_prop
3976     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3977     \prop_get:cnN {
3978       l_stex_symdecl_#1 _prop
3979     }{ args } \l_tmpa_str
3980     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
3981       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
3982     }
3983     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3984     \tl_set:Nn \arg { \__stex_terms_arg: }
3985     \str_if_empty:NTF \l_tmpa_str {
3986       \stex_mathml_intent:nn{#1}{
3987         \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3988       }
3989     }{
3990       \seq_clear:N \l__stex_terms_tmp_seq
3991       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
3992         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
3993         \bool_lazy_or:nnT{
3994           \str_if_eq_p:nn{a}{\str_item:Nn \l_tmpa_str{##1}}
3995         }{
3996           \str_if_eq_p:nn{B}{\str_item:Nn \l_tmpa_str{##1}}
3997         }{

```

```

3998         \tl_put_right:Nn \l__stex_terms_tmp_tl +
3999     }
4000     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4001 }
4002 \stex_mathml_intent:nn{
4003     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}]{ args }(
4004     \seq_use:Nn \l__stex_terms_tmp_seq ,
4005     )
4006 }{
4007     \str_if_in:NnTF \l_tmpa_str b {
4008         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4009     }{
4010         \str_if_in:NnTF \l_tmpa_str B {
4011             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4012         }{
4013             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4014         }
4015     }
4016 }
4017 }
4018 % TODO check that all arguments exist
4019 }{
4020     \_stex_reset:N \l_stex_argnames_seq
4021     \_stex_reset:N \STEXInternalCurrentSymbolStr
4022     \_stex_reset:N \arg
4023     \_stex_reset:N \comp
4024     \_stex_reset:N \l__stex_terms_custom_args_prop
4025     %\bool_set_true:N \l_stex_allow_semantic_bool
4026 }
4027 }
4028
4029 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
4030     \tl_if_empty:nTF {#2}{
4031         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
4032         \bool_set_true:N \l_tmpa_bool
4033         \bool_do_while:Nn \l_tmpa_bool {
4034             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
4035             \int_incr:N \l_tmpa_int
4036         }{
4037             \bool_set_false:N \l_tmpa_bool
4038         }
4039     }
4040     }{
4041         \int_set:Nn \l_tmpa_int { #2 }
4042     }
4043     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
4044     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
4045         \msg_error:nnxxx{stex}{error/overarity}
4046         {\int_use:N \l_tmpa_int}
4047         {\STEXInternalCurrentSymbolStr}
4048         {\str_count:N \l_tmpa_str}
4049     }
4050     \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
4051     \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {

```

```

4052 \bool_lazy_any:nF {
4053   {\str_if_eq_p:Vn \l_tmpa_str {a}}
4054   {\str_if_eq_p:Vn \l_tmpa_str {B}}
4055 }{
4056   \msg_error:nnxx{stex}{error/doubleargument}
4057   {\int_use:N \l_tmpa_int}
4058   {\STEXInternalCurrentSymbolStr}
4059 }
4060 }
4061 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
4062 \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
4063   \bool_set_true:N \l_stex_allow_semantic_bool
4064   \use:nn
4065 }
4066 {
4067 \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4068   \IfBooleanTF#1{
4069     \stex_annotate_invisible:n { %TODO
4070       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4071     }
4072   }{ %TODO
4073     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4074   }
4075 }}
4076 {\bool_set_false:N \l_stex_allow_semantic_bool}
4077 }
4078
4079
4080 \cs_new_protected:Nn \_stex_term_arg:nn {
4081   \bool_set_true:N \l_stex_allow_semantic_bool
4082   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
4083   \bool_set_false:N \l_stex_allow_semantic_bool
4084 }
4085
4086 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
4087   \exp_args:Nnx \use:nn
4088   { \int_set:Nn \l__stex_terms_downprec { #2 }
4089     \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4090       \_stex_term_arg:nn { #1 }{ #3 }
4091     }
4092   }
4093   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4094 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 88.)

`\STEXInternalTermMathAssocArgiiii`

```

4095 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4#5 {
4096   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
4097   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
4098   \tl_if_empty:nTF { #3 }{
4099     \STEXInternalTermMathArgiii{#5#1}{#2}{}
4100   }{
4101     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{

```

```

4102     \expandafter\if\expandafter\relax\noexpand#3
4103     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
4104   \else
4105     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
4106   \fi
4107   \l_tmpa_tl
4108 }{
4109   \__stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
4110 }
4111 }
4112 }
4113
4114 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nnn {
4115   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
4116   \str_if_empty:NTF \l_tmpa_str {
4117     \exp_args:Nx \cs_if_eq:NNTF {
4118       \tl_head:N #1
4119     } \stex_invoke_sequence:n {
4120       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
4121       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
4122       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
4123       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
4124       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
4125         \exp_not:n{\exp_args:Nnx \use:nn} {
4126           \exp_not:n {
4127             \def\comp{\_varcomp}
4128             \str_set:Nn \STEXInternalCurrentSymbolStr
4129             } {varseq://\l_tmpa_str}
4130             \exp_not:n{ ##1 }
4131           }{
4132             \exp_not:n {
4133               \_stex_reset:N \comp
4134               \_stex_reset:N \STEXInternalCurrentSymbolStr
4135             }
4136           }
4137         }}}
4138       \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4139       \seq_reverse:N \l_tmpa_seq
4140       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4141       \seq_map_inline:Nn \l_tmpa_seq {
4142         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4143           \exp_args:Nno
4144           \l_tmpa_cs { ##1 } \l_tmpa_tl
4145         }
4146       }
4147       \tl_set:Nx \l_tmpa_tl {
4148         \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4149           \exp_args:No \exp_not:n \l_tmpa_tl
4150         }
4151       }
4152       \exp_args:No\l_tmpb_tl\l_tmpa_tl
4153     }{
4154       \__stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4155     }

```

```

4156 } {
4157   \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4158 }
4159
4160 }
4161
4162 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nnn {
4163   \clist_set:Nn \l_tmpa_clist{ #2 }
4164   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4165     \tl_set:Nn \l_tmpa_tl {
4166       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4167         \_stex_term_arg:nn{A#3#1}{ #2 } }
4168     }
4169   }{
4170     \clist_reverse:N \l_tmpa_clist
4171     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4172     \tl_set:Nx \l_tmpa_tl {
4173       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4174         \_stex_term_arg:nn{A#3#1}{
4175           \exp_args:No \exp_not:n \l_tmpa_tl
4176         }
4177       }
4178     }
4179     \clist_map_inline:Nn \l_tmpa_clist {
4180       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4181         \l_tmpa_cs {
4182           \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4183             \_stex_term_arg:nn{A#3#1}{##1}
4184           }
4185         } \l_tmpa_tl
4186       }
4187     }
4188   }
4189   \exp_args:No\l_tmpb_tl\l_tmpa_tl
4190 }

```

(End definition for `\STEXInternalTermMathAssocArgiiii`. This function is documented on page 89.)

30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
4191 \tl_const:Nx \infprec {\int_use:N \c_max_int}
4192 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4193 \int_new:N \l__stex_terms_downprec
4194 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_terms_downprec`. These variables are documented on page 89.)

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
4195 \tl_set:Nn \l__stex_terms_left_bracket_str (
4196 \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

```

4197 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4198   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4199     \bool_set_false:N \l__stex_terms_brackets_done_bool
4200     #2
4201   } {
4202     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4203       \bool_if:NTF \l__stex_inarray_bool { #2 }{
4204         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4205         \dobrackets { #2 }
4206       }
4207     }{ #2 }
4208   }
4209 }

```

(End definition for _stex_terms_maybe_brackets:nn.)

\dobrackets

```

4210 \bool_new:N \l__stex_terms_brackets_done_bool
4211 %\RequirePackage{scalerel}
4212 \cs_new_protected:Npn \dobrackets #1 {
4213   %\ThisStyle{\if D\m@switch
4214   %   \exp_args:Nnx \use:nn
4215   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4216   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
4217   %   \else
4218   %     \exp_args:Nnx \use:nn
4219   %     {
4220   %       \bool_set_true:N \l__stex_terms_brackets_done_bool
4221   %       \int_set:Nn \l__stex_terms_downprec \infprec
4222   %       \l__stex_terms_left_bracket_str
4223   %       #1
4224   %     }
4225   %     {
4226   %       \bool_set_false:N \l__stex_terms_brackets_done_bool
4227   %       \l__stex_terms_right_bracket_str
4228   %       \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4229   %     }
4230   %\fi}
4231 }

```

(End definition for \dobrackets. This function is documented on page 89.)

\withbrackets

```

4232 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4233   \exp_args:Nnx \use:nn
4234   {
4235     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4236     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4237     #3
4238   }
4239 }

```

```

4240 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4241 {\l__stex_terms_left_bracket_str}
4242 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4243 {\l__stex_terms_right_bracket_str}
4244 }
4245 }

```

(End definition for `\withbrackets`. This function is documented on page 89.)

`\STEXinvisible`

```

4246 \cs_new_protected:Npn \STEXinvisible #1 {
4247 \stex_annotate_invisible:n { #1 }
4248 }

```

(End definition for `\STEXinvisible`. This function is documented on page 89.)

OMDoc terms:

`\STEXInternalTermMathOMSiiii`

```

4249 \cs_new_protected:Nn \_stex_term_oms:nnn {
4250 \stex_annotate:nnn{ OMID }{ #2 }{
4251 #3
4252 }
4253 }
4254
4255 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4256 \__stex_terms_maybe_brackets:nn { #3 }{
4257 \stex_mathml_intent:nn{#1} {
4258 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4259 }
4260 }
4261 }

```

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

`_stex_term_math_omv:nn`

```

4262 \cs_new_protected:Nn \_stex_term_omv:nn {
4263 \stex_annotate:nnn{ OMV }{ #1 }{
4264 #2
4265 }
4266 }

```

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

`\STEXInternalTermMathOMAiiii`

```

4267 \cs_new_protected:Nn \_stex_term_oma:nnn {
4268 \stex_annotate:nnn{ OMA }{ #2 }{
4269 #3
4270 }
4271 }
4272
4273 \cs_new_protected:Npn \STEXInternalTermMathOMAiiii #1#2#3#4 {
4274 \exp_args:Nnx \use:nn {
4275 \seq_clear:N \l__stex_terms_tmp_seq
4276 \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4277 \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {

```

```

4278     \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4279   }
4280   \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4281     \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4282     \bool_lazy_or:nnT{
4283       \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4284     }{
4285       \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4286     }{
4287       \tl_put_right:Nn \l__stex_terms_tmp_tl +
4288     }
4289     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4290   }
4291 }
4292 \__stex_terms_maybe_brackets:nn { #3 }{
4293   \stex_mathml_intent:nn{
4294     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4295       \seq_use:Nn \l__stex_terms_tmp_seq ,
4296     )
4297   }{
4298     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4299   }
4300 }
4301 }{
4302   \_stex_reset:N \l_stex_argnames_seq
4303 }
4304 }

```

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

`\STEXInternalTermMathOMBiiii`

```

4305 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4306   \stex_annotate:nnn{ OMBIND }{ #2 }{
4307     #3
4308   }
4309 }
4310
4311 \cs_new_protected:Npn \STEXInternalTermMathOMBiiii #1#2#3#4 {
4312   \exp_args:Nnx \use:nn {
4313     \seq_clear:N \l__stex_terms_tmp_seq
4314     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4315       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4316         \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4317       }
4318       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4319         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4320         \bool_lazy_or:nnT{
4321           \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4322         }{
4323           \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4324         }{
4325           \tl_put_right:Nn \l__stex_terms_tmp_tl +
4326         }
4327         \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl

```

```

4328   }
4329 }
4330 \__stex_terms_maybe_brackets:nn { #3 }{
4331   \stex_mathml_intent:nn{
4332     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4333       \seq_use:Nn \l__stex_terms_tmp_seq ,
4334     )
4335   }{
4336     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4337   }
4338 }
4339 }{
4340   \stex_reset:N \l_stex_argnames_seq
4341 }
4342 }

```

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

`\symref`
`\symname`

```

4343 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4344
4345 \keys_define:nn { stex / symname } {
4346   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
4347   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
4348   root     .tl_set_x:N      = \l__stex_terms_root_tl
4349 }
4350
4351 \cs_new_protected:Nn \stex_symname_args:n {
4352   \tl_clear:N \l__stex_terms_post_tl
4353   \tl_clear:N \l__stex_terms_pre_tl
4354   \tl_clear:N \l__stex_terms_root_str
4355   \keys_set:nn { stex / symname } { #1 }
4356 }
4357
4358 \NewDocumentCommand \symref { m m }{
4359   \let\compemph_uri_prev:\compemph@uri
4360   \let\compemph@uri\symrefemph@uri
4361   \STEXsymbol{#1}!\{ #2 }
4362   \let\compemph@uri\compemph_uri_prev:
4363 }
4364
4365 \NewDocumentCommand \synonym { 0{} m m }{
4366   \stex_symname_args:n { #1 }
4367   \let\compemph_uri_prev:\compemph@uri
4368   \let\compemph@uri\symrefemph@uri
4369   % TODO
4370   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4371   \let\compemph@uri\compemph_uri_prev:
4372 }
4373
4374 \NewDocumentCommand \symname { 0{} m }{
4375   \stex_symname_args:n { #1 }
4376   \stex_get_symbol:n { #2 }
4377   \str_set:Nx \l_tmpa_str {

```

```

4378 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4379 }
4380 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4381
4382 \let\compemph_uri_prev:\compemph@uri
4383 \let\compemph@uri\symrefemph@uri
4384 \exp_args:Nnx \use:nn
4385 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4386 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4387 } }
4388 \let\compemph@uri\compemph_uri_prev:
4389 }
4390
4391 \NewDocumentCommand \Symname { 0{} m }{
4392 \stex_symname_args:n { #1 }
4393 \stex_get_symbol:n { #2 }
4394 \str_set:Nx \l_tmpa_str {
4395 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4396 }
4397 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4398 \let\compemph_uri_prev:\compemph@uri
4399 \let\compemph@uri\symrefemph@uri
4400 \exp_args:Nnx \use:nn
4401 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
4402 \exp_after:wN \stex_capitalize:n \l_tmpa_str
4403 \l__stex_terms_post_tl
4404 } }
4405 \let\compemph@uri\compemph_uri_prev:
4406 }

```

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

30.3 Notation Components

```

4407 <@@=stex_notationcomps>

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

4408 \cs_new_protected:Npn \_comp #1 {
4409 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4410 \stex_html_backend:TF {
4411 \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4412 }{
4413 \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4414 }
4415 }
4416 }
4417
4418 \cs_new_protected:Npn \_varcomp #1 {
4419 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4420 \stex_html_backend:TF {
4421 \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4422 }{
4423 \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
4424 }

```

```

4425 }
4426 }
4427
4428 \def\comp{\_comp}
4429
4430 \cs_new_protected:Npn \compemph@uri #1 #2 {
4431   \compemph{ #1 }
4432 }
4433
4434
4435 \cs_new_protected:Npn \compemph #1 {
4436   #1
4437 }
4438
4439 \cs_new_protected:Npn \defemph@uri #1 #2 {
4440   \defemph{#1}
4441 }
4442
4443 \cs_new_protected:Npn \defemph #1 {
4444   \textbf{#1}
4445 }
4446
4447 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4448   \symrefemph{#1}
4449 }
4450
4451 \cs_new_protected:Npn \symrefemph #1 {
4452   \emph{#1}
4453 }
4454
4455 \cs_new_protected:Npn \varemp@uri #1 #2 {
4456   \varemp{#1}
4457 }
4458
4459 \cs_new_protected:Npn \varemp #1 {
4460   #1
4461 }

```

(End definition for `\comp` and others. These functions are documented on page 89.)

`\ellipses`

```

4462 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 89.)

```

\parray
\prmatrix 4463 \bool_new:N \l_stex_inarray_bool
\parrayline 4464 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 4465 \NewDocumentCommand \parray { m m } {
\parraycell 4466   \begingroup
4467   \bool_set_true:N \l_stex_inarray_bool
4468   \begin{array}{#1}
4469     #2
4470   \end{array}
4471   \endgroup

```

```

4472 }
4473
4474 \NewDocumentCommand \prmatrix { m } {
4475   \begingroup
4476   \bool_set_true:N \l_stex_inarray_bool
4477   \begin{matrix}
4478     #1
4479   \end{matrix}
4480   \endgroup
4481 }
4482
4483 \def \maybepline {
4484   \bool_if:NT \l_stex_inarray_bool {\hline}
4485 }
4486
4487 \def \parrayline #1 #2 {
4488   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
4489 }
4490
4491 \def \pmrow #1 { \parrayline{}{ #1 } }
4492
4493 \def \parraylineh #1 #2 {
4494   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
4495 }
4496
4497 \def \parraycell #1 {
4498   #1 \bool_if:NT \l_stex_inarray_bool {&}
4499 }

```

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

30.4 Variables

4500 <@@=stex_variables>

\stex_invoke_variable:n Invokes a variable

```

4501 \cs_new_protected:Nn \stex_invoke_variable:n {
4502   \if_mode_math:
4503     \exp_after:wN \__stex_variables_invoke_math:n
4504   \else:
4505     \exp_after:wN \__stex_variables_invoke_text:n
4506   \fi: {#1}
4507 }
4508
4509 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4510   \peek_charcode_remove:NTF ! {
4511     \__stex_variables_invoke_op_custom:nn {#1}
4512   }{
4513     \__stex_variables_invoke_custom:nn {#1}
4514   }
4515 }
4516
4517
4518 \cs_new_protected:Nn \__stex_variables_invoke_math:n {

```

```

4519 \peek_charcode_remove:NTF ! {
4520   \peek_charcode_remove:NTF ! {
4521     \peek_charcode:NTF [ {
4522       % TODO throw error
4523     }{
4524       \__stex_variables_invoke_op_custom:nn
4525     }
4526   }{
4527     \__stex_variables_invoke_op:n { #1 }
4528   }
4529 }{
4530   \peek_charcode_remove:NTF * {
4531     \__stex_variables_invoke_custom:nn { #1 }
4532   }{
4533     \__stex_variables_invoke_math_ii:n { #1 }
4534   }
4535 }
4536 }
4537
4538 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4539   \exp_args:Nnx \use:nn {
4540     \def\comp{\_varcomp}
4541     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4542     \bool_set_false:N \l_stex_allow_semantic_bool
4543     \stex_term_omv:nn {var://#1}{
4544       \comp{ #2 }
4545     }
4546   }{
4547     \stex_reset:N \comp
4548     \stex_reset:N \STEXInternalCurrentSymbolStr
4549     \bool_set_true:N \l_stex_allow_semantic_bool
4550   }
4551 }
4552
4553 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4554   \cs_if_exist:cTF {
4555     stex_var_op_notation_ #1 _cs
4556   }{
4557     \exp_args:Nnx \use:nn {
4558       \def\comp{\_varcomp}
4559       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4560       \stex_term_omv:nn { var://#1 }{
4561         \use:c{stex_var_op_notation_ #1 _cs }
4562       }
4563     }{
4564       \stex_reset:N \comp
4565       \stex_reset:N \STEXInternalCurrentSymbolStr
4566     }
4567   }{
4568     \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4569       \__stex_variables_invoke_math_ii:n {#1}
4570     }{
4571       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
4572     }

```



```

4573 }
4574 }
4575
4576 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4577   \cs_if_exist:cTF {
4578     stex_var_notation_#1_cs
4579   }{
4580     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4581       \_stex_reset:N \comp
4582       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4583       \_stex_reset:N \STEXInternalCurrentSymbolStr
4584       \bool_set_true:N \l_stex_allow_semantic_bool
4585     }
4586     \def\comp{\_varcomp}
4587     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4588     \bool_set_false:N \l_stex_allow_semantic_bool
4589     \use:c{stex_var_notation_#1_cs}
4590   }{
4591     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
4592   }
4593 }
4594
4595 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4596   \exp_args:Nnx \use:nn {
4597     \def\comp{\_varcomp}
4598     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4599     \prop_clear:N \l__stex_terms_custom_args_prop
4600     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4601     \prop_get:cnN {
4602       l_stex_symdecl_var://#1 _prop
4603     }{ args } \l_tmpa_str
4604     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4605     \tl_set:Nn \arg { \_stex_terms_arg: }
4606     \str_if_empty:NTF \l_tmpa_str {
4607       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4608     }{
4609       \str_if_in:NnTF \l_tmpa_str b {
4610         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4611       }{
4612         \str_if_in:NnTF \l_tmpa_str B {
4613           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4614         }{
4615           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4616         }
4617       }
4618     }
4619     % TODO check that all arguments exist
4620   }{
4621     \_stex_reset:N \STEXInternalCurrentSymbolStr
4622     \_stex_reset:N \arg
4623     \_stex_reset:N \comp
4624     \_stex_reset:N \l__stex_terms_custom_args_prop
4625     %\bool_set_true:N \l_stex_allow_semantic_bool
4626   }

```

```
4627 }
```

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

30.5 Sequences

```
4628 <@@=stex_sequences>
4629
4630 \cs_new_protected:Nn \stex_invoke_sequence:n {
4631   \peek_charcode_remove:NTF ! {
4632     \stex_term_omv:nn {varseq://#1}{
4633       \exp_args:Nnx \use:nn {
4634         \def\comp{\_varcomp}
4635         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4636         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4637       }{
4638         \_stex_reset:N \comp
4639         \_stex_reset:N \STEXInternalCurrentSymbolStr
4640       }
4641     }
4642   }{
4643     \bool_set_false:N \l_stex_allow_semantic_bool
4644     \def\comp{\_varcomp}
4645     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4646     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4647       \_stex_reset:N \comp
4648       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4649       \_stex_reset:N \STEXInternalCurrentSymbolStr
4650       \bool_set_true:N \l_stex_allow_semantic_bool
4651     }
4652     \use:c { stex_varseq_#1_cs }
4653   }
4654 }
4655 </package>
```

Chapter 31

STEX -Structural Features Implementation

```
4656 ⟨*package⟩
4657
4658 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4659
4660 Warnings and error messages
4661 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4662   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4663 }
4664 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4665   Symbol~#1~not~assigned~in~interpretmodule~#2
4666 }
4667 \msg_new:nnn{stex}{error/unknownstructure}{
4668   No~structure~#1~found!
4669 }
4670
4671 \msg_new:nnn{stex}{error/unknownfield}{
4672   No~field~#1~in~instance~#2~found!\\#3
4673 }
4674
4675 \msg_new:nnn{stex}{error/keyval}{
4676   Invalid~key=value~pair~#1
4677 }
4678 \msg_new:nnn{stex}{error/instantiate/missing}{
4679   Assignments~missing~in~instantiate:~#1
4680 }
4681 \msg_new:nnn{stex}{error/incompatible}{
4682   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4683 }
4684
```

31.1 Imports with modification

```

4685 <@@=stex_copymodule>
4686 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4687   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4688     \tl_set:Nn \l_tmpa_tl { #1 }
4689     \__stex_copymodule_get_symbol_from_cs:
4690   }{
4691     % argument is a string
4692     % is it a command name?
4693     \cs_if_exist:cTF { #1 }{
4694       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4695       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4696       \str_if_empty:NTF \l_tmpa_str {
4697         \exp_args:Nx \cs_if_eq:NNTF {
4698           \tl_head:N \l_tmpa_tl
4699         } \stex_invoke_symbol:n {
4700           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4701         }{
4702           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4703         }
4704       } {
4705         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4706       }
4707     }{
4708       % argument is not a command name
4709       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4710       % \l_stex_all_symbols_seq
4711     }
4712   }
4713 }
4714
4715 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4716   \str_set:Nn \l_tmpa_str { #1 }
4717   \bool_set_false:N \l_tmpa_bool
4718   \bool_if:NF \l_tmpa_bool {
4719     \tl_set:Nn \l_tmpa_tl {
4720       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4721     }
4722     \str_set:Nn \l_tmpa_str { #1 }
4723     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4724     \seq_map_inline:Nn #2 {
4725       \str_set:Nn \l_tmpb_str { ##1 }
4726       \str_if_eq:eeT { \l_tmpa_str } {
4727         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4728       } {
4729         \seq_map_break:n {
4730           \tl_set:Nn \l_tmpa_tl {
4731             \str_set:Nn \l_stex_get_symbol_uri_str {
4732               ##1
4733             }
4734           }
4735         }
4736       }

```

```

4737     }
4738     \l_tmpa_tl
4739   }
4740 }
4741
4742 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4743   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4744     { \tl_tail:N \l_tmpa_tl }
4745   \tl_if_single:NTF \l_tmpa_tl {
4746     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4747       \exp_after:wN \str_set:Nn \exp_after:wN
4748         \l_stex_get_symbol_uri_str \l_tmpa_tl
4749       \__stex_copymodule_get_symbol_check:n { #1 }
4750     }{
4751       % TODO
4752       % tail is not a single group
4753     }
4754   }{
4755     % TODO
4756     % tail is not a single group
4757   }
4758 }
4759
4760 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4761   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4762     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4763       :~\seq_use:Nn #1 {,~}
4764     }
4765   }
4766 }
4767
4768 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4769   % import module
4770   \stex_import_module_uri:nn { #1 } { #2 }
4771   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4772   \stex_import_require_module:nnnn
4773     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4774     { \l_stex_import_path_str } { \l_stex_import_name_str }
4775
4776   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4777   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4778
4779   % fields
4780   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4781   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4782     \seq_map_inline:cn {c_stex_module_###1_constants}{
4783       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4784         ###1 ? ####1
4785       }
4786     }
4787   }
4788
4789   % setup prop
4790   \seq_clear:N \l_tmpa_seq

```

```

4791 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4792   name      = \l_stex_current_copymodule_name_str ,
4793   module    = \l_stex_current_module_str ,
4794   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4795   includes  = \l_tmpa_seq %,
4796 % fields    = \l_tmpa_seq
4797 }
4798 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4799   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4800 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4801 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4802
4803 \stex_if_do_html:T {
4804   \begin{stex_annotate_env} {#4} {
4805     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4806   }
4807   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4808 }
4809 }
4810
4811 \cs_new_protected:Nn \stex_copymodule_end:n {
4812   % apply to every field
4813   \def \l_tmpa_cs ##1 ##2 {#1}
4814
4815   \tl_clear:N \__stex_copymodule_module_tl
4816   \tl_clear:N \__stex_copymodule_exec_tl
4817
4818   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4819   \seq_clear:N \__stex_copymodule_fields_seq
4820
4821   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4822     \seq_map_inline:cn {c_stex_module_##1_constants}{
4823
4824       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4825       \l_tmpa_cs{##1}{####1}
4826
4827       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4828         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4829         \stex_if_do_html:T {
4830           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4831             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4832           }
4833         }
4834       }{
4835         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4836       }
4837
4838       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4839       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4840       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4841
4842       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4843         \stex_if_do_html:T {
4844           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4845         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4846     }
4847 }
4848 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4849 }
4850
4851 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4852 \tl_put_right:Nx \__stex_copymodule_module_tl {
4853     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4854     \prop_set_from_keyval:cn {
4855         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4856     }{
4857         \prop_to_keyval:N \l_tmpa_prop
4858     }
4859 }
4860
4861 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4862     \stex_if_do_html:T {
4863         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4864             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4865         }
4866     }
4867     \tl_put_right:Nx \__stex_copymodule_module_tl {
4868         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4869             \stex_invoke_symbol:n {
4870                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4871             }
4872         }
4873     }
4874 }
4875
4876 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4877
4878 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4879     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4880 }
4881
4882 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4883     \stex_if_do_html:TF{
4884         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4885     }{
4886         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4887     }
4888 }
4889 }
4890 }
4891
4892
4893 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4894 \tl_put_left:Nx \__stex_copymodule_module_tl {
4895     \prop_set_from_keyval:cn {
4896         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4897 }{
4898     \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4899     }
4900   }
4901
4902   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4903     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4904   }
4905
4906   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4907   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4908   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4909
4910   \__stex_copymodule_exec_tl
4911   \stex_if_do_html:T {
4912     \end{stex_annotate_env}
4913   }
4914 }
4915
4916 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4917   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4918   \stex_deactivate_macro:Nn \symdecl {module~environments}
4919   \stex_deactivate_macro:Nn \symdef {module~environments}
4920   \stex_deactivate_macro:Nn \notation {module~environments}
4921   \stex_reactivate_macro:N \assign
4922   \stex_reactivate_macro:N \renamedekl
4923   \stex_reactivate_macro:N \donotcopy
4924   \stex_smsmode_do:
4925 }{
4926   \stex_copymodule_end:n {}
4927 }
4928
4929 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4930   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4931   \stex_deactivate_macro:Nn \symdecl {module~environments}
4932   \stex_deactivate_macro:Nn \symdef {module~environments}
4933   \stex_deactivate_macro:Nn \notation {module~environments}
4934   \stex_reactivate_macro:N \assign
4935   \stex_reactivate_macro:N \renamedekl
4936   \stex_reactivate_macro:N \donotcopy
4937   \stex_smsmode_do:
4938 }{
4939   \stex_copymodule_end:n {
4940     \tl_if_exist:cF {
4941       l__stex_copymodule_copymodule_##1?##2_def_tl
4942     }{
4943       \str_if_eq:eeF {
4944         \prop_item:cn{
4945           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4946         }{ true }{
4947           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4948             ##1?##2
4949           }{\l_stex_current_copymodule_name_str}
4950         }
4951       }
4952     }

```



```

4953 }
4954
4955 \iffalse \begin{stex_annotate_env} \fi
4956 \NewDocumentEnvironment {realization} { 0 } { m } {
4957   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4958   \stex_deactivate_macro:Nn \symdecl {module~environments}
4959   \stex_deactivate_macro:Nn \symdef {module~environments}
4960   \stex_deactivate_macro:Nn \notation {module~environments}
4961   \stex_reactivate_macro:N \donotcopy
4962   \stex_reactivate_macro:N \assign
4963   \stex_smsmode_do:
4964 } {
4965   \stex_import_module_uri:nn { #1 } { #2 }
4966   \tl_clear:N \__stex_copymodule_exec_tl
4967   \tl_set:Nx \__stex_copymodule_module_tl {
4968     \stex_import_require_module:nnnn
4969     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4970     { \l_stex_import_path_str } { \l_stex_import_name_str }
4971   }
4972
4973   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4974     \seq_map_inline:cn {c_stex_module_##1_constants}{
4975       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4976       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4977         \stex_if_do_html:T {
4978           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4979             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4980               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4981             }
4982           }
4983         }
4984         \tl_put_right:Nx \__stex_copymodule_module_tl {
4985           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4986         }
4987       }
4988     }
4989
4990   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4991
4992   \__stex_copymodule_exec_tl
4993   \stex_if_do_html:T {\end{stex_annotate_env}}
4994 }
4995
4996 \NewDocumentCommand \donotcopy { m } {
4997   \str_clear:N \l_stex_import_name_str
4998   \str_set:Nn \l_tmpa_str { #1 }
4999   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5000   \seq_map_inline:Nn \l_stex_all_modules_seq {
5001     \str_set:Nn \l_tmpb_str { ##1 }
5002     \str_if_eq:eeT { \l_tmpa_str } {
5003       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5004     } {
5005       \seq_map_break:n {
5006         \stex_if_do_html:T {

```

```

5007         \stex_if_smsmode:F {
5008             \stex_annotate_invisible:nnn{donotcopy}{##1}{
5009                 \stex_annotate:nnn{domain}{##1}{}}
5010         }
5011     }
5012 }
5013 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
5014 }
5015 }
5016 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
5017     \str_set:Nn \l_tmpb_str { #####1 }
5018     \str_if_eq:eeT { \l_tmpa_str } {
5019         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5020     } {
5021         \seq_map_break:n {\seq_map_break:n {
5022             \stex_if_do_html:T {
5023                 \stex_if_smsmode:F {
5024                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
5025                         \stex_annotate:nnn{domain}{
5026                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
5027                         }{}
5028                     }
5029                 }
5030             }
5031             \str_set:Nx \l_stex_import_name_str {
5032                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
5033             }
5034         }}
5035     }
5036 }
5037 }
5038 \str_if_empty:NTF \l_stex_import_name_str {
5039     % TODO throw error
5040 }{
5041     \stex_collect_imports:n {\l_stex_import_name_str }
5042     \seq_map_inline:Nn \l_stex_collect_imports_seq {
5043         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
5044         \seq_map_inline:cn {c_stex_module_###1_constants}{
5045             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
5046             \bool_lazy_any:nT {
5047                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
5048                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
5049                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
5050             }{
5051                 % TODO throw error
5052             }
5053         }
5054     }
5055     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
5056     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
5057     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
5058 }
5059 \stex_smsmode_do:
5060 }

```

```

5061
5062 \NewDocumentCommand \assign { m m }{
5063   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
5064   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
5065   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
5066   \stex_smsmode_do:
5067 }
5068
5069 \keys_define:nn { stex / renamedecl } {
5070   name          .str_set_x:N = \l_stex_renamedecl_name_str
5071 }
5072 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
5073   \str_clear:N \l_stex_renamedecl_name_str
5074   \keys_set:nn { stex / renamedecl } { #1 }
5075 }
5076
5077 \NewDocumentCommand \renamedecl { O{} m m }{
5078   \__stex_copymodule_renamedecl_args:n { #1 }
5079   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
5080   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
5081   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
5082   \str_if_empty:NTF \l_stex_renamedecl_name_str {
5083     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5084       \l_stex_get_symbol_uri_str
5085     } }
5086   } {
5087     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
5088       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
5089       \prop_set_eq:cc {l_stex_symdecl_
5090         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5091       _prop
5092       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
5093       \seq_set_eq:cc {l_stex_symdecl_
5094         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5095       _notations
5096       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
5097       \prop_put:cnx {l_stex_symdecl_
5098         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5099       _prop
5100       }{ name }{ \l_stex_renamedecl_name_str }
5101       \prop_put:cnx {l_stex_symdecl_
5102         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5103       _prop
5104       }{ module }{ \l_stex_current_module_str }
5105       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
5106         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5107       }
5108       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5109         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5110       } }
5111     }
5112   \stex_smsmode_do:
5113 }
5114

```

```

5115 \stex_deactivate_macro:Nn \assign {copymodules}
5116 \stex_deactivate_macro:Nn \renamedekl {copymodules}
5117 \stex_deactivate_macro:Nn \donotcopy {copymodules}
5118
5119

```

31.2 The feature environment

`structural@feature (env.)`

```

5120 <@@=stex_features>
5121
5122 \NewDocumentEnvironment{structural_feature_module}{m m m}{
5123   \stex_if_in_module:F {
5124     \msg_set:nnn{stex}{error/nomodule}{
5125       Structural~Feature~has~to~occur~in~a~module:\\
5126       Feature~#2~of~type~#1\\
5127       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
5128     }
5129     \msg_error:nn{stex}{error/nomodule}
5130   }
5131
5132   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
5133
5134   \stex_module_setup:nn{meta=NONE}{#2 - #1}
5135
5136   \stex_if_do_html:T {
5137     \begin{stex_annotate_env}{feature:#1}{\l_stex_feature_parent_str ? #2 - #1}
5138     \stex_annotate_invisible:nnn{header}{#3}
5139   }
5140 }{
5141   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5142   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5143   \stex_debug:nn{features}{
5144     Feature: \l_stex_last_feature_str
5145   }
5146   \stex_if_do_html:T {
5147     \end{stex_annotate_env}
5148   }
5149 }

```

31.3 Structure

`structure (env.)`

```

5150 <@@=stex_structures>
5151 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5152   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
5153     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
5154   }
5155   \prop_gput:cxn {c_stex_module_ \l_stex_current_module_str _structures}
5156   {#1}{#2}
5157 }
5158

```

```

5159 \keys_define:nn { stex / features / structure } {
5160   name          .str_set_x:N = \l__stex_structures_name_str ,
5161 }
5162
5163 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5164   \str_clear:N \l__stex_structures_name_str
5165   \keys_set:nn { stex / features / structure } { #1 }
5166 }
5167
5168 \NewDocumentEnvironment{mathstructure}{m O{}}{
5169   \__stex_structures_structure_args:n { #2 }
5170   \str_if_empty:NT \l__stex_structures_name_str {
5171     \str_set:Nx \l__stex_structures_name_str { #1 }
5172   }
5173   \stex_suppress_html:n {
5174     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5175     \exp_args:Nx \stex_symdecl_do:nn {
5176       name = \l__stex_structures_name_str ,
5177       def = {\STEXsymbol{module-type}}{
5178         \STEXInternalTermMathOMSiiii {
5179           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5180             { ns } ?
5181           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5182             { name } / \l__stex_structures_name_str - structure
5183         }{}{}{}
5184       }}
5185     }{ #1 }
5186   }
5187   \exp_args:Nnnx
5188   \begin{structural_feature_module}{ structure }
5189     { \l__stex_structures_name_str }{}
5190   \stex_smsmode_do:
5191 }{
5192   \end{structural_feature_module}
5193   \_stex_reset_up_to_module:n \l_stex_last_feature_str
5194   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5195   \seq_clear:N \l_tmpa_seq
5196   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5197     \seq_map_inline:cn{c_stex_module_##1_constants}{
5198       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5199     }
5200   }
5201   \exp_args:Nnno
5202   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5203   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5204   \stex_add_structure_to_current_module:nn
5205     \l__stex_structures_name_str
5206     \l_stex_last_feature_str
5207
5208   \stex_execute_in_module:x {
5209     \tl_set:cn { #1 }{
5210       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
5211     }
5212   }

```

```

5213 }
5214
5215 \cs_new:Nn \stex_invoke_structure:nn {
5216   \stex_invoke_symbol:n { #1?#2 }
5217 }
5218
5219 \cs_new_protected:Nn \stex_get_structure:n {
5220   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5221     \tl_set:Nn \l_tmpa_tl { #1 }
5222     \__stex_structures_get_from_cs:
5223   }{
5224     \cs_if_exist:cTF { #1 }{
5225       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5226       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5227       \str_if_empty:NTF \l_tmpa_str {
5228         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
5229           \__stex_structures_get_from_cs:
5230         }{
5231           \__stex_structures_get_from_string:n { #1 }
5232         }
5233       }{
5234         \__stex_structures_get_from_string:n { #1 }
5235       }
5236     }{
5237       \__stex_structures_get_from_string:n { #1 }
5238     }
5239   }
5240 }
5241
5242 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5243   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5244     { \tl_tail:N \l_tmpa_tl }
5245   \str_set:Nx \l_tmpa_str {
5246     \exp_after:wN \use_i:nn \l_tmpa_tl
5247   }
5248   \str_set:Nx \l_tmpb_str {
5249     \exp_after:wN \use_ii:nn \l_tmpa_tl
5250   }
5251   \str_set:Nx \l_stex_get_structure_str {
5252     \l_tmpa_str ? \l_tmpb_str
5253   }
5254   \str_set:Nx \l_stex_get_structure_module_str {
5255     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5256   }
5257 }
5258
5259 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5260   \tl_set:Nn \l_tmpa_tl {
5261     \msg_error:nnn{stex}{error/unknownstructure}{#1}
5262   }
5263   \str_set:Nn \l_tmpa_str { #1 }
5264   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5265
5266   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

5267 \prop_if_exist:cT {c_stex_module_##1_structures} {
5268 \prop_map_inline:cn {c_stex_module_##1_structures} {
5269 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5270 \prop_map_break:n{\seq_map_break:n{
5271 \tl_set:Nn \l_tmpa_tl {
5272 \str_set:Nn \l_stex_get_structure_str {##1?####1}
5273 \str_set:Nn \l_stex_get_structure_module_str {####2}
5274 }
5275 }}
5276 }
5277 }
5278 }
5279 }
5280 \l_tmpa_tl
5281 }

```

\instantiate

```

5282
5283 \NewDocumentEnvironment{usestructure}{m}{
5284 \stex_get_structure:n {#1}
5285 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5286 }{}
5287
5288 \keys_define:nn { stex / instantiate } {
5289 name .str_set_x:N = \l__stex_structures_name_str
5290 }
5291 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5292 \str_clear:N \l__stex_structures_name_str
5293 \keys_set:nn { stex / instantiate } { #1 }
5294 }
5295
5296 \NewDocumentEnvironment{extstructure}{m m O{}}{
5297 \begin{mathstructure}{#1}[#3]
5298 \seq_set_split:Nnn\__stex_structures_extstructure_imports_seq,{#2}
5299 \seq_map_inline:Nn\__stex_structures_extstructure_imports_seq {
5300 \stex_get_structure:n {##1}
5301 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5302 \stex_if_smsmode:F {
5303 \stex_annotate_invisible:nnn
5304 {import} {\l_stex_get_structure_module_str} {}
5305 }
5306 \exp_args:Nx \stex_add_import_to_current_module:n {
5307 \l_stex_get_structure_module_str
5308 }
5309 }
5310 \stex_smsmode_do:
5311 \ignorespacesandpars
5312 }{
5313 \end{mathstructure}
5314 }
5315
5316 \NewDocumentEnvironment{extstructure*}{m m O{}}{
5317 % TODO
5318 \begin{extstructure}{#1}[#2][#3]

```

```

5319 }{
5320   \end{extstructure}
5321 }
5322
5323 \NewDocumentCommand \instantiate {m O{} m m O{}}{
5324   \begin{group}
5325     \stex_get_structure:n {#3}
5326     \__stex_structures_instantiate_args:n { #2 }
5327     \str_if_empty:NT \l__stex_structures_name_str {
5328       \str_set:Nn \l__stex_structures_name_str { #1 }
5329     }
5330     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5331     \seq_clear:N \l__stex_structures_fields_seq
5332     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5333     \seq_map_inline:Nn \l_stex_collect_imports_seq {
5334       \seq_map_inline:cn {c_stex_module_##1_constants}{
5335         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
5336       }
5337     }
5338
5339     \tl_if_empty:nF{#5}{
5340       \seq_set_split:Nnn \l_tmpa_seq , {#5}
5341       \prop_clear:N \l_tmpa_prop
5342       \seq_map_inline:Nn \l_tmpa_seq {
5343         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5344         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5345           \msg_error:nnn{stex}{error/keyval}{##1}
5346         }
5347         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5348         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5349         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5350         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
5351         \exp_args:Nxx \str_if_eq:nnF
5352           {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5353           {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5354           \msg_error:nnxxx{stex}{error/incompatible}
5355           {\l__stex_structures_dom_str}
5356           {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5357           {\l_stex_get_symbol_uri_str}
5358           {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5359         }
5360         \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
5361       }
5362     }
5363
5364     \seq_map_inline:Nn \l__stex_structures_fields_seq {
5365       \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5366       \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5367
5368       \stex_add_constant_to_current_module:n {\l_tmpa_str}
5369       \stex_execute_in_module:x {
5370         \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
5371           name = \l_tmpa_str ,
5372           args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,

```



```

5373         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5374         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5375         argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5376     }
5377     \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5378 }
5379
5380 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5381     \stex_find_notation:nn{##1}{ }
5382     \stex_execute_in_module:x {
5383         \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5384     }
5385
5386     \stex_copy_control_sequence_ii:ccN
5387     {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5388     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5389     \l_tmpa_tl
5390     \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5391
5392
5393     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5394         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5395         \stex_execute_in_module:x {
5396             \tl_set:cn
5397             {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
5398             { \exp_args:No \exp_not:n \l_tmpa_cs}
5399         }
5400     }
5401
5402 }
5403
5404 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
5405 }
5406
5407 \stex_execute_in_module:x {
5408     \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
5409     domain = \l_stex_get_structure_module_str ,
5410     \prop_to_keyval:N \l_tmpa_prop
5411 }
5412 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
5413 }
5414 \stex_debug:nn{instantiate}{
5415     Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
5416     \prop_to_keyval:N \l_tmpa_prop
5417 }
5418 \exp_args:Nxx \stex_symdecl_do:nn {
5419     type={\STEXsymbol{module-type}}{
5420         \STEXInternalTermMathOMSiiii {
5421             \l_stex_get_structure_module_str
5422         }{ }{0}{ }
5423     }
5424 }{\l__stex_structures_name_str}
5425 % {
5426     \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures

```

```

5427         \tl_set:Nn \l_stex_notation_after_do_tl {\_stex_notation_final:}
5428         \stex_notation_do:nnnnn{}{0}{}{}{\comp{#4}}
5429     % }
5430     %\exp_args:Nx \notation{\l_stex_structures_name_str}{\comp{#5}}
5431 \endgroup
5432 \stex_smsmode_do:\ignorespacesandpars
5433 }
5434
5435 \cs_new_protected:Nn \stex_symbol_or_var:n {
5436     \cs_if_exist:cTF{#1}{
5437         \cs_set_eq:Nc \l_tmpa_tl { #1 }
5438         \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
5439         \str_if_empty:NTF \l_tmpa_str {
5440             \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
5441             \stex_invoke_variable:n {
5442                 \bool_set_true:N \l_stex_symbol_or_var_bool
5443                 \bool_set_false:N \l_stex_instance_or_symbol_bool
5444                 \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5445                 \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5446                 \str_set:Nx \l_stex_get_symbol_uri_str {
5447                     \exp_after:wN \use:n \l_tmpa_tl
5448                 }
5449             }{ % TODO \stex_invoke_varinstance:n
5450                 \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
5451                     \bool_set_true:N \l_stex_symbol_or_var_bool
5452                     \bool_set_true:N \l_stex_instance_or_symbol_bool
5453                     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5454                     \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5455                     \str_set:Nx \l_stex_get_symbol_uri_str {
5456                         \exp_after:wN \use:n \l_tmpa_tl
5457                     }
5458                 }{
5459                     \bool_set_false:N \l_stex_symbol_or_var_bool
5460                     \stex_get_symbol:n{#1}
5461                 }
5462             }
5463         }{
5464             \_stex_structures_symbolorvar_from_string:n{ #1 }
5465         }
5466     }{
5467         \_stex_structures_symbolorvar_from_string:n{ #1 }
5468     }
5469 }
5470
5471 \cs_new_protected:Nn \_stex_structures_symbolorvar_from_string:n {
5472     \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5473         \bool_set_true:N \l_stex_symbol_or_var_bool
5474         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5475     }{
5476         \bool_set_false:N \l_stex_symbol_or_var_bool
5477         \stex_get_symbol:n{#1}
5478     }
5479 }
5480

```

```

5481 \keys_define:nn { stex / varinstantiate } {
5482   name      .str_set_x:N = \l__stex_structures_name_str,
5483   bind      .choices:nn =
5484     {forall,exists}
5485     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5486
5487 }
5488 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5489   \str_clear:N \l__stex_structures_name_str
5490   \str_clear:N \l__stex_structures_bind_str
5491   \keys_set:nn { stex / varinstantiate } { #1 }
5492 }
5493
5494 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
5495   \beginingroup
5496     \stex_get_structure:n {#3}
5497     \__stex_structures_varinstantiate_args:n { #2 }
5498     \str_if_empty:NT \l__stex_structures_name_str {
5499       \str_set:Nn \l__stex_structures_name_str { #1 }
5500     }
5501     \stex_if_do_html:TF{
5502       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5503     }{\use:n}
5504     {
5505       \stex_if_do_html:T{
5506         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{\}
5507       }
5508       \seq_clear:N \l__stex_structures_fields_seq
5509       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5510       \seq_map_inline:Nn \l_stex_collect_imports_seq {
5511         \seq_map_inline:cn {c_stex_module_##1_constants}{
5512           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5513         }
5514       }
5515       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5516       \prop_clear:N \l_tmpa_prop
5517       \tl_if_empty:nF {#5} {
5518         \seq_set_split:Nnn \l_tmpa_seq , {#5}
5519         \seq_map_inline:Nn \l_tmpa_seq {
5520           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5521           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5522             \msg_error:nnn{stex}{error/keyval}{##1}
5523           }
5524           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
5525           \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5526           \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq:nn
5527           \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5528           \stex_if_do_html:T{
5529             \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5530               \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\} \l_stex_get_symbol_uri_str}{\}
5531           }
5532           \bool_if:NTF \l_stex_symbol_or_var_bool {
5533             \exp_args:Nxx \str_if_eq:nnF
5534               {\prop_item:cn{l_stex_symdecl \l__stex_structures_dom_str _prop}}{args}}

```

```

5535         {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}{\
5536         \msg_error:nnxxxx{stex}{error/incompatible}
5537         {\l__stex_structures_dom_str}
5538         {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
5539         {\l_stex_get_symbol_uri_str}
5540         {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}
5541     }
5542     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n
5543 }{
5544     \exp_args:Nxx \str_if_eq:nnF
5545         {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
5546         {\prop_item:cn{l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{args}}{\
5547         \msg_error:nnxxxx{stex}{error/incompatible}
5548         {\l__stex_structures_dom_str}
5549         {\prop_item:cn{l_stex_symdecl\_l__stex_structures_dom_str _prop}{args}}
5550         {\l_stex_get_symbol_uri_str}
5551         {\prop_item:cn{l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{args}}
5552     }
5553     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n
5554 }
5555 }
5556 }
5557 \tl_gclear:N \g__stex_structures_aftergroup_tl
5558 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5559     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl
5560     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5561     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5562         \stex_find_notation:nn{##1}{
5563         \cs_gset_eq:cc{g__stex_structures_tmpa\_l_tmpa_str _cs}
5564         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5565         \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa\_l
5566         \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5567         \cs_gset_eq:cc {g__stex_structures_tmpa_op\_l_tmpa_str _cs}
5568         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5569         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5570     }
5571 }
5572 }
5573 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5574     \prop_set_from_keyval:cn { l_stex_symdecl_ var://\l_tmpa_str _prop}{
5575         name      = \l_tmpa_str ,
5576         args      = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5577         arity     = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5578         assocs    = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5579         argnames  = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}} ,
5580     }
5581     \cs_set_eq:cc {stex_var_notation\_l_tmpa_str _cs}
5582     {g__stex_structures_tmpa\_l_tmpa_str _cs}
5583     \cs_set_eq:cc {stex_var_op_notation\_l_tmpa_str _cs}
5584     {g__stex_structures_tmpa_op\_l_tmpa_str _cs}
5585 }
5586 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5587 }
5588 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {

```

```

5589     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
5590         domain = \l_stex_get_structure_module_str ,
5591         \prop_to_keyval:N \l_tmpa_prop
5592     }
5593     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5594     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
5595         \exp_args:Nnx \exp_not:N \use:nn {
5596             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5597             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5598                 \exp_not:n{
5599                     \_varcomp{#4}
5600                 }
5601             }
5602         }{
5603             \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5604         }
5605     }
5606 }
5607 }
5608 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
5609 \aftergroup\g__stex_structures_aftergroup_tl
5610 \endgroup
5611 \stex_smsmode_do:\ignorespacesandpars
5612 }
5613
5614 \cs_new_protected:Nn \stex_invoke_instance:n {
5615     \peek_charcode_remove:NTF ! {
5616         \stex_invoke_symbol:n{#1}
5617     }{
5618         \_stex_invoke_instance:nn {#1}
5619     }
5620 }
5621
5622
5623 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5624     \peek_charcode_remove:NTF ! {
5625         \exp_args:Nnx \use:nn {
5626             \def\comp{\_varcomp}
5627             \use:c{l_stex_varinstance_#1_op_tl}
5628         }{
5629             \_stex_reset:N \comp
5630         }
5631     }{
5632         \_stex_invoke_varinstance:nn {#1}
5633     }
5634 }
5635
5636 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5637     \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5638         \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5639     }{
5640         \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5641         \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5642             \prop_to_keyval:N \l_tmpa_prop

```

```

5643     }
5644   }
5645 }
5646
5647 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
5648   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5649     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5650     \l_tmpa_tl
5651   }{
5652     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{ }
5653   }
5654 }

```

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

\stex_invoke_structure:nnn

```

5655 % #1: URI of the instance
5656 % #2: URI of the instantiated module
5657 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5658   \tl_if_empty:nTF{ #3 }{
5659     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5660       c_stex_feature_ #2 _prop
5661     }
5662     \tl_clear:N \l_tmpa_tl
5663     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5664     \seq_map_inline:Nn \l_tmpa_seq {
5665       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5666       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5667       \cs_if_exist:cT {
5668         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5669       }{
5670         \tl_if_empty:NF \l_tmpa_tl {
5671           \tl_put_right:Nn \l_tmpa_tl {,}
5672         }
5673         \tl_put_right:Nx \l_tmpa_tl {
5674           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5675         }
5676       }
5677     }
5678     \exp_args:No \mathstrut \l_tmpa_tl
5679   }{
5680     \stex_invoke_symbol:n{#1/#3}
5681   }
5682 }

```

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

```

5683 </package>

```

Chapter 32

STEX -Statements Implementation

```
5684 <*package>
5685
5686 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5687
5688 <@@=stex_statements>
5689
5690 Warnings and error messages
5691
5692 \titleemph
5693
5694 \def\titleemph#1{\textbf{#1}}
5695
5696 (End definition for \titleemph. This function is documented on page ??.)
```

32.1 Definitions

definiendum

```
5691 \keys_define:nn {stex / definiendum }{
5692   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5693   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5694   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5695   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5696 }
5697 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5698   \str_clear:N \l__stex_statements_definiendum_root_str
5699   \tl_clear:N \l__stex_statements_definiendum_post_tl
5700   \str_clear:N \l__stex_statements_definiendum_gfa_str
5701   \keys_set:nn { stex / definiendum }{ #1 }
5702 }
5703 \NewDocumentCommand \definiendum { O{} m m } {
5704   \__stex_statements_definiendum_args:n { #1 }
5705   \stex_get_symbol:n { #2 }
5706   \stex_ref_new_sym_target:n \l__stex_get_symbol_uri_str
5707   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5708     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5709     \tl_set:Nn \l_tmpa_tl { #3 }
5710   } {
5711     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5712     \tl_set:Nn \l_tmpa_tl {
5713       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5714     }
5715   }
5716 } {
5717   \tl_set:Nn \l_tmpa_tl { #3 }
5718 }
5719
5720 % TODO root
5721 \stex_html_backend:TF {
5722   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5723 } {
5724   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5725 }
5726 }
5727 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

5728
5729 \NewDocumentCommand \definame { 0{ } m } {
5730   \__stex_statements_definiendum_args:n { #1 }
5731   % TODO: root
5732   \stex_get_symbol:n { #2 }
5733   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5734   \str_set:Nx \l_tmpa_str {
5735     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5736   }
5737   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5738   \stex_html_backend:TF {
5739     \stex_if_do_html:T {
5740       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5741         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5742       }
5743     }
5744   } {
5745     \exp_args:Nnx \defemph@uri {
5746       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5747     } { \l_stex_get_symbol_uri_str }
5748   }
5749 }
5750 \stex_deactivate_macro:Nn \definame {definition~environments}
5751
5752 \NewDocumentCommand \Definame { 0{ } m } {
5753   \__stex_statements_definiendum_args:n { #1 }
5754   \stex_get_symbol:n { #2 }
5755   \str_set:Nx \l_tmpa_str {
5756     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5757   }
5758   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```



```

5759 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5760 \stex_html_backend:TF {
5761   \stex_if_do_html:T {
5762     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5763       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5764     }
5765   }
5766 } {
5767   \exp_args:Nnx \defemph@uri {
5768     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5769   } { \l_stex_get_symbol_uri_str }
5770 }
5771 }
5772 \stex_deactivate_macro:Nn \Definame {definition-environments}
5773
5774 \NewDocumentCommand \premise { m }{
5775   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5776 }
5777 \NewDocumentCommand \conclusion { m }{
5778   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5779 }
5780 \NewDocumentCommand \definiens { 0{} m }{
5781   \str_clear:N \l_stex_get_symbol_uri_str
5782   \tl_if_empty:nF {#1} {
5783     \stex_get_symbol:n { #1 }
5784   }
5785   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5786     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5787       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5788     }{
5789       % TODO throw error
5790     }
5791   }
5792   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5793   {\l_stex_current_module_str}{
5794     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5795   }{true}{
5796     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5797     \exp_args:Nx \stex_add_to_current_module:n {
5798       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5799     }
5800   }
5801 }
5802 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5803 }
5804
5805 \NewDocumentCommand \varbindforall {m}{
5806   \stex_symbol_or_var:n {#1}
5807   \bool_if:NTF\l_stex_symbol_or_var_bool{
5808     \stex_if_do_html:T {
5809       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5810     }
5811   }{
5812     % todo throw error

```

```

5813 }
5814 }
5815
5816 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5817 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5818 \stex_deactivate_macro:Nn \definiens {definition~environments}
5819 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5820

```

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

sdefinition (env.)

```

5821
5822 \keys_define:nn {stex / sdefinition }{
5823   type      .str_set_x:N = \sdefinitiontype,
5824   id        .str_set_x:N = \sdefinitionid,
5825   name      .str_set_x:N = \sdefinitionname,
5826   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5827   title     .tl_set:N    = \sdefinitiontitle
5828 }
5829 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5830   \str_clear:N \sdefinitiontype
5831   \str_clear:N \sdefinitionid
5832   \str_clear:N \sdefinitionname
5833   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5834   \tl_clear:N \sdefinitiontitle
5835   \keys_set:nn { stex / sdefinition }{ #1 }
5836 }
5837
5838 \NewDocumentEnvironment{sdefinition}{0{}}{
5839   \__stex_statements_sdefinition_args:n{ #1 }
5840   \stex_reactivate_macro:N \definiendum
5841   \stex_reactivate_macro:N \definame
5842   \stex_reactivate_macro:N \Definame
5843   \stex_reactivate_macro:N \premise
5844   \stex_reactivate_macro:N \definiens
5845   \stex_reactivate_macro:N \varbindforall
5846   \stex_if_smsmode:F{
5847     \seq_clear:N \l_tmpb_seq
5848     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5849       \tl_if_empty:nF{ ##1 }{
5850         \stex_get_symbol:n { ##1 }
5851         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5852           \l_stex_get_symbol_uri_str
5853         }
5854       }
5855     }
5856     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5857     \exp_args:Nnnx
5858     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5859     \str_if_empty:NF \sdefinitiontype {
5860       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5861     }
5862     \str_if_empty:NF \sdefinitionname {

```

```

5863     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5864   }
5865   \clist_set:No \l_tmpa_clist \sdefinitiontype
5866   \tl_clear:N \l_tmpa_tl
5867   \clist_map_inline:Nn \l_tmpa_clist {
5868     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5869       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5870     }
5871   }
5872   \tl_if_empty:NTF \l_tmpa_tl {
5873     \__stex_statements_sdefinition_start:
5874   }{
5875     \l_tmpa_tl
5876   }
5877 }
5878 \stex_ref_new_doc_target:n \sdefinitionid
5879 \stex_smsmode_do:
5880 }{
5881   \stex_suppress_html:n {
5882     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5883   }
5884   \stex_if_smsmode:F {
5885     \clist_set:No \l_tmpa_clist \sdefinitiontype
5886     \tl_clear:N \l_tmpa_tl
5887     \clist_map_inline:Nn \l_tmpa_clist {
5888       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5889         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5890       }
5891     }
5892     \tl_if_empty:NTF \l_tmpa_tl {
5893       \__stex_statements_sdefinition_end:
5894     }{
5895       \l_tmpa_tl
5896     }
5897     \end{stex_annotate_env}
5898   }
5899 }

```

\stexpatchdefinition

```

5900 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5901   \stex_par:\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5902     ~(\sdefinitiontitle)
5903   }~}
5904 }
5905 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5906
5907 \newcommand\stexpatchdefinition[3] [] {
5908   \str_set:Nx \l_tmpa_str{ #1 }
5909   \str_if_empty:NTF \l_tmpa_str {
5910     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5911     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5912   }{
5913     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5914     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5915     }
5916 }

```

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

`\inlinedef` inline:

```

5917 \keys_define:nn {stex / inlinedef }{
5918   type      .str_set_x:N = \sdefinitiontype,
5919   id        .str_set_x:N = \sdefinitionid,
5920   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5921   name      .str_set_x:N = \sdefinitionname
5922 }
5923 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5924   \str_clear:N \sdefinitiontype
5925   \str_clear:N \sdefinitionid
5926   \str_clear:N \sdefinitionname
5927   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5928   \keys_set:nn { stex / inlinedef }{ #1 }
5929 }
5930 \NewDocumentCommand \inlinedef { 0{} m } {
5931   \begingroup
5932   \__stex_statements_inlinedef_args:n{ #1 }
5933   \stex_reactivate_macro:N \definiendum
5934   \stex_reactivate_macro:N \definame
5935   \stex_reactivate_macro:N \Definame
5936   \stex_reactivate_macro:N \premise
5937   \stex_reactivate_macro:N \definiens
5938   \stex_reactivate_macro:N \varbindforall
5939   \stex_ref_new_doc_target:n \sdefinitionid
5940   \stex_if_smsmode:TF{\stex_suppress_html:n {
5941     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5942   }}{
5943     \seq_clear:N \l_tmpb_seq
5944     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5945       \tl_if_empty:nF{ ##1 }{
5946         \stex_get_symbol:n { ##1 }
5947         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5948           \l_stex_get_symbol_uri_str
5949         }
5950       }
5951     }
5952     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5953     \ifvmode\noindent\fi
5954     \exp_args:Nnx
5955     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5956       \str_if_empty:NF \sdefinitiontype {
5957         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5958       }
5959       #2
5960       \str_if_empty:NF \sdefinitionname {
5961         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5962         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5963       }
5964     }

```

```

5965 }
5966 \endgroup
5967 \stex_smsmode_do:
5968 }

```

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

32.2 Assertions

sassertion (*env.*)

```

5970 \keys_define:nn {stex / sassertion }{
5971   type      .str_set_x:N = \sassertiontype,
5972   id        .str_set_x:N = \sassertionid,
5973   title     .tl_set:N     = \sassertiontitle ,
5974   for       .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5975   name      .str_set_x:N  = \sassertionname
5976 }
5977 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5978   \str_clear:N \sassertiontype
5979   \str_clear:N \sassertionid
5980   \str_clear:N \sassertionname
5981   \clist_clear:N \l__stex_statements_sassertion_for_clist
5982   \tl_clear:N \sassertiontitle
5983   \keys_set:nn { stex / sassertion }{ #1 }
5984 }
5985
5986 %\tl_new:N \g__stex_statements_aftergroup_tl
5987
5988 \NewDocumentEnvironment{sassertion}{0}{}{
5989   \__stex_statements_sassertion_args:n{ #1 }
5990   \stex_reactivate_macro:N \premise
5991   \stex_reactivate_macro:N \conclusion
5992   \stex_reactivate_macro:N \varbindforall
5993   \stex_if_smsmode:F {
5994     \seq_clear:N \l_tmpb_seq
5995     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5996       \tl_if_empty:NF{ ##1 }{
5997         \stex_get_symbol:n { ##1 }
5998         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5999           \l_stex_get_symbol_uri_str
6000         }
6001       }
6002     }
6003     \exp_args:Nnnx
6004     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {},}}
6005     \str_if_empty:NF \sassertiontype {
6006       \stex_annotate_invisible:nnn{type}{\sassertiontype}{-}
6007     }
6008     \str_if_empty:NF \sassertionname {
6009       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{-}
6010     }
6011     \clist_set:Nn \l_tmpa_clist \sassertiontype

```

```

6012 \tl_clear:N \l_tmpa_tl
6013 \clist_map_inline:Nn \l_tmpa_clist {
6014   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
6015     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
6016   }
6017 }
6018 \tl_if_empty:NTF \l_tmpa_tl {
6019   \__stex_statements_sassertion_start:
6020 }{
6021   \l_tmpa_tl
6022 }
6023 }
6024 \str_if_empty:NTF \sassertionid {
6025   \str_if_empty:NF \sassertionname {
6026     \stex_ref_new_doc_target:n {}
6027   }
6028 } {
6029   \stex_ref_new_doc_target:n \sassertionid
6030 }
6031 \stex_smsmode_do:
6032 ){
6033   \str_if_empty:NF \sassertionname {
6034     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassertionname}}
6035     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6036   }
6037   \stex_if_smsmode:F {
6038     \clist_set:Nn \l_tmpa_clist \sassertiontype
6039     \tl_clear:N \l_tmpa_tl
6040     \clist_map_inline:Nn \l_tmpa_clist {
6041       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
6042         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
6043       }
6044     }
6045     \tl_if_empty:NTF \l_tmpa_tl {
6046       \__stex_statements_sassertion_end:
6047     }{
6048       \l_tmpa_tl
6049     }
6050     \end{stex_annotate_env}
6051   }
6052 }

```

\stexpatchassertion

```

6053
6054 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
6055   \stex_par:\noindent\titileemph{Assertion~\tl_if_empty:NF \sassertiontitle {
6056     (\sassertiontitle)
6057   }~}
6058 }
6059 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
6060
6061 \newcommand\stexpatchassertion[3] [] {
6062   \str_set:Nx \l_tmpa_str{ #1 }
6063   \str_if_empty:NTF \l_tmpa_str {

```

```

6064     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
6065     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
6066   }{
6067     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
6068     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
6069   }
6070 }

```

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

`\inlineass` inline:

```

6071 \keys_define:nn {stex / inlineass }{
6072   type      .str_set_x:N = \sassertiontype,
6073   id        .str_set_x:N = \sassertionid,
6074   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
6075   name      .str_set_x:N = \sassertionname
6076 }
6077 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
6078   \str_clear:N \sassertiontype
6079   \str_clear:N \sassertionid
6080   \str_clear:N \sassertionname
6081   \clist_clear:N \l__stex_statements_sassertion_for_clist
6082   \keys_set:nn { stex / inlineass }{ #1 }
6083 }
6084 \NewDocumentCommand \inlineass { 0{} m } {
6085   \begin{group}
6086     \stex_reactivate_macro:N \premise
6087     \stex_reactivate_macro:N \conclusion
6088     \stex_reactivate_macro:N \varbindforall
6089     \__stex_statements_inlineass_args:n{ #1 }
6090     \str_if_empty:NTF \sassertionid {
6091       \str_if_empty:NF \sassertionname {
6092         \stex_ref_new_doc_target:n {}
6093       }
6094     } {
6095       \stex_ref_new_doc_target:n \sassertionid
6096     }
6097
6098     \stex_if_smsmode:TF{
6099       \str_if_empty:NF \sassertionname {
6100         \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
6101       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6102     }
6103   }{
6104     \seq_clear:N \l_tmpb_seq
6105     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6106       \tl_if_empty:nF{ ##1 }{
6107         \stex_get_symbol:n { ##1 }
6108         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6109           \l_stex_get_symbol_uri_str
6110         }
6111       }
6112     }
6113     \ifvmode\noindent\fi

```

```

6114 \exp_args:Nnx
6115 \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{
6116   \str_if_empty:NF \sassertiontype {
6117     \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
6118   }
6119   #2
6120   \str_if_empty:NF \sassertionname {
6121     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6122     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6123     \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
6124   }
6125 }
6126 }
6127 \endgroup
6128 \stex_smsmode_do:
6129 }

```

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

32.3 Examples

`sexample (env.)`

```

6130
6131 \keys_define:nn {stex / sexample }{
6132   type      .str_set_x:N = \exampletype,
6133   id        .str_set_x:N = \sexampleid,
6134   title     .tl_set:N     = \sexampletitle,
6135   name      .str_set_x:N = \sexamplename ,
6136   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
6137 }
6138 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
6139   \str_clear:N \sexampletype
6140   \str_clear:N \sexampleid
6141   \str_clear:N \sexamplename
6142   \tl_clear:N \sexampletitle
6143   \clist_clear:N \l__stex_statements_sexample_for_clist
6144   \keys_set:nn { stex / sexample }{ #1 }
6145 }
6146
6147 \NewDocumentEnvironment{sexample}{0{}}{
6148   \__stex_statements_sexample_args:n{ #1 }
6149   \stex_reactivate_macro:N \premise
6150   \stex_reactivate_macro:N \conclusion
6151   \stex_if_smsmode:F {
6152     \seq_clear:N \l_tmpb_seq
6153     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6154       \tl_if_empty:nF{ ##1 }{
6155         \stex_get_symbol:n { ##1 }
6156         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6157           \l_stex_get_symbol_uri_str
6158         }
6159       }
6160     }

```



```

6161 \exp_args:Nnnx
6162 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
6163 \str_if_empty:NF \sexamplotype {
6164   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{\}
6165 }
6166 \str_if_empty:NF \sexamplename {
6167   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{\}
6168 }
6169 \clist_set:No \l_tmpa_clist \sexamplotype
6170 \tl_clear:N \l_tmpa_tl
6171 \clist_map_inline:Nn \l_tmpa_clist {
6172   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6173     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
6174   }
6175 }
6176 \tl_if_empty:NTF \l_tmpa_tl {
6177   \__stex_statements_sexample_start:
6178 }{
6179   \l_tmpa_tl
6180 }
6181 }
6182 \str_if_empty:NF \sexampleid {
6183   \stex_ref_new_doc_target:n \sexampleid
6184 }
6185 \stex_smsmode_do:
6186 }{
6187   \str_if_empty:NF \sexamplename {
6188     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6189   }
6190   \stex_if_smsmode:F {
6191     \clist_set:No \l_tmpa_clist \sexamplotype
6192     \tl_clear:N \l_tmpa_tl
6193     \clist_map_inline:Nn \l_tmpa_clist {
6194       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6195         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6196       }
6197     }
6198     \tl_if_empty:NTF \l_tmpa_tl {
6199       \__stex_statements_sexample_end:
6200     }{
6201       \l_tmpa_tl
6202     }
6203     \end{stex_annotate_env}
6204   }
6205 }

```

\stexpatchexample

```

6206
6207 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6208   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
6209     (\sexampltitle)
6210   }~}
6211 }
6212 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}

```

```

6213
6214 \newcommand\stexpatchexample[3][] {
6215   \str_set:Nx \l_tmpa_str{ #1 }
6216   \str_if_empty:NTF \l_tmpa_str {
6217     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6218     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6219   }{
6220     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6221     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6222   }
6223 }

```

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

\inlineex inline:

```

6224 \keys_define:nn {stex / inlineex }{
6225   type      .str_set_x:N = \sexamplotype,
6226   id        .str_set_x:N = \sexampleid,
6227   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
6228   name      .str_set_x:N = \sexamplename
6229 }
6230 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6231   \str_clear:N \sexamplotype
6232   \str_clear:N \sexampleid
6233   \str_clear:N \sexamplename
6234   \clist_clear:N \l__stex_statements_sexample_for_clist
6235   \keys_set:nn { stex / inlineex }{ #1 }
6236 }
6237 \NewDocumentCommand \inlineex { 0{ } m } {
6238   \begingroup
6239   \stex_reactivate_macro:N \premise
6240   \stex_reactivate_macro:N \conclusion
6241   \__stex_statements_inlineex_args:n{ #1 }
6242   \str_if_empty:NF \sexampleid {
6243     \stex_ref_new_doc_target:n \sexampleid
6244   }
6245   \stex_if_smsmode:TF{
6246     \str_if_empty:NF \sexamplename {
6247       \stex_suppress_html:n{\stex_symdecl_do:nn{ }\sexamplename}}
6248     }
6249   }{
6250     \seq_clear:N \l_tmpb_seq
6251     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6252       \tl_if_empty:NF{ ##1 }{
6253         \stex_get_symbol:n { ##1 }
6254         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6255           \l_stex_get_symbol_uri_str
6256         }
6257       }
6258     }
6259     \ifvmode\noindent\fi
6260     \exp_args:Nnx
6261     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6262       \str_if_empty:NF \sexamplotype {

```



```

6310 \seq_clear:N \l_tmpb_seq
6311 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6312   \tl_if_empty:nF{ ##1 }{
6313     \stex_get_symbol:n { ##1 }
6314     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6315       \l_stex_get_symbol_uri_str
6316     }
6317   }
6318 }
6319 \exp_args:Nnnx
6320 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6321 \str_if_empty:NF \sparagraphtype {
6322   \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6323 }
6324 \str_if_empty:NF \sparagraphfrom {
6325   \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6326 }
6327 \str_if_empty:NF \sparagraphto {
6328   \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6329 }
6330 \str_if_empty:NF \sparagraphname {
6331   \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6332 }
6333 \clist_set:No \l_tmpa_clist \sparagraphtype
6334 \tl_clear:N \l_tmpa_tl
6335 \clist_map_inline:Nn \sparagraphtype {
6336   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6337     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
6338   }
6339 }
6340 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6341 \tl_if_empty:NTF \l_tmpa_tl {
6342   \__stex_statements_sparagraph_start:
6343 }{
6344   \l_tmpa_tl
6345 }
6346 }
6347 \clist_set:No \l_tmpa_clist \sparagraphtype
6348 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6349 {
6350   \stex_reactivate_macro:N \definiendum
6351   \stex_reactivate_macro:N \definame
6352   \stex_reactivate_macro:N \Definame
6353   \stex_reactivate_macro:N \premise
6354   \stex_reactivate_macro:N \definiens
6355 }
6356 \str_if_empty:NTF \sparagraphid {
6357   \str_if_empty:NTF \sparagraphname {
6358     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6359       \stex_ref_new_doc_target:n {}
6360     }
6361   } {
6362     \stex_ref_new_doc_target:n {}
6363   }

```

```

6364 } {
6365   \stex_ref_new_doc_target:n \sparagraphid
6366 }
6367 \exp_args:NNx
6368 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6369   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6370     \tl_if_empty:nF{ ##1 }{
6371       \stex_get_symbol:n { ##1 }
6372       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6373     }
6374   }
6375 }
6376 \stex_smsmode_do:
6377 \ignorespacesandpars
6378 }{
6379   \str_if_empty:NF \sparagraphname {
6380     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6381     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6382   }
6383   \stex_if_smsmode:F {
6384     \clist_set:No \l_tmpa_clist \sparagraphtype
6385     \tl_clear:N \l_tmpa_tl
6386     \clist_map_inline:Nn \l_tmpa_clist {
6387       \tl_if_exist:cT {\__stex_statements_sparagraph_##1_end:}{
6388         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sparagraph_##1_end:}}
6389       }
6390     }
6391     \tl_if_empty:NTF \l_tmpa_tl {
6392       \__stex_statements_sparagraph_end:
6393     }{
6394       \l_tmpa_tl
6395     }
6396     \end{stex_annotate_env}
6397   }
6398 }

```

\stexpatchparagraph

```

6399 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6400   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6401     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6402       \titleemph{\l_stex_sparagraph_title_tl}:~
6403     }
6404   }{
6405     \titleemph{\l_stex_sparagraph_start_tl}~
6406   }
6407 }
6408 }
6409 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6410
6411 \newcommand\stexpatchparagraph[3]{} {
6412   \str_set:Nx \l_tmpa_str{ #1 }
6413   \str_if_empty:NTF \l_tmpa_str {
6414     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6415     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }

```

```

6416     }{
6417         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
6418         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
6419     }
6420 }
6421
6422 \keys_define:nn { stex / inlinepara } {
6423     id      .str_set_x:N = \sparagraphid ,
6424     type    .str_set_x:N = \sparagraphtype ,
6425     for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6426     from    .tl_set:N     = \sparagraphfrom ,
6427     to      .tl_set:N     = \sparagraphto ,
6428     name    .str_set:N    = \sparagraphname
6429 }
6430 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6431     \tl_clear:N \sparagraphfrom
6432     \tl_clear:N \sparagraphto
6433     \str_clear:N \sparagraphid
6434     \str_clear:N \sparagraphtype
6435     \clist_clear:N \l__stex_statements_sparagraph_for_clist
6436     \str_clear:N \sparagraphname
6437     \keys_set:nn { stex / inlinepara }{ #1 }
6438 }
6439 \NewDocumentCommand \inlinepara { 0{} m } {
6440     \begingroup
6441     \__stex_statements_inlinepara_args:n{ #1 }
6442     \clist_set:Nn \l_tmpa_clist \sparagraphtype
6443     \str_if_empty:NTF \sparagraphid {
6444         \str_if_empty:NTF \sparagraphname {
6445             \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6446                 \stex_ref_new_doc_target:n {}
6447             }
6448         } {
6449             \stex_ref_new_doc_target:n {}
6450         }
6451     } {
6452         \stex_ref_new_doc_target:n \sparagraphid
6453     }
6454     \stex_if_smsmode:TF{
6455         \str_if_empty:NF \sparagraphname {
6456             \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6457             \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6458         }
6459     }{
6460         \seq_clear:N \l_tmpb_seq
6461         \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6462             \tl_if_empty:nF{ ##1 }{
6463                 \stex_get_symbol:n { ##1 }
6464                 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6465                     \l_stex_get_symbol_uri_str
6466                 }
6467             }
6468         }
6469         \ifvmode\noindent\fi

```

```

6470 \exp_args:Nnx
6471 \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6472   \str_if_empty:NF \sparagraphtype {
6473     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6474   }
6475   \str_if_empty:NF \sparagraphfrom {
6476     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6477   }
6478   \str_if_empty:NF \sparagraphto {
6479     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6480   }
6481   \str_if_empty:NF \sparagraphname {
6482     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6483     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6484     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6485   }
6486   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6487     \clist_map_inline:Nn \l_tmpb_seq {
6488       \stex_ref_new_sym_target:n {##1}
6489     }
6490   }
6491   #2
6492 }
6493 }
6494 \endgroup
6495 \stex_smsmode_do:
6496 }
6497

```

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

```

6498 </package>

```

Chapter 33

The Implementation

```
6499 <*package>
6500 <@@=stex_sproof>
6501
6502 %%%%%%%%%% sproof.dtx %%%%%%%%%%
6503
```

33.1 Proofs

We first define some keys for the proof environment.

```
6504 \keys_define:nn { stex / spf } {
6505   id          .str_set_x:N = \spfid,
6506   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
6507   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
6508   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
6509   type        .str_set_x:N = \spftype,
6510   title       .tl_set:N    = \spftitle,
6511   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
6512   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
6513   term        .tl_set:N    = \l__stex_sproof_spf_term_tl,
6514   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
6515   hide        .bool_set:N  = \l__stex_sproof_spf_hide_bool
6516 }
6517 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
6518   \str_clear:N \spfid
6519   \tl_clear:N \l__stex_sproof_spf_for_tl
6520   \tl_clear:N \l__stex_sproof_spf_from_tl
6521   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6522   \str_clear:N \spftype
6523   \tl_clear:N \spftitle
6524   \tl_clear:N \l__stex_sproof_spf_continues_tl
6525   \tl_clear:N \l__stex_sproof_spf_term_tl
6526   \tl_clear:N \l__stex_sproof_spf_functions_tl
6527   \tl_clear:N \l__stex_sproof_spf_method_tl
6528   \bool_set_false:N \l__stex_sproof_spf_hide_bool
6529   \keys_set:nn { stex / spf }{ #1 }
6530 }
6531 \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`

```
6532 \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.

```
6533 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6534 \cs_new_protected:Npn \sproofnumber {
6535   \int_set:Nn \l_tmpa_int {1}
6536   \bool_while_do:nn {
6537     \int_compare_p:nNn {
6538       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6539     } > 0
6540   }{
6541     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6542     \int_incr:N \l_tmpa_int
6543   }
6544 }
6545 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
6546   \int_set:Nn \l_tmpa_int {1}
6547   \bool_while_do:nn {
6548     \int_compare_p:nNn {
6549       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6550     } > 0
6551   }{
6552     \int_incr:N \l_tmpa_int
6553   }
6554   \int_compare:nNnF \l_tmpa_int = 1 {
6555     \int_decr:N \l_tmpa_int
6556   }
6557   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6558     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6559   }
6560 }
6561
6562 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6563   \int_set:Nn \l_tmpa_int {1}
6564   \bool_while_do:nn {
6565     \int_compare_p:nNn {
6566       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6567     } > 0
6568   }{
6569     \int_incr:N \l_tmpa_int
6570   }
6571   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6572 }
6573
```

```

6574 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6575   \int_set:Nn \l_tmpa_int {1}
6576   \bool_while_do:nn {
6577     \int_compare_p:nNn {
6578       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6579     } > 0
6580   }{
6581     \int_incr:N \l_tmpa_int
6582   }
6583   \int_decr:N \l_tmpa_int
6584   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6585 }

```

\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

```

6586 \def\sproof@box{
6587   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6588 }
6589 \def\sproofend{
6590   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6591     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6592   }
6593 }

```

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

spf@*kw

```

6594 \def\spf@proofsketch@kw{Proof~Sketch}
6595 \def\spf@proof@kw{Proof}
6596 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6597 \AddToHook{begindocument}{
6598   \ltx@ifpackageloaded{babel}{
6599     \makeatletter
6600     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6601     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6602       \input{sproof-ngerman.ldf}
6603     }
6604     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6605       \input{sproof-finnish.ldf}
6606     }
6607     \clist_if_in:NnT \l_tmpa_clist {french}{
6608       \input{sproof-french.ldf}
6609     }
6610     \clist_if_in:NnT \l_tmpa_clist {russian}{
6611       \input{sproof-russian.ldf}
6612     }
6613     \makeatother
6614   }{}
6615 }

```

spfsketch

```

6616 \newcommand\spfsketch[2] [] {
6617   \begin{group}
6618   \let \premise \stex_proof_premise:
6619   \_stex_sproof_spf_args:n{#1}
6620   \stex_if_smsmode:TF {
6621     \str_if_empty:NF \spfid {
6622       \stex_ref_new_doc_target:n \spfid
6623     }
6624   }{
6625     \seq_clear:N \l_tmpa_seq
6626     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6627       \tl_if_empty:nF{ ##1 }{
6628         \stex_get_symbol:n { ##1 }
6629         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6630           \l_stex_get_symbol_uri_str
6631         }
6632       }
6633     }
6634     \exp_args:Nnx
6635     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {},,}{
6636       \str_if_empty:NF \spftype {
6637         \stex_annotate_invisible:nnn{type}{\spftype}{
6638         }
6639       }
6640       \clist_set:Nn \l_tmpa_clist \spftype
6641       \tl_set:Nn \l_tmpa_tl {
6642         \titleemph{
6643           \tl_if_empty:NTF \spftitle {
6644             \spf@proofsketch@kw
6645           }{
6646             \spftitle
6647           }
6648         }
6649       }
6650       \clist_map_inline:Nn \l_tmpa_clist {
6651         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6652           \tl_clear:N \l_tmpa_tl
6653         }
6654       }
6655       \str_if_empty:NF \spfid {
6656         \stex_ref_new_doc_target:n \spfid
6657       }
6658       \l_tmpa_tl #2 \sproofend
6659     }
6660   }
6661   \endgroup
6662   \stex_smsmode_do:
6663 }

```

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

```

\_stex_sproof_maybe_comment:
\_stex_sproof_maybe_comment_end:
\_stex_sproof_start_comment:
6664 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6665
6666 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6667   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6668     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6669   }
6670 }
6671 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6672   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6673 }
6674 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6675   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6676 }
6677

```

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

\stexcommentfont

```

6678 \cs_new_protected:Npn \stexcommentfont {
6679   \small\itshape
6680 }

```

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

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

```

6681 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6682   \seq_clear:N \l_tmpa_seq
6683   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6684     \tl_if_empty:NF{ ##1 }{
6685       \stex_get_symbol:n { ##1 }
6686       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6687         \l_stex_get_symbol_uri_str
6688       }
6689     }
6690   }
6691   \exp_args:Nnnx
6692   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6693   \str_if_empty:NF \spftype {
6694     \stex_annotate_invisible:nnn{type}{\spftype}{}
6695   }
6696   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6697   \str_if_empty:NF \spfid {
6698     \stex_ref_new_doc_target:n \spfid
6699   }
6700   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6701   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6702     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6703   }
6704   \begin{list}{}{
6705     \setlength\topsep{0pt}
6706     \setlength\parsep{0pt}
6707     \setlength\rightmargin{0pt}

```

```

6708 } \__stex_sproof_maybe_comment:
6709 }
6710 \cs_new_protected:Nn \__stex_sproof_end_env:n {
6711   \stex_if_smsmode:F{
6712     \__stex_sproof_maybe_comment_end:
6713     \end{list}
6714     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6715       \stex_html_backend:F{\egroup}
6716     }
6717     \clist_set:No \l_tmpa_clist \spftype
6718     #1
6719     \end{stex_annotate_env}
6720     \end{stex_annotate_env}
6721   }
6722 }
6723 }
6724 \NewDocumentEnvironment{sproof}{s O{} m}{
6725   \intarray_gzero:N \l__stex_sproof_counter_intarray
6726   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6727   \stex_reactivate_macro:N \yield
6728   \stex_reactivate_macro:N \eqstep
6729   \stex_reactivate_macro:N \assumption
6730   \stex_reactivate_macro:N \conclude
6731   \stex_reactivate_macro:N \spfstep
6732   \__stex_sproof_spf_args:n{#2}
6733   \stex_if_smsmode:TF {
6734     \str_if_empty:NF \spfid {
6735       \stex_ref_new_doc_target:n \spfid
6736     }
6737   }{
6738     \__stex_sproof_start_env:nnn{sproof}{#3}{
6739       \clist_set:No \l_tmpa_clist \spftype
6740       \tl_clear:N \l_tmpa_tl
6741       \clist_map_inline:Nn \l_tmpa_clist {
6742         \tl_if_exist:cT {\__stex_sproof_sproof_##1_start:}{
6743           \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_start:}}
6744         }
6745         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6746           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6747         }
6748       }
6749       \tl_if_empty:NTF \l_tmpa_tl {
6750         \__stex_sproof_sproof_start:
6751       }{
6752         \l_tmpa_tl
6753       }
6754     }
6755   }
6756   \stex_smsmode_do:
6757 }{\__stex_sproof_end_env:n{
6758   \tl_clear:N \l_tmpa_tl
6759   \clist_map_inline:Nn \l_tmpa_clist {
6760     \tl_if_exist:cT {\__stex_sproof_sproof_##1_end:}{
6761       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_end:}}

```

```

6762     }
6763   }
6764   \tl_if_empty:NTF \l_tmpa_tl {
6765     \__stex_sproof_sproof_end:
6766   }{
6767     \l_tmpa_tl
6768   }
6769 }}
6770 \NewDocumentEnvironment{subproof}{s O{} m}{
6771   \__stex_sproof_spf_args:n{#2}
6772   \stex_if_smsmode:TF {
6773     \str_if_empty:NF \spfid {
6774       \stex_ref_new_doc_target:n \spfid
6775     }
6776   }{
6777     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6778   }
6779   \__stex_sproof_add_counter:
6780   \stex_smsmode_do:
6781 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6782   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6783     \__stex_sproof_inc_counter:
6784   }
6785   \aftergroup\__stex_sproof_maybe_comment:
6786 }
6787 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6788
6789 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6790   \par\noindent\titleemph{
6791     \tl_if_empty:NTF \spftype {
6792       \spf@proof@kw
6793     }{
6794       \spftype
6795     }
6796   }:
6797 }
6798 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6799
6800 \newcommand\stexpatchproof[3] [] {
6801   \str_set:Nx \l_tmpa_str{ #1 }
6802   \str_if_empty:NTF \l_tmpa_str {
6803     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6804     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6805   }{
6806     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6807     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6808   }
6809 }

\pstep
\conclude
\assumption
\have
\eqstep

```

```

6814 type .str_set_x:N = \spftype,
6815 title .tl_set:N = \spftitle,
6816 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6817 term .tl_set:N = \l__stex_sproof_spf_term_tl
6818 }
6819 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6820 \str_clear:N \spfstepid
6821 \clist_clear:N \l__stex_sproof_spf_for_clist
6822 \str_clear:N \spftype
6823 \tl_clear:N \l__stex_sproof_spf_method_tl
6824 \tl_clear:N \l__stex_sproof_spf_term_tl
6825 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6826 \keys_set:nn { stex / spfsteps }{ #1 }
6827 }
6828
6829 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6830 \NewDocumentCommand #1 {s O{} +m} {
6831 \__stex_sproof_maybe_comment_end:
6832
6833 \__stex_sproof_spfstep_args:n{##2}
6834 \stex_annotate:nnn{spfstep}{#2}{
6835 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6836 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6837 }
6838 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6839 #4
6840 }{
6841 \item[\IfBooleanTF ##1 {}{#3}]
6842 }
6843 \ignorespacesandpars ##3
6844 }
6845 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6846 \__stex_sproof_maybe_comment:
6847 }
6848 \stex_deactivate_macro:Nn #1 {sproof~environments}
6849 }
6850
6851 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6852 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6853 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6854
6855 \NewDocumentCommand \eqstep {s m}{
6856 \__stex_sproof_maybe_comment_end:
6857 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6858 $=$
6859 }{
6860 \item[$=$]
6861 }
6862 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6863 \__stex_sproof_maybe_comment:
6864 }
6865 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6866
6867 \NewDocumentCommand \yield {+m}{

```

```

6868 \stex_annotate:nnn{spfyield}{\}{ #1 }
6869 }
6870 \stex_deactivate_macro:Nn \yield {sproof~environments}
6871
6872 \NewDocumentEnvironment{spfblock}{\}{
6873   \item[]
6874   \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6875 }{
6876   \aftergroup\__stex_sproof_maybe_comment:
6877 }
6878 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6879

```

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

`\spfidea`

```

6880 \NewDocumentCommand\spfidea{0{} +m}{
6881   \__stex_sproof_spf_args:n{#1}
6882   \titleemph{
6883     \tl_if_empty:NTF \spftype {Proof~Idea}{
6884       \spftype
6885     }:
6886   }~#2
6887   \sproofend
6888 }

```

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

```

6889 \newcommand\spfjust[1]{
6890   #1
6891 }
6892 \</package>

```

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

Chapter 34

STEX -Others Implementation

```
6893 <*package>
6894
6895 %%%%%%%%%% others.dtx %%%%%%%%%%
6896
6897 <@@=stex_others>
        Warnings and error messages
6898 % None

\MSC Math subject classifier

6899 \NewDocumentCommand \MSC {m} {
6900 % TODO
6901 }

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

6902 \@ifpackageloaded{tikzinput}{
6903 \RequirePackage{stex-tikzinput}
6904 }{}
6905
6906 \bool_if:NT \c_stex_persist_mode_bool {
6907 \let__stex_notation_restore_notation_old:nnnnn
6908 \__stex_notation_restore_notation:nnnnn
6909 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6910 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6911 \ExplSyntaxOn
6912 }
6913 \def__stex_notation_restore_notation:nnnnn{
6914 \ExplSyntaxOff
6915 \catcode'\sim10
6916 \__stex_notation_restore_notation_new:nnnnn
6917 }
6918 \input{\jobname.sms}
6919 \let__stex_notation_restore_notation:nnnnn
6920 \__stex_notation_restore_notation_old:nnnnn
6921 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```

6922     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6923     \l_tmpa_str
6924     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6925     \c_stex_mathhub_main_manifest_prop
6926     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6927   }
6928 }
6929
6930 \stex_get_document_uri:
6931 </package>

```

Chapter 35

STEX -Metatheory Implementation

```
6932 <*package>
6933 <@@=stex_modules>
6934
6935 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6936
6937 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6938 \begingroup
6939 \stex_module_setup:nn{
6940   ns=\c_stex_metatheory_ns_str,
6941   meta=NONE
6942 }{Metatheory}
6943 \stex_reactivate_macro:N \symdecl
6944 \stex_reactivate_macro:N \notation
6945 \stex_reactivate_macro:N \symdef
6946 \ExplSyntaxOff
6947 \csname stex_suppress_html:n\endcsname{
6948   % is-a (a:A, a \in A, a is an A, etc.)
6949   \symdecl{isa}[args=ai]
6950   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6951   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6952   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6953
6954   % bind (\forall, \Pi, \lambda etc.)
6955   \symdecl{bind}[args=Bi,assoc=pre]
6956   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6957   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6958   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6959
6960   % implicit bind
6961   \symdecl{implicitbind}[args=Bi,assoc=pre]
6962   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6963   \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6964   \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6965
6966   % dummy variable
```

```

6967 \symdecl{dummyvar}
6968 \notation{dummyvar}[underscore]{\comp\_}
6969 \notation{dummyvar}[dot]{\comp\cdot}
6970 \notation{dummyvar}[dash]{\comp{\rm --}}
6971
6972 %fromto (function space, Hom-set, implication etc.)
6973 \symdecl{fromto}[args=ai]
6974 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6975 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6976
6977 % mapto (lambda etc.)
6978 \symdecl{mapto}[args=Bi]
6979 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6980 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6981 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
6982
6983 % function/operator application
6984 \symdecl{apply}[args=ia]
6985 \notation{apply}[prec=0;0x\infp,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6986 \notation{apply}[prec=0;0x\infp,lambda]{#1 \; #2 }{##1 \; ; ##2}
6987
6988 % collection of propositions/booleans/truth values
6989 \symdecl{prop}[name=proposition]
6990 \notation{prop}[prop]{\comp{\rm prop}}
6991 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6992
6993 \symdecl{judgmentholds}[args=1]
6994 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6995
6996 % sequences
6997 \symdecl{seqtype}[args=1]
6998 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6999
7000 \symdecl{seqexpr}[args=a]
7001 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
7002
7003 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1
7004 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
7005 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
7006 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
7007 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
7008 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
7009 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7010 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
7011 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
7012
7013 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
7014 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{#2}}
7015
7016 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
7017 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
7018 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}{##1\comp,##2,##3}
7019
7020 % nat literals

```

```

7021 \symdef{natliteral}{\comp{\mathtt{Ord}}}
7022
7023 % letin (''let'', local definitions, variable substitution)
7024 \symdecl{letin}[args=bii]
7025 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
7026 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
7027 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
7028
7029 % structures
7030 \symdecl*{module-type}[args=1]
7031 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
7032 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
7033 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
7034
7035 % objects
7036 \symdecl{object}
7037 \notation{object}{\comp{\mathtt{OBJECT}}}
7038
7039 }
7040
7041 % The following are abbreviations in the sTeX corpus that are left over from earlier
7042 % developments. They will eventually be phased out.
7043
7044 \ExplSyntaxOn
7045 \stex_add_to_current_module:n{
7046   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
7047   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
7048   \def\livar{\csname sequence-index\endcsname[li]}
7049   \def\uivar{\csname sequence-index\endcsname[ui]}
7050   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
7051   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
7052 }
7053 \__stex_modules_end_module:
7054 \endgroup
7055 \</package>

```

Chapter 36

Tikzinput Implementation

```
7056 <@@=tikzinput>
7057 <*package>
7058
7059 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
7060
7061 \ProvidesExplPackage{tikzinput}{2022/08/08}{3.2.0}{tikzinput package}
7062 \RequirePackage{l3keys2e}
7063
7064 \keys_define:nn { tikzinput } {
7065   image .bool_set:N = \c_tikzinput_image_bool,
7066   image .default:n = false ,
7067   unknown .code:n = {}
7068 }
7069
7070 \ProcessKeysOptions { tikzinput }
7071
7072 \bool_if:NTF \c_tikzinput_image_bool {
7073   \RequirePackage{graphicx}
7074
7075   \providecommand\usetikzlibrary[]{}
7076   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
7077 }{
7078   \RequirePackage{tikz}
7079   \RequirePackage{standalone}
7080
7081   \newcommand \tikzinput [2] [] {
7082     \setkeys{Gin}{#1}
7083     \ifx \Gin@ewidth \Gin@exclamation
7084       \ifx \Gin@eheight \Gin@exclamation
7085         \input { #2 }
7086       \else
7087         \resizebox{!}{ \Gin@eheight }{
7088           \input { #2 }
7089         }
7090       \fi
7091     \else
7092       \ifx \Gin@eheight \Gin@exclamation
7093         \resizebox{ \Gin@ewidth }{!}{
```

```

7094         \input { #2 }
7095     }
7096     \else
7097         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
7098             \input { #2 }
7099         }
7100     \fi
7101 \fi
7102 }
7103 }
7104
7105 \newcommand \ctikzinput [2] [] {
7106     \begin{center}
7107         \tikzinput [#1] {#2}
7108     \end{center}
7109 }
7110
7111 \@ifpackageloaded{stex}{
7112     \RequirePackage{stex-tikzinput}
7113 }{}
7114
7115 </package>
7116 <*stex>
7117 \ProvidesExplPackage{stex-tikzinput}{2022/08/08}{3.2.0}{stex-tikzinput}
7118 \RequirePackage{stex}
7119 \RequirePackage{tikzinput}
7120
7121 \newcommand\mhtikzinput[2] []{%
7122     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
7123     \stex_in_repository:nn\Gin@mhrepos{
7124         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
7125     }
7126 }
7127 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
7128
7129 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
7130     \pgfkeys@spdef\pgf@temp{#1}
7131     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
7132     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
7133     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
7134     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
7135     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
7136     \catcode'\@=11
7137     \catcode'\|=12
7138     \catcode'\$=3
7139     \pgfutil@InputIfFileExists{#2}{-}{-}
7140     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
7141     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
7142     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
7143 }
7144
7145
7146 \newcommand\libusetikzlibrary[1]{

```

```

7147 \prop_if_exist:NF \l_stex_current_repository_prop {
7148   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7149 }
7150 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
7151   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7152 }
7153 \seq_clear:N \l__tikzinput_libinput_files_seq
7154 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
7155 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
7156
7157 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
7158   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
7159   \IfFileExists{ \l_tmpa_str }{
7160     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7161   }{}
7162   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
7163   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
7164 }
7165
7166 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
7167 \IfFileExists{ \l_tmpa_str }{
7168   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7169 }{}
7170
7171 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
7172   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
7173 }{
7174   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7175     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7176       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7177     }
7178   }{
7179     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7180   }
7181 }
7182 }
7183 </stex>

```


Chapter 37

document-structure.sty Implementation

```
7184 <*package>
7185 <@@=document_structure>
7186 \ProvidesExplPackage{document-structure}{2022/08/08}{3.2.0}{Modular Document Structure}
7187 \RequirePackage{13keys2e}
```

37.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option xxx will just set the appropriate switches to true (otherwise they stay false).

```
7188
7189 \keys_define:nn{ document-structure }{
7190   class      .str_set_x:N = \c_document_structure_class_str,
7191   topsect    .str_set_x:N = \c_document_structure_topsect_str,
7192   unknown    .code:n      = {
7193     \PassOptionsToClass{\CurrentOption}{stex}
7194     \PassOptionsToClass{\CurrentOption}{tikzinput}
7195   }
7196   % showignores .bool_set:N = \c_document_structure_showignores_bool,
7197 }
7198 \ProcessKeysOptions{ document-structure }
7199 \str_if_empty:NT \c_document_structure_class_str {
7200   \str_set:Nn \c_document_structure_class_str {article}
7201 }
7202 \str_if_empty:NT \c_document_structure_topsect_str {
7203   \str_set:Nn \c_document_structure_topsect_str {section}
7204 }
```

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

```
7205 \RequirePackage{xspace}
7206 \RequirePackage{comment}
7207 \RequirePackage{stex}
7208 \AddToHook{begindocument}{
```

```

7209 \ltx@ifpackageloaded{babel}{
7210     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7211     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7212         \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7213     }
7214 }{}
7215 }

```

`\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.

```

7216 \int_new:N \l_document_structure_section_level_int
7217 \str_case:NnF \c_document_structure_topsect_str {
7218     {part}}{
7219     \int_set:Nn \l_document_structure_section_level_int {0}
7220 }
7221 {chapter}}{
7222     \int_set:Nn \l_document_structure_section_level_int {1}
7223 }
7224 }{
7225     \str_case:NnF \c_document_structure_class_str {
7226         {book}}{
7227             \int_set:Nn \l_document_structure_section_level_int {0}
7228         }
7229         {report}}{
7230             \int_set:Nn \l_document_structure_section_level_int {0}
7231         }
7232     }{
7233         \int_set:Nn \l_document_structure_section_level_int {2}
7234     }
7235 }

```

37.2 Document Structure

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

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

EdN:9

```

7236 \def\current@section@level{document}%
7237 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
7238 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

7239 \cs_new_protected:Npn \skipfragment {

```

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

```

7240 \ifcase\l_document_structure_section_level_int
7241 \or\stepcounter{part}
7242 \or\stepcounter{chapter}
7243 \or\stepcounter{section}
7244 \or\stepcounter{subsection}
7245 \or\stepcounter{subsubsection}
7246 \or\stepcounter{paragraph}
7247 \or\stepcounter{subparagraph}
7248 \fi
7249 }

```

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

`blindfragment (env.)`

```

7250 \newcommand\at@begin@blindsfragment[1]{
7251 \newenvironment{blindfragment}
7252 {
7253 \int_incr:N\l_document_structure_section_level_int
7254 \at@begin@blindsfragment\l_document_structure_section_level_int
7255 }{}

```

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

```

7256 \newcommand\sfragment@nonum[2]{
7257 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7258 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7259 }

```

(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.

```

7260 \newcommand\sfragment@num[2]{
7261 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7262 \@nameuse{#1}{#2}
7263 }{
7264 \cs_if_exist:NTF\rdfmata@sectioning{
7265 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7266 }{
7267 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7268 }
7269 }
7270 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7271 }

```

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

`sfragment (env.)`

```

7272 \keys_define:nn { document-structure / sfragment }{
7273 id .str_set_x:N = \l__document_structure_sfragment_id_str,
7274 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

7275 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7276 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7277 srccite        .tl_set:N    = \l__document_structure_sfragment_srccite_tl,
7278 type          .tl_set:N    = \l__document_structure_sfragment_type_tl,
7279 short         .tl_set:N    = \l__document_structure_sfragment_short_tl,
7280 intro         .tl_set:N    = \l__document_structure_sfragment_intro_tl,
7281 imports       .tl_set:N    = \l__document_structure_sfragment_imports_tl,
7282 loadmodules    .bool_set:N  = \l__document_structure_sfragment_loadmodules_bool
7283 }
7284 \cs_new_protected:Nn \__document_structure_sfragment_args:n {
7285   \str_clear:N \l__document_structure_sfragment_id_str
7286   \str_clear:N \l__document_structure_sfragment_date_str
7287   \clist_clear:N \l__document_structure_sfragment_creators_clist
7288   \clist_clear:N \l__document_structure_sfragment_contributors_clist
7289   \tl_clear:N \l__document_structure_sfragment_srccite_tl
7290   \tl_clear:N \l__document_structure_sfragment_type_tl
7291   \tl_clear:N \l__document_structure_sfragment_short_tl
7292   \tl_clear:N \l__document_structure_sfragment_imports_tl
7293   \tl_clear:N \l__document_structure_sfragment_intro_tl
7294   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7295   \keys_set:nn { document-structure / sfragment } { #1 }
7296 }

```

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.

```

7297 \newif\if@mainmatter\@mainmattertrue
7298 \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.

```

7299 \keys_define:nn { document-structure / sectioning }{
7300   name      .str_set_x:N = \l__document_structure_sect_name_str ,
7301   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
7302   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
7303   clear     .default:n   = {true} ,
7304   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
7305   num       .default:n   = {true}
7306 }
7307 \cs_new_protected:Nn \__document_structure_sect_args:n {
7308   \str_clear:N \l__document_structure_sect_name_str
7309   \str_clear:N \l__document_structure_sect_ref_str
7310   \bool_set_false:N \l__document_structure_sect_clear_bool
7311   \bool_set_false:N \l__document_structure_sect_num_bool
7312   \keys_set:nn { document-structure / sectioning } { #1 }
7313 }
7314 \newcommand\omdoc@sectioning[3][]{
7315   \__document_structure_sect_args:n {#1 }
7316   \let\omdoc@sect@name\l__document_structure_sect_name_str
7317   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7318   \if@mainmatter% numbering not overridden by frontmatter, etc.
7319     \bool_if:NTF \l__document_structure_sect_num_bool {
7320       \sfragment@num{#2}{#3}
7321     }{

```

```

7322     \sfragment@nonum{#2}{#3}
7323   }
7324   \def\current@section@level{\omdoc@sect@name}
7325   \else
7326     \sfragment@nonum{#2}{#3}
7327   \fi
7328 }% 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.

```

7329 \newcommand\sfragment@redefine@addtocontents[1]{%
7330 %\edef\__document_structureimport{#1}%
7331 %\@for\@I:=\__document_structureimport\do{%
7332 %\edef\@path{\csname module@\@I @path\endcsname}%
7333 %\@ifundefined{tf@toc}\relax%
7334 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
7335 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7336 %\def\addcontentsline##1##2##3{%
7337 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7338 %\else% hyperref.sty not loaded
7339 %\def\addcontentsline##1##2##3{%
7340 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
7341 %\fi
7342 }% 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.

```

7343 \newenvironment{sfragment}[2][ ]% keys, title
7344 {
7345   \__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.

```

7346   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7347
7348   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7349     \sfragment@redefine@addtocontents{
7350       %\@ifundefined{module@id}\used@modules%
7351       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7352     }
7353   }

```

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

```

7354
7355   \stex_document_title:n { #2 }
7356
7357   \int_incr:N\l__document_structure_section_level_int
7358   \ifcase\l__document_structure_section_level_int
7359     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7360     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7361     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
7362     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

7363 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7364 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
7365 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
7366 \fi
7367 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
7368 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7369   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7370 }
7371 }% for customization
7372 {}

```

and finally, we localize the sections

```

7373 \newcommand\omdoc@part@kw{Part}
7374 \newcommand\omdoc@chapter@kw{Chapter}
7375 \newcommand\omdoc@section@kw{Section}
7376 \newcommand\omdoc@subsection@kw{Subsection}
7377 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7378 \newcommand\omdoc@paragraph@kw{paragraph}
7379 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

37.3 Front and Backmatter

Index markup is provided by the `omtext` package [Kohlhase:smmtf:git], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

7380 \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).

```

7381 \cs_if_exist:NTF\frontmatter{
7382   \let\__document_structure_orig_frontmatter\frontmatter
7383   \let\frontmatter\relax
7384 }{
7385   \tl_set:Nn\__document_structure_orig_frontmatter{
7386     \clearpage
7387     \@mainmatterfalse
7388     \pagenumbering{roman}
7389   }
7390 }
7391 \cs_if_exist:NTF\backmatter{
7392   \let\__document_structure_orig_backmatter\backmatter
7393   \let\backmatter\relax
7394 }{
7395   \tl_set:Nn\__document_structure_orig_backmatter{
7396     \clearpage
7397     \@mainmatterfalse
7398     \pagenumbering{roman}
7399   }

```

7400 }

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

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

```
7401 \newenvironment{frontmatter}{
7402   \__document_structure_orig_frontmatter
7403 }{
7404   \cs_if_exist:NTF\mainmatter{
7405     \mainmatter
7406   }{
7407     \clearpage
7408     \@mainmattertrue
7409     \pagenumbering{arabic}
7410   }
7411 }
```

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

```
7412 \newenvironment{backmatter}{
7413   \__document_structure_orig_backmatter
7414 }{
7415   \cs_if_exist:NTF\mainmatter{
7416     \mainmatter
7417   }{
7418     \clearpage
7419     \@mainmattertrue
7420     \pagenumbering{arabic}
7421   }
7422 }
```

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

```
7423 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7424 \def \c__document_structure_document_str{document}
7425 \newcommand\afterprematurestop{}
7426 \def\prematurestop@endsfragment{
7427   \unless\ifx\@currenvir\c__document_structure_document_str
7428     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7429     \expandafter\prematurestop@endsfragment
7430   \fi
7431 }
7432 \providecommand\prematurestop{
7433   \message{Stopping~sTeX~processing~prematurely}
7434   \prematurestop@endsfragment
7435   \afterprematurestop
7436   \end{document}
7437 }
```

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

37.4 Global Variables

\setSGvar set a global variable

```
7438 \RequirePackage{etoolbox}
7439 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
7440 \newrobustcmd\useSGvar[1]{%
7441   \@ifundefined{sTeX@Gvar@#1}
7442   {\PackageError{document-structure}
7443    {The sTeX Global variable #1 is undefined}
7444    {set it with \protect\setSGvar}}
7445   \@nameuse{sTeX@Gvar@#1}}
```

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

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

```
7446 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7447   \@ifundefined{sTeX@Gvar@#1}
7448   {\PackageError{document-structure}
7449    {The sTeX Global variable #1 is undefined}
7450    {set it with \protect\setSGvar}}
7451   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}
```

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

Chapter 38

NotesSlides – Implementation

38.1 Class and Package Options

We define some Package Options and switches for the `notesslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `notesslides` package. We pass the `nontheorem` option to the `statements` package when we are not in notes mode, since the `beamer` package has its own (overlay-aware) theorem environments.

```
7452 \*cls)
7453 \@@=notesslides)
7454 \ProvidesExplClass{notesslides}{2022/08/08}{3.2.0}{notesslides Class}
7455 \RequirePackage{13keys2e}
7456
7457 \keys_define:nn{notesslides / cls}{
7458   class .str_set_x:N = \c__notesslides_class_str,
7459   notes .bool_set:N = \c__notesslides_notes_bool ,
7460   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7461   docopt .str_set_x:N = \c__notesslides_docopt_str,
7462   unknown .code:n = {
7463     \PassOptionsToPackage{\CurrentOption}{document-structure}
7464     \PassOptionsToClass{\CurrentOption}{beamer}
7465     \PassOptionsToPackage{\CurrentOption}{notesslides}
7466     \PassOptionsToPackage{\CurrentOption}{stex}
7467   }
7468 }
7469 \ProcessKeysOptions{ notesslides / cls }
7470
7471 \str_if_empty:NF \c__notesslides_class_str {
7472   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7473 }
7474
7475 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7476   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7477 }
7478 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7479   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7480 }
7481
7482 \RequirePackage{stex}
```

```

7483 \stex_html_backend:T {
7484   \bool_set_true:N\c__notesslides_notes_bool
7485 }
7486
7487 \bool_if:NTF \c__notesslides_notes_bool {
7488   \PassOptionsToPackage{notes=true}{notesslides}
7489   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7490 }{
7491   \PassOptionsToPackage{notes=false}{notesslides}
7492   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7493 }
7494 \</cls>

```

now we do the same for the notesslides package.

```

7495 <*package>
7496 \ProvidesExplPackage{notesslides}{2022/08/08}{3.2.0}{notesslides Package}
7497 \RequirePackage{l3keys2e}
7498
7499 \keys_define:nn{notesslides / pkg}{
7500   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7501   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7502   notes            .bool_set:N = \c__notesslides_notes_bool ,
7503   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7504   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7505   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7506   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7507   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
7508   unknown          .code:n      = {
7509     \PassOptionsToClass{\CurrentOption}{stex}
7510     \PassOptionsToClass{\CurrentOption}{tikzinput}
7511   }
7512 }
7513 \ProcessKeysOptions{ notesslides / pkg }
7514
7515 \RequirePackage{stex}
7516 \stex_html_backend:T {
7517   \bool_set_true:N\c__notesslides_notes_bool
7518 }
7519
7520 \newif\ifnotes
7521 \bool_if:NTF \c__notesslides_notes_bool {
7522   \notesttrue
7523 }{
7524   \notestfalse
7525 }
7526

```

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

```

7527 \str_if_empty:NTF \c__notesslides_topsect_str {
7528   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7529 }{
7530   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7531 }
7532 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
7533 </package>
```

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

```
7534 <*cls>
7535 \bool_if:NTF \c__notesslides_notes_bool {
7536   \str_if_empty:NT \c__notesslides_class_str {
7537     \str_set:Nn \c__notesslides_class_str {article}
7538   }
7539   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
7540     {\c__notesslides_class_str}
7541 }{
7542   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7543   \newcounter{Item}
7544   \newcounter{paragraph}
7545   \newcounter{subparagraph}
7546   \newcounter{Hfootnote}
7547 }
7548 \RequirePackage{document-structure}
```

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

```
7549 \RequirePackage{notesslides}
7550 </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 \LaTeX packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the \LaTeX -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.

```
7551 <*package>
7552 \bool_if:NT \c__notesslides_notes_bool {
7553   \RequirePackage{a4wide}
7554   \RequirePackage{marginnote}
7555   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7556   \RequirePackage{mdframed}
7557   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7558   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7559 }
7560 \RequirePackage{stex-tikzinput}
7561 \RequirePackage{comment}
7562 \RequirePackage{url}
7563 \RequirePackage{graphicx}
7564 \RequirePackage{pgf}
7565 \RequirePackage{bookmark}
```

38.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class.

```
7566 \bool_if:NT \c__notesslides_notes_bool {
7567   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7568 }
```

```

7569 \NewDocumentCommand \libusetheme {0{} m} {
7570   \libusepackage[#1]{beamertheme#2}
7571 }
7572

```

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

```

7573 \newcounter{slide}
7574 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7575 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

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

```

7576 \bool_if:NTF \c__notesslides_notes_bool {
7577   \renewenvironment{note}{\ignorespaces}{}
7578 }{
7579   \excludecomment{note}
7580 }

```

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.

```

7581 \bool_if:NT \c__notesslides_notes_bool {
7582   \newlength{\slideframewidth}
7583   \setlength{\slideframewidth}{1.5pt}

```

frame (*env.*) We first define the keys.

```

7584 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7585   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7586     \bool_set_true:N #1
7587   }{
7588     \bool_set_false:N #1
7589   }
7590 }
7591 \keys_define:nn{notesslides / frame}{
7592   label .str_set_x:N = \l__notesslides_frame_label_str,
7593   allowframebreaks .code:n = {
7594     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7595   },
7596   allowdisplaybreaks .code:n = {
7597     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7598   },
7599   fragile .code:n = {
7600     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7601   },
7602   shrink .code:n = {
7603     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7604   },
7605   squeeze .code:n = {
7606     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7607   },
7608   t .code:n = {

```

```

7609     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7610   },
7611   unknown    .code:n      = {}
7612 }
7613 \cs_new_protected:Nn \__notesslides_frame_args:n {
7614   \str_clear:N \l__notesslides_frame_label_str
7615   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7616   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7617   \bool_set_true:N \l__notesslides_frame_fragile_bool
7618   \bool_set_true:N \l__notesslides_frame_shrink_bool
7619   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7620   \bool_set_true:N \l__notesslides_frame_t_bool
7621   \keys_set:nn { notesslides / frame }{ #1 }
7622 }

```

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

```

7623 \renewenvironment{frame}[1][]{
7624   \__notesslides_frame_args:n{#1}
7625   \sffamily
7626   \stepcounter{slide}
7627   \def\@currentlabel{\theslide}
7628   \str_if_empty:NF \l__notesslides_frame_label_str {
7629     \label{\l__notesslides_frame_label_str}
7630   }

```

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

```

7631   \def\itemize@level{outer}
7632   \def\itemize@outer{outer}
7633   \def\itemize@inner{inner}
7634   \renewcommand\newpage{\addtocounter{framenum}{1}}
7635   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7636   \renewenvironment{itemize}{
7637     \ifx\itemize@level\itemize@outer
7638       \def\itemize@label{$\rhd$}
7639     \fi
7640     \ifx\itemize@level\itemize@inner
7641       \def\itemize@label{$\scriptstyle\rhd$}
7642     \fi
7643     \begin{list}
7644       {\itemize@label}
7645       {\setlength{\labelsep}{.3em}
7646        \setlength{\labelwidth}{.5em}
7647        \setlength{\leftmargin}{1.5em}
7648       }
7649     \edef\itemize@level{\itemize@inner}
7650   }{
7651     \end{list}
7652   }

```

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

```

7653   \stex_html_backend:TF {
7654     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7655     \mdf@patchamsthm
7656   }{
7657     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7658     }
7659   }{
7660     \stex_html_backend:TF {
7661       \miko@slidelabel\egroup\end{stex_annotate_env}
7662     }\medskip\miko@slidelabel\end{mdframed}}
7663   }

```

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

`\frametitle`

```

7664   \renewcommand{\frametitle}[1]{
7665     \stex_document_title:n { #1 }
7666     {\Large\bf\sf\color{blue}{#1}}\medskip
7667   }
7668 }

```

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

EdN:10

`\pause` 10

```

7669 \bool_if:NT \c__notesslides_notes_bool {
7670   \newcommand\pause{}
7671 }

```

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

`nparagraph (env.)`

```

7672 \bool_if:NTF \c__notesslides_notes_bool {
7673   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7674 }{
7675   \excludecomment{nparagraph}
7676 }

```

`nfragment (env.)`

```

7677 \bool_if:NTF \c__notesslides_notes_bool {
7678   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1][#2]}\end{sfragment}}
7679 }{
7680   \excludecomment{nfragment}
7681 }

```

`ndefinition (env.)`

```

7682 \bool_if:NTF \c__notesslides_notes_bool {
7683   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7684 }{
7685   \excludecomment{ndefinition}
7686 }

```

`nassertion (env.)`

```

7687 \bool_if:NTF \c__notesslides_notes_bool {
7688   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7689 }{
7690   \excludecomment{nassertion}
7691 }

```

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

`nsproof (env.)`

```
7692 \bool_if:NTF \c__notesslides_notes_bool {
7693   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7694 }{
7695   \excludecomment{nproof}
7696 }
```

`nexample (env.)`

```
7697 \bool_if:NTF \c__notesslides_notes_bool {
7698   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7699 }{
7700   \excludecomment{nexample}
7701 }
```

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

```
7702 \def\inputref@preskip{\smallskip}
7703 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
7704 \let\orig@inputref\inputref
7705 \def\inputref{@ifstar\ninputref\orig@inputref}
7706 \newcommand\ninputref[2] []{
7707   \bool_if:NT \c__notesslides_notes_bool {
7708     \orig@inputref[#1]{#2}
7709   }
7710 }
```

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

38.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

`\setslidelogo` The default logo is the \TeX logo. Customization can be done by `\setslidelogo{<logo name>}`.

```
7711 \newlength{\slidelogoheight}
7712
7713 \RequirePackage{graphicx}
7714
7715 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7716 \providecommand\mhgraphics[2] []{
7717   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7718   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7719 }
7720
7721 \bool_if:NTF \c__notesslides_notes_bool {
7722   \setlength{\slidelogoheight}{.4cm}
7723 }{
7724   \setlength{\slidelogoheight}{.25cm}
7725 }
```

```

7726 \ifcsname slidelogo\endcsname\else
7727 \newsavebox{\slidelogo}
7728 \sbox{\slidelogo}{\TeX}
7729 \fi
7730 \newrobustcmd{\setslidelogo}[2][\{
7731 \tl_if_empty:nTF{#1}{
7732 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7733 }{
7734 \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7735 }
7736 }

```

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

\author In notes mode, we redefine the `\author` macro so that it does not disregard the optional argument (as `beamerarticle` does). We want to use it to set the source later.

```

7737 \bool_if:NT \c__notesslides_notes_bool {
7738 \def\author{\@dblarg\ns@author}
7739 \long\def\ns@author[#1]#2{%
7740 \def\c__notesslides_shortauthor{#1}%
7741 \def\@author{#2}
7742 }
7743 }

```

(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.

```

7744 \newrobustcmd{\setsource}[1]{\def\source{#1}}

```

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

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

```

7745 \def\copyrightnotice{%
7746 \footnotesize\copyright : \hspace{.3ex}%
7747 \ifcsname source\endcsname\source\else%
7748 \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7749 \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7750 ?source/author?\fi%
7751 \fi}
7752 \newsavebox{\cclogo}
7753 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7754 \newif\ifcchref\cchreffalse
7755 \AtBeginDocument{
7756 \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7757 }
7758 \def\licensing{
7759 \ifcchref
7760 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7761 \else
7762 {\usebox{\cclogo}}

```



```

7763 \fi
7764 }
7765 \newrobustcmd{\setlicensing}[2][]{
7766   \def\@url{#1}
7767   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7768   \ifx\@url\@empty
7769     \def\licensing{\usebox{\cclogo}}
7770   \else
7771     \def\licensing{
7772       \ifcchref
7773         \href{#1}{\usebox{\cclogo}}
7774       \else
7775         {\usebox{\cclogo}}
7776       \fi
7777     }
7778   \fi
7779 }

```

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

EdN:11

```

\slidelabel Now, we set up the slide label for the article mode.11
7780 \newrobustcmd\miko@slidelabel{
7781   \vbox to \slidelogoheight{
7782     \vss\hbox to \slidewidth
7783       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7784   }
7785 }

```

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

38.4 Frame Images

\frameimage We have to make sure that the width is overwritten, for that we check the \Gin@ewidth macro from the graphicx package. We also add the label key.

```

7786 \def\Gin@mhrepos{}
7787 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7788 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7789 \newrobustcmd\frameimage[2][]{
7790   \stepcounter{slide}
7791   \bool_if:NT \c__notesslides_frameimages_bool {
7792     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7793     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7794     \begin{center}
7795       \bool_if:NTF \c__notesslides_fiboxed_bool {
7796         \fbox{
7797           \ifx\Gin@ewidth\@empty
7798             \ifx\Gin@mhrepos\@empty
7799               \mhgraphics[width=\slidewidth,#1]{#2}
7800             \else
7801               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7802             \fi
7803           \else% Gin@ewidth empty

```

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

```

7804         \ifx\Gin@mhrepos\@empty
7805         \mhgraphics[#1]{#2}
7806     \else
7807         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7808     \fi
7809 \fi% Gin@ewidth empty
7810 }
7811 }{
7812     \ifx\Gin@ewidth\@empty
7813     \ifx\Gin@mhrepos\@empty
7814         \mhgraphics[width=\slidewidth,#1]{#2}
7815     \else
7816         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7817     \fi
7818     \ifx\Gin@mhrepos\@empty
7819         \mhgraphics[#1]{#2}
7820     \else
7821         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7822     \fi
7823     \fi% Gin@ewidth empty
7824 }
7825 \end{center}
7826 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7827 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7828 }
7829 } % ifmks@sty@frameimages

```

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

38.5 Sectioning

If the `sectocframes` option is set, then we make section frames. We first define counters for `part` and `chapter`, which `beamer.cls` does not have and we make the `section` counter which it does dependent on `chapter`.

```

7830 \stex_html_backend:F {
7831     \bool_if:NT \c__notesslides_sectocframes_bool {
7832         \str_if_eq:VnTF \__notesslidestopsect{part}{
7833             \newcounter{chapter}\counterwithin*{section}{chapter}
7834         }{
7835             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7836                 \newcounter{chapter}\counterwithin*{section}{chapter}
7837             }
7838         }
7839     }
7840 }

```

`\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`

```

7841 \def\part@prefix{}
7842 \@ifpackageloaded{document-structure}{
7843     \str_case:VnF \__notesslidestopsect {

```

```

7844 {part}{
7845   \int_set:Nn \l_document_structure_section_level_int {0}
7846   \def\thesection{\arabic{chapter}.\arabic{section}}
7847   \def\part@prefix{\arabic{chapter}.}
7848 }
7849 {chapter}{
7850   \int_set:Nn \l_document_structure_section_level_int {1}
7851   \def\thesection{\arabic{chapter}.\arabic{section}}
7852   \def\part@prefix{\arabic{chapter}.}
7853 }
7854 }{
7855   \int_set:Nn \l_document_structure_section_level_int {2}
7856   \def\part@prefix{}
7857 }
7858 }
7859
7860 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the `sfragment` environment that chooses the L^AT_EX sectioning macros according to `\section@level`.

`sfragment (env.)`

```

7861 \renewenvironment{sfragment}[2][]{
7862   \__document_structure_sfragment_args:n { #1 }
7863   \int_incr:N \l_document_structure_section_level_int
7864   \bool_if:NT \c__notesslides_sectocframes_bool {
7865     \stepcounter{slide}
7866     \begin{frame}[noframenumbering]
7867     \vfill\Large\centering
7868     \red{
7869       \ifcase\l_document_structure_section_level_int\or
7870         \stepcounter{part}
7871         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7872         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7873         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7874         \def\currentsectionlevel{\omdoc@part@kw}
7875       \or
7876         \stepcounter{chapter}
7877         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7878         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7879         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7880         \def\currentsectionlevel{\omdoc@chapter@kw}
7881       \or
7882         \stepcounter{section}
7883         \def\__notesslideslabel{\part@prefix\arabic{section}}
7884         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7885         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7886         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7887         \def\currentsectionlevel{\omdoc@section@kw}
7888       \or
7889         \stepcounter{subsection}
7890         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7891         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

7892         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7893         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7894         \def\currentsectionlevel{\omdoc@subsection@kw}
7895     \or
7896         \stepcounter{subsubsection}
7897         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7898         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7899         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\thesubsubsection}
7900         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesubsubsection}
7901         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7902     \or
7903         \stepcounter{paragraph}
7904         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7905         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7906         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\theparagraph}
7907         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\theparagraph}
7908         \def\currentsectionlevel{\omdoc@paragraph@kw}
7909     \else
7910         \def\__notesslideslabel{}
7911         \def\currentsectionlevel{\omdoc@paragraph@kw}
7912     \fi% end ifcase
7913     \__notesslideslabel\quad #2%
7914 }%
7915 \vfill%
7916 \end{frame}%
7917 }
7918 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7919     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7920 }
7921 }{}
7922 }

```

We set up a beamer template for theorems like ams style, but without a block environment.

```

7923 \def\inserttheorembodyfont{\normalfont}
7924 %\bool_if:NF \c__notesslides_notes_bool {
7925 %   \defbeamertemplate{theorem begin}{miko}
7926 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7927 %    \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7928 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
7929 %   \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

7930 % \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.

```

7931 % \expandafter\def\csname Parent2\endcsname{}
7932 %}
7933
7934 \AddToHook{begindocument}{ % this does not work for some reason
7935     \setbeamertemplate{theorems}[ams style]
7936 }
7937 \bool_if:NT \c__notesslides_notes_bool {
7938     \renewenvironment{columns}[1][{}]{%

```

```

7939     \par\noindent%
7940     \begin{minipage}%
7941     \slidewidth\centering\leavevmode%
7942   }{%
7943     \end{minipage}\par\noindent%
7944   }%
7945   \newsavebox\columnbox%
7946   \renewenvironment<>{column}[2][]{%
7947     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7948   }{%
7949     \end{minipage}\end{lrbox}\usebox\columnbox%
7950   }%
7951 }

7952 \bool_if:NTF \c__notesslides_noproblems_bool {
7953   \newenvironment{problems}{}{}
7954 }{
7955   \excludacomment{problems}
7956 }

```

38.6 Excursions

\excursion The excursion macros are very simple, we define a new internal macro `\excursionref` and use it in `\excursion`, which is just an `\inputref` that checks if the new macro is defined before formatting the file in the argument.

```

7957 \gdef\printexcursions{}
7958 \newcommand\excursionref[2]{% label, text
7959   \bool_if:NT \c__notesslides_notes_bool {
7960     \begin{sparagraph}[title=Excursion]
7961       #2 \sref[fallback=the appendix]{#1}.
7962     \end{sparagraph}
7963   }
7964 }
7965 \newcommand\activate@excursion[2][{}{
7966   \gappto\printexcursions{\inputref{#1}{#2}}
7967 }
7968 \newcommand\excursion[4][{}{ repos, label, path, text
7969   \bool_if:NT \c__notesslides_notes_bool {
7970     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7971   }
7972 }

```

(End definition for `\excursion`. This function is documented on page 62.)

\excursiongroup

```

7973 \keys_define:nn{notesslides / excursiongroup }{
7974   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7975   intro       .tl_set:N    = \l__notesslides_excursion_intro_tl,
7976   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7977 }
7978 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7979   \tl_clear:N \l__notesslides_excursion_intro_tl
7980   \str_clear:N \l__notesslides_excursion_id_str

```

```

7981 \str_clear:N \l__notesslides_excursion_mhrepos_str
7982 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7983 }
7984 \newcommand\excursionsgroup[1][]{
7985   \__notesslides_excursion_args:n{ #1 }
7986   \ifdefempty\printexcursions{}% only if there are excursions
7987   {\begin{note}
7988     \begin{sfragment}[#1]{Excursions}%
7989     \ifdefempty\l__notesslides_excursion_intro_tl}{
7990       \inputref[\l__notesslides_excursion_mhrepos_str]{
7991         \l__notesslides_excursion_intro_tl
7992       }
7993     }
7994     \printexcursions%
7995     \end{sfragment}
7996   \end{note}}
7997 }
7998 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7999 \end{package}

```

(End definition for \excursionsgroup. This function is documented on page 62.)

Chapter 39

The Implementation

39.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
8000 <*package>
8001 <@@=problems>
8002 \ProvidesExplPackage{problem}{2022/08/08}{3.2.0}{Semantic Markup for Problems}
8003 \RequirePackage{13keys2e}
8004 \RequirePackage{amssymb}% for \Box
8005
8006 \keys_define:nn { problem / pkg }{
8007   notes      .default:n    = { true },
8008   notes      .bool_set:N   = \c__problems_notes_bool,
8009   gnotes     .default:n    = { true },
8010   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
8011   hints      .default:n    = { true },
8012   hints      .bool_set:N   = \c__problems_hints_bool,
8013   solutions  .default:n    = { true },
8014   solutions  .bool_set:N   = \c__problems_solutions_bool,
8015   pts        .default:n    = { true },
8016   pts        .bool_set:N   = \c__problems_pts_bool,
8017   min        .default:n    = { true },
8018   min        .bool_set:N   = \c__problems_min_bool,
8019   boxed      .default:n    = { true },
8020   boxed      .bool_set:N   = \c__problems_boxed_bool,
8021   test       .default:n    = { true },
8022   test       .bool_set:N   = \c__problems_test_bool,
8023   unknown    .code:n       = {
8024     \PassOptionsToPackage{\CurrentOption}{stex}
8025   }
8026 }
8027 \newif\ifsolutions
8028
8029 \ProcessKeysOptions{ problem / pkg }
8030 \bool_if:NTF \c__problems_solutions_bool {
8031   \solutionstrue
```

```

8032 }{
8033   \solutionsfalse
8034 }
8035 \RequirePackage{stex}

```

Then we make sure that the necessary packages are loaded (in the right versions).

```

8036 \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.

```

8037 \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.

```

8038 \def\prob@problem@kw{Problem}
8039 \def\prob@solution@kw{Solution}
8040 \def\prob@hint@kw{Hint}
8041 \def\prob@note@kw{Note}
8042 \def\prob@gnote@kw{Grading}
8043 \def\prob@pt@kw{pt}
8044 \def\prob@min@kw{min}
8045 \def\prob@correct@kw{Correct}
8046 \def\prob@wrong@kw{Wrong}

```

(End definition for `\prob@*@kw`. This function is documented on page ??.)

For the other languages, we set up triggers

```

8047 \AddToHook{begindocument}{
8048   \ltx@ifpackageloaded{babel}{
8049     \makeatletter
8050     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8051     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8052       \input{problem-ngerman.ldf}
8053     }
8054     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8055       \input{problem-finnish.ldf}
8056     }
8057     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8058       \input{problem-french.ldf}
8059     }
8060     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8061       \input{problem-russian.ldf}
8062     }
8063     \makeatother
8064   }{ }
8065 }

```

39.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```

8066 \keys_define:nn{ problem / problem }{
8067   id .str_set_x:N = \l__problems_prob_id_str,

```



```

8068 pts      .tl_set:N      = \l__problems_prob_pts_tl,
8069 min      .tl_set:N      = \l__problems_prob_min_tl,
8070 title    .tl_set:N      = \l__problems_prob_title_tl,
8071 type     .tl_set:N      = \l__problems_prob_type_tl,
8072 imports  .tl_set:N      = \l__problems_prob_imports_tl,
8073 name     .str_set_x:N     = \l__problems_prob_name_str,
8074 refnum   .int_set:N      = \l__problems_prob_refnum_int
8075 }
8076 \cs_new_protected:Nn \__problems_prob_args:n {
8077   \str_clear:N \l__problems_prob_id_str
8078   \str_clear:N \l__problems_prob_name_str
8079   \tl_clear:N \l__problems_prob_pts_tl
8080   \tl_clear:N \l__problems_prob_min_tl
8081   \tl_clear:N \l__problems_prob_title_tl
8082   \tl_clear:N \l__problems_prob_type_tl
8083   \tl_clear:N \l__problems_prob_imports_tl
8084   \int_zero_new:N \l__problems_prob_refnum_int
8085   \keys_set:nn { problem / problem }{ #1 }
8086   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
8087     \let\l__problems_prob_refnum_int\undefined
8088   }
8089 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

8090 \newcounter{problem}[section]
8091 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
8092 \def\theplainsproblem{\arabic{problem}}
8093 \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.

```

8094 \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

```

8095 \newcommand\prob@number{
8096   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
8097     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
8098   }{
8099     \int_if_exist:NTF \l__problems_prob_refnum_int {
8100       \prob@label{\int_use:N \l__problems_prob_refnum_int }
8101     }{
8102       \prob@label\theplainsproblem
8103     }
8104   }
8105 }
8106 \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.

```

8107 \newcommand\prob@title[3]{%
8108   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
8109     #2 \l__problems_inclprob_title_tl #3
8110   }{
8111     \tl_if_empty:NTF \l__problems_prob_title_tl {
8112       #1
8113     }{
8114       #2 \l__problems_prob_title_tl #3
8115     }
8116   }
8117 }

```

(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.

```

8118 \def\prob@heading{
8119   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
8120   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
8121 }

```

(End definition for `\prob@heading`. This function is documented on page ??.)

With this in place, we can now define the `problem` environment. It comes in two shapes, depending on whether we are in boxed mode or not. In both cases we increment the problem number and output the points and minutes (depending) on whether the respective options are set.

`sproblem (env.)`

```

8122 \newenvironment{sproblem}[1][]{
8123   \__problems_prob_args:n{#1}%\sref@target%
8124   \@in@omtexttrue% we are in a statement (for inline definitions)
8125   \refstepcounter{sproblem}\record@problem
8126   \def\current@section@level{\prob@problem@kw}
8127
8128   \str_if_empty:NT \l__problems_prob_name_str {
8129     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
8130     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
8131     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
8132   }
8133
8134   \stex_if_do_html:T{
8135     \tl_if_empty:NF \l__problems_prob_title_tl {
8136       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
8137     }
8138   }
8139
8140   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
8141
8142   \stex_reactivate_macro:N \STEXexport
8143   \stex_reactivate_macro:N \importmodule

```

```

8144 \stex_reactivate_macro:N \symdecl
8145 \stex_reactivate_macro:N \notation
8146 \stex_reactivate_macro:N \symdef
8147
8148 \stex_if_do_html:T{
8149   \begin{stex_annotate_env} {problem} {
8150     \l_stex_module_ns_str ? \l_stex_module_name_str
8151   }
8152
8153   \stex_annotate_invisible:nnn{header}{} {
8154     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
8155     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
8156     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
8157       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
8158     }
8159   }
8160 }
8161
8162 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
8163
8164
8165 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
8166   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
8167 }{
8168   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
8169 }
8170 \str_if_exist:NTF \l__problems_inclprob_id_str {
8171   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
8172 }{
8173   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
8174 }
8175
8176
8177 \stex_if_smsmode:F {
8178   \clist_set:No \l_tmpa_clist \sproblemtype
8179   \tl_clear:N \l_tmpa_tl
8180   \clist_map_inline:Nn \l_tmpa_clist {
8181     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8182       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8183     }
8184   }
8185   \tl_if_empty:NTF \l_tmpa_tl {
8186     \__problems_sproblem_start:
8187   }{
8188     \l_tmpa_tl
8189   }
8190 }
8191 \stex_ref_new_doc_target:n \sproblemid
8192 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8193 }{
8194   \__stex_modules_end_module:
8195   \stex_if_smsmode:F{
8196     \clist_set:No \l_tmpa_clist \sproblemtype
8197     \tl_clear:N \l_tmpa_tl

```

```

8198 \clist_map_inline:Nn \l_tmpa_clist {
8199 \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8200 \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
8201 }
8202 }
8203 \tl_if_empty:NTF \l_tmpa_tl {
8204 \__problems_sproblem_end:
8205 }{
8206 \l_tmpa_tl
8207 }
8208 }
8209 \stex_if_do_html:T{
8210 \end{stex_annotate_env}
8211 }
8212
8213 \smallskip
8214 }
8215
8216 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8217
8218
8219
8220 \cs_new_protected:Nn \__problems_sproblem_start: {
8221 \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
8222 }
8223 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
8224
8225 \newcommand\stexpatchproblem[3][] {
8226 \str_set:Nx \l_tmpa_str{ #1 }
8227 \str_if_empty:NTF \l_tmpa_str {
8228 \tl_set:Nn \__problems_sproblem_start: { #2 }
8229 \tl_set:Nn \__problems_sproblem_end: { #3 }
8230 }{
8231 \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8232 \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8233 }
8234 }
8235
8236
8237 \bool_if:NT \c__problems_boxed_bool {
8238 \surroundwithmdframed{problem}
8239 }

```

\record@problem This macro records information about the problems in the *.aux file.

```

8240 \def\record@problem{
8241 \protected@write\@auxout{}
8242 {
8243 \string\@problem{\prob@number}
8244 {
8245 \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8246 \l__problems_inclprob_pts_tl
8247 }{
8248 \l__problems_prob_pts_tl
8249 }

```

```

8250 }%
8251 {
8252   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8253     \l__problems_inclprob_min_tl
8254   }{
8255     \l__problems_prob_min_tl
8256   }
8257 }
8258 }
8259 }

```

(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).

```

8260 \def\@problem#1#2#3{}

```

(End definition for `\@problem`. This function is documented on page ??.)

`solution (env.)` The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```

8261 \keys_define:nn { problem / solution }{
8262   id          .str_set_x:N = \l__problems_solution_id_str ,
8263   for         .str_set_x:N = \l__problems_solution_for_str ,
8264   type       .str_set_x:N = \l__problems_solution_type_str ,
8265   title      .tl_set:N     = \l__problems_solution_title_tl
8266 }
8267 \cs_new_protected:Nn \__problems_solution_args:n {
8268   \str_clear:N \l__problems_solution_id_str
8269   \str_clear:N \l__problems_solution_type_str
8270   \str_clear:N \l__problems_solution_for_str
8271   \tl_clear:N \l__problems_solution_title_tl
8272   \keys_set:nn { problem / solution }{ #1 }
8273 }

```

`\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.

```

8274 \box_new:N \l__problems_solution_box
8275 \newenvironment{solution}[1][{}]{
8276   \__problems_solution_args:n{#1}
8277   \stex_html_backend:TF{
8278     \stex_if_do_html:T{
8279       \begin{stex_annotate_env}{solution}{ }
8280       \str_if_empty:NF \l__problems_solution_type_str {
8281         \par\noindent
8282         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
8283       }
8284       \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8285     }
8286   }{
8287     \setbox\l__problems_solution_box\vbox\bgroup
8288     \par\smallskip\hrule\smallskip
8289     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems
8290   }

```

```

8291 }{
8292   \stex_html_backend:TF{
8293     \stex_if_do_html:T{
8294       \end{stex_annotate_env}
8295     }
8296   }{
8297     \smallskip\hrule
8298     \egroup
8299     \bool_if:NT \c__problems_solutions_bool {
8300       \strut\par\noindent
8301       \box\l__problems_solution_box
8302     }
8303   }
8304 }
8305
8306 \newcommand\startsolutions{
8307   \bool_set_true:N \c__problems_solutions_bool
8308   \solutionstrue
8309   % \specialcomment{solution}{\@startsolution}{
8310   %   \bool_if:NF \c__problems_boxed_bool {
8311   %     \hrule\medskip
8312   %   }
8313   %   \end{small}%
8314   % }
8315   % \bool_if:NT \c__problems_boxed_bool {
8316   %   \surroundwithmdframed{solution}
8317   % }
8318 }

```

(End definition for \startsolutions. This function is documented on page 64.)

\stopsolutions

```

8319 \newcommand\stopsolutions{\bool_set_false:N \c__problems_solutions_bool \solutionsfalse}%\ex

```

(End definition for \stopsolutions. This function is documented on page 64.)

exnote (env.)

```

8320 \bool_if:NTF \c__problems_notes_bool {
8321   \newenvironment{exnote}[1][]{
8322     \par\smallskip\hrule\smallskip
8323     \noindent\textbf{\prob@note@kw :~ }\small
8324   }{
8325     \smallskip\hrule
8326   }
8327 }{
8328   \excludacomment{exnote}
8329 }

```

hint (env.)

```

8330 \bool_if:NTF \c__problems_notes_bool {
8331   \newenvironment{hint}[1][]{
8332     \par\smallskip\hrule\smallskip
8333     \noindent\textbf{\prob@hint@kw :~ }\small
8334   }{

```

```

8335     \smallskip\hrule
8336   }
8337   \newenvironment{exhint}[1][ ]{
8338     \par\smallskip\hrule\smallskip
8339     \noindent\textbf{\prob@hint@kw :~ }\small
8340   }{
8341     \smallskip\hrule
8342   }
8343   }{
8344     \excludecomment{hint}
8345     \excludecomment{exhint}
8346   }

```

gnote (*env.*)

```

8347   \bool_if:NTF \c__problems_notes_bool {
8348     \newenvironment{gnote}[1][ ]{
8349       \par\smallskip\hrule\smallskip
8350       \noindent\textbf{\prob@gnote@kw :~ }\small
8351     }{
8352       \smallskip\hrule
8353     }
8354   }{
8355     \excludecomment{gnote}
8356   }

```

39.3 Markup for Added Value Services

39.4 Multiple Choice Blocks

EdN:12

mcb (*env.*) ¹²

```

8357   \newenvironment{mcb}{
8358     \begin{enumerate}
8359   }{
8360     \end{enumerate}
8361   }

```

we define the keys for the mcb macro

```

8362   \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8363     \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8364       \bool_set_true:N #1
8365     }{
8366       \bool_set_false:N #1
8367     }
8368   }
8369   \keys_define:nn { problem / mcb }{
8370     id .str_set_x:N = \l__problems_mcc_id_str ,
8371     feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
8372     T .default:n = { false } ,
8373     T .bool_set:N = \l__problems_mcc_t_bool ,
8374     F .default:n = { false } ,

```

¹²EdNOTE: MK: maybe import something better here from a dedicated MC package

```

8375 F .bool_set:N = \l__problems_mcc_f_bool ,
8376 Ttext .tl_set:N = \l__problems_mcc_Ttext_tl ,
8377 Ftext .tl_set:N = \l__problems_mcc_Ftext_tl
8378 }
8379 \cs_new_protected:Nn \l__problems_mcc_args:n {
8380 \str_clear:N \l__problems_mcc_id_str
8381 \tl_clear:N \l__problems_mcc_feedback_tl
8382 \bool_set_false:N \l__problems_mcc_t_bool
8383 \bool_set_false:N \l__problems_mcc_f_bool
8384 \tl_clear:N \l__problems_mcc_Ttext_tl
8385 \tl_clear:N \l__problems_mcc_Ftext_tl
8386 \str_clear:N \l__problems_mcc_id_str
8387 \keys_set:nn { problem / mcc }{ #1 }
8388 }

```

\mcc

```

8389 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8390 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8391 \newcommand\mcc[2][] {
8392 \l__problems_mcc_args:n{ #1 }
8393 \item[{$\Box$}] #2
8394 \bool_if:NT \c__problems_solutions_bool {
8395 \
8396 \bool_if:NT \l__problems_mcc_t_bool {
8397 \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8398 }
8399 \bool_if:NT \l__problems_mcc_f_bool {
8400 \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8401 }
8402 \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8403 \emph{\l__problems_mcc_feedback_tl}
8404 }
8405 }
8406 } %solutions

```

(End definition for \mcc. This function is documented on page 65.)

39.5 Filling in Concrete Solutions

\includeproblem This is embarrassingly simple, but can grow over time.

```

8407 \newcommand\fillinsol[2][] {%
8408 \def\@test{#1}
8409 \quad%
8410 \ifsolutions\textcolor{red}{#1!}\else%
8411 \fbox{\ifx\@test\empty\phantom{huge{21}}\else\hspace{#1}\fi}%
8412 \fi}

```

(End definition for \includeproblem. This function is documented on page 67.)

39.6 Including Problems

\includeproblem The `\includeproblem` command is essentially a glorified `\input` statement, it sets some internal macros first that overwrite the local points. Importantly, it resets the `inclprob` keys after the input.

```

8413
8414 \keys_define:nn{ problem / inclproblem }{
8415   id      .str_set_x:N = \l__problems_inclprob_id_str,
8416   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8417   min     .tl_set:N    = \l__problems_inclprob_min_tl,
8418   title   .tl_set:N    = \l__problems_inclprob_title_tl,
8419   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
8420   type    .tl_set:N    = \l__problems_inclprob_type_tl,
8421   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8422 }
8423 \cs_new_protected:Nn \l__problems_inclprob_args:n {
8424   \str_clear:N \l__problems_prob_id_str
8425   \tl_clear:N \l__problems_inclprob_pts_tl
8426   \tl_clear:N \l__problems_inclprob_min_tl
8427   \tl_clear:N \l__problems_inclprob_title_tl
8428   \tl_clear:N \l__problems_inclprob_type_tl
8429   \int_zero_new:N \l__problems_inclprob_refnum_int
8430   \str_clear:N \l__problems_inclprob_mhrepos_str
8431   \keys_set:nn { problem / inclproblem }{ #1 }
8432   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8433     \let\l__problems_inclprob_pts_tl\undefined
8434   }
8435   \tl_if_empty:NT \l__problems_inclprob_min_tl {
8436     \let\l__problems_inclprob_min_tl\undefined
8437   }
8438   \tl_if_empty:NT \l__problems_inclprob_title_tl {
8439     \let\l__problems_inclprob_title_tl\undefined
8440   }
8441   \tl_if_empty:NT \l__problems_inclprob_type_tl {
8442     \let\l__problems_inclprob_type_tl\undefined
8443   }
8444   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8445     \let\l__problems_inclprob_refnum_int\undefined
8446   }
8447 }
8448
8449 \cs_new_protected:Nn \l__problems_inclprob_clear: {
8450   \let\l__problems_inclprob_id_str\undefined
8451   \let\l__problems_inclprob_pts_tl\undefined
8452   \let\l__problems_inclprob_min_tl\undefined
8453   \let\l__problems_inclprob_title_tl\undefined
8454   \let\l__problems_inclprob_type_tl\undefined
8455   \let\l__problems_inclprob_refnum_int\undefined
8456   \let\l__problems_inclprob_mhrepos_str\undefined
8457 }
8458 \l__problems_inclprob_clear:
8459
8460 \newcommand\includeproblem[2][ ]{
8461   \l__problems_inclprob_args:n{ #1 }

```

```

8462 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8463   \stex_html_backend:TF {
8464     \str_clear:N \l_tmpa_str
8465     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8466       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8467     }
8468     \stex_annotate_invisible:nnn{includeproblem}{
8469       \l_tmpa_str / #2
8470     }{}
8471   }{
8472     \begingroup
8473     \inputreftrue
8474     \tl_if_empty:nTF{ ##1 }{
8475       \input{#2}
8476     }{
8477       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8478     }
8479     \endgroup
8480   }
8481 }
8482 \__problems_inclprob_clear:
8483 }

```

(End definition for `\includeproblem`. This function is documented on page 67.)

39.7 Reporting Metadata

For messages it is OK to have them in English as the whole documentation is, and we can therefore assume authors can deal with it.

```

8484 \AddToHook{enddocument}{
8485   \bool_if:NT \c__problems_pts_bool {
8486     \message{Total:~\arabic{pts}~points}
8487   }
8488   \bool_if:NT \c__problems_min_bool {
8489     \message{Total:~\arabic{min}~minutes}
8490   }
8491 }

```

The margin pars are reader-visible, so we need to translate

```

8492 \def\pts#1{
8493   \bool_if:NT \c__problems_pts_bool {
8494     \marginpar{#1~\prob@pt@kw}
8495   }
8496 }
8497 \def\min#1{
8498   \bool_if:NT \c__problems_min_bool {
8499     \marginpar{#1~\prob@min@kw}
8500   }
8501 }

```

`\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.

```

8502 \newcounter{pts}
8503 \def\show@pts{
8504   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8505     \bool_if:NT \c__problems_pts_bool {
8506       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8507       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8508     }
8509   }{
8510     \tl_if_exist:NT \l__problems_prob_pts_tl {
8511       \bool_if:NT \c__problems_pts_bool {
8512         \tl_if_empty:NT\l__problems_prob_pts_tl{
8513           \tl_set:Nn \l__problems_prob_pts_tl {0}
8514         }
8515         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8516         \addtocounter{pts}{\l__problems_prob_pts_tl}
8517       }
8518     }
8519   }
8520 }

```

(End definition for \show@pts. This function is documented on page ??.)
and now the same for the minutes

\show@min

```

8521 \newcounter{min}
8522 \def\show@min{
8523   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8524     \bool_if:NT \c__problems_min_bool {
8525       \marginpar{\l__problems_inclprob_pts_tl\ min}
8526       \addtocounter{min}{\l__problems_inclprob_min_tl}
8527     }
8528   }{
8529     \tl_if_exist:NT \l__problems_prob_min_tl {
8530       \bool_if:NT \c__problems_min_bool {
8531         \tl_if_empty:NT\l__problems_prob_min_tl{
8532           \tl_set:Nn \l__problems_prob_min_tl {0}
8533         }
8534         \marginpar{\l__problems_prob_min_tl\ min}
8535         \addtocounter{min}{\l__problems_prob_min_tl}
8536       }
8537     }
8538   }
8539 }
8540 \</package>

```

(End definition for \show@min. This function is documented on page ??.)

39.8 Testing and Spacing

\testspace

```

8541 \newcommand\testspace[1]{\bool_if:NT \c__problems_boxed_bool {\vspace*{#1}}}

```

(End definition for \testspace. This function is documented on page ??.)

`\testnewpage`

```
8542 \newcommand\testnewpage{\bool_if:NT \c__problems_boxed_bool {\newpage}}
```

(End definition for \testnewpage. This function is documented on page ??.)

`\testemptypage`

```
8543 \newcommand\testemptypage[1] [] {%
```

```
8544 \bool_if:NT \c__problems_boxed_bool {\begin{center}\hwexam@testemptypage@kw\end{center}\vfil
```

(End definition for \testemptypage. This function is documented on page ??.)

`\test*space`

```
8545 \newcommand\testsmallspace{\testspace{1cm}}
```

```
8546 \newcommand\testmedspace{\testspace{2cm}}
```

```
8547 \newcommand\testbigspace{\testspace{3cm}}
```

*(End definition for \test*space. This function is documented on page ??.)*

Chapter 40

Implementation: The hwexam Package

40.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
8548 \*package>
8549 \ProvidesExplPackage{hwexam}{2022/08/08}{3.2.0}{homework assignments and exams}
8550 \RequirePackage{13keys2e}
8551
8552 \newif\iftest\testfalse
8553 \DeclareOption{test}{\testtrue\PassOptionsToPackage{\CurrentOption}{problem}}
8554 \newif\ifmultiple\multiplefalse
8555 \DeclareOption{multiple}{\multipletrue}
8556 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8557 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8558 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
8559 \RequirePackage{keyval}[1997/11/10]
8560 \RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
8561 \newcommand\hwexam@assignment@kw{Assignment}
8562 \newcommand\hwexam@given@kw{Given}
8563 \newcommand\hwexam@due@kw{Due}
8564 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8565 \newcommand\hwexam@minutes@kw{minutes}
8566 \newcommand\correction@probs@kw{prob.}
8567 \newcommand\correction@pts@kw{total}
8568 \newcommand\correction@reached@kw{reached}
8569 \newcommand\correction@sum@kw{Sum}
8570 \newcommand\correction@grade@kw{grade}
8571 \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

```

8572 \AddToHook{begindocument}{
8573 \ltx@ifpackageloaded{babel}{
8574 \makeatletter
8575 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8576 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8577 \input{hwexam-ngerman.ldf}
8578 }
8579 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8580 \input{hwexam-finnish.ldf}
8581 }
8582 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8583 \input{hwexam-french.ldf}
8584 }
8585 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8586 \input{hwexam-russian.ldf}
8587 }
8588 \makeatother
8589 }{}
8590 }
8591

```

40.2 Assignments

Then we set up a counter for problems and make the problem counter inherited from `problem.sty` depend on it. Furthermore, we specialize the `\prob@label` macro to take the assignment counter into account.

```

8592 \newcounter{assignment}
8593 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8594 \keys_define:nn { hwexam / assignment } {
8595 id .str_set:N = \l_@@_assign_id_str,
8596 number .int_set:N = \l_@@_assign_number_int,
8597 title .tl_set:N = \l_@@_assign_title_tl,
8598 type .tl_set:N = \l_@@_assign_type_tl,
8599 given .tl_set:N = \l_@@_assign_given_tl,
8600 due .tl_set:N = \l_@@_assign_due_tl,
8601 loadmodules .code:n = {
8602 \bool_set_true:N \l_@@_assign_loadmodules_bool
8603 }
8604 }
8605 \cs_new_protected:Nn \_@@_assignment_args:n {
8606 \str_clear:N \l_@@_assign_id_str
8607 \int_set:Nn \l_@@_assign_number_int {-1}
8608 \tl_clear:N \l_@@_assign_title_tl
8609 \tl_clear:N \l_@@_assign_type_tl
8610 \tl_clear:N \l_@@_assign_given_tl
8611 \tl_clear:N \l_@@_assign_due_tl
8612 \bool_set_false:N \l_@@_assign_loadmodules_bool
8613 \keys_set:nn { hwexam / assignment }{ #1 }
8614 }

```

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.

```

8615 \newcommand\given@due[2]{
8616 \bool_lazy_all:nF {
8617 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8618 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8619 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8620 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8621 }{ #1 }
8622
8623 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8624 \tl_if_empty:NF \l_@@_assign_given_tl {
8625 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8626 }
8627 }{
8628 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8629 }
8630
8631 \bool_lazy_or:nnF {
8632 \bool_lazy_and_p:nn {
8633 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8634 }{
8635 \tl_if_empty_p:V \l_@@_assign_due_tl
8636 }
8637 }{
8638 \bool_lazy_and_p:nn {
8639 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8640 }{
8641 \tl_if_empty_p:V \l_@@_assign_due_tl
8642 }
8643 }{ ,~ }
8644
8645 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8646 \tl_if_empty:NF \l_@@_assign_due_tl {
8647 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8648 }
8649 }{
8650 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8651 }
8652
8653 \bool_lazy_all:nF {
8654 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8655 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8656 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8657 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8658 }{ #2 }
8659 }

```

`\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.

```

8660 \newcommand\assignment@title[3]{
8661 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8662 \tl_if_empty:NTF \l_@@_assign_title_tl {
8663 #1
8664 }{
8665 #2\l_@@_assign_title_tl#3
8666 }
8667 }{
8668 #2\l_@@_inclasssign_title_tl#3
8669 }
8670 }

```

(End definition for \assignment@title. This function is documented on page ??.)

\assignment@number Like \assignment@title only for the number, and no around part.

```

8671 \newcommand\assignment@number{
8672 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8673 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8674 \arabic{assignment}
8675 } {
8676 \int_use:N \l_@@_assign_number_int
8677 }
8678 }{
8679 \int_use:N \l_@@_inclasssign_number_int
8680 }
8681 }

```

(End definition for \assignment@number. This function is documented on page ??.)

With them, we can define the central **assignment** environment. This has two forms (separated by \ifmultiple) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents. We first define an assignment counter

assignment (env.) For the **assignment** environment we delegate the work to the **@assignment** environment that depends on whether **multiple** option is given.

```

8682 \newenvironment{assignment}[1][]{
8683 \_@@_assignment_args:n { #1 }
8684 %\sref@target
8685 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8686 \global\stepcounter{assignment}
8687 }{
8688 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8689 }
8690 \setcounter{sproblem}{0}
8691 \renewcommand\prob@label[1]{\assignment@number.##1}
8692 \def\current@section@level{\document@hwexamtype}
8693 %\sref@label{id{\document@hwexamtype \thesection}
8694 \begin{@assignment}
8695 }{
8696 \end{@assignment}
8697 }

```


In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

8698 \def\ass@title{
8699 {\protect\document@hwexamtype}\arabic{assignment}
8700 \assignment@title{}\;{}{}\;} -- \given@due{}\}
8701 }
8702 \ifmultiple
8703 \newenvironment{@assignment}{
8704 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8705 \begin{sfragment}[loadmodules]{\ass@title}
8706 }{
8707 \begin{sfragment}{\ass@title}
8708 }
8709 }{
8710 \end{sfragment}
8711 }

```

for the single-page case we make a title block from the same components.

```

8712 \else
8713 \newenvironment{@assignment}{
8714 \begin{center}\bf
8715 \Large@title\strut\
8716 \document@hwexamtype\arabic{assignment}\assignment@title{}\;{}{}\;{}\\
8717 \large\given@due{--\;}\;{};--}
8718 \end{center}
8719 }{}
8720 \fi% multiple

```

40.3 Including Assignments

\in*assignment This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points. Importantly, it resets the `inclassig` keys after the input.

```

8721 \keys_define:nn { hwexam / inclassignment } {
8722 %id .str_set_x:N = \l_@@_assign_id_str,
8723 number .int_set:N = \l_@@_inclassign_number_int,
8724 title .tl_set:N = \l_@@_inclassign_title_tl,
8725 type .tl_set:N = \l_@@_inclassign_type_tl,
8726 given .tl_set:N = \l_@@_inclassign_given_tl,
8727 due .tl_set:N = \l_@@_inclassign_due_tl,
8728 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8729 }
8730 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8731 \int_set:Nn \l_@@_inclassign_number_int {-1}
8732 \tl_clear:N \l_@@_inclassign_title_tl
8733 \tl_clear:N \l_@@_inclassign_type_tl
8734 \tl_clear:N \l_@@_inclassign_given_tl
8735 \tl_clear:N \l_@@_inclassign_due_tl
8736 \str_clear:N \l_@@_inclassign_mhrepos_str
8737 \keys_set:nn { hwexam / inclassignment }{ #1 }
8738 }
8739 \l_@@_inclassignment_args:n {}
8740
8741 \newcommand\inputassignment[2][{}]{

```

```

8742 \_@@_inclassassignment_args:n { #1 }
8743 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8744 \input{#2}
8745 }{
8746 \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8747 \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8748 }
8749 }
8750 \_@@_inclassassignment_args:n {}
8751 }
8752 \newcommand\includeassignment[2][]{
8753 \newpage
8754 \inputassignment[#1]{#2}
8755 }

```

(End definition for \in*assignment. This function is documented on page ??.)

40.4 Typesetting Exams

\quizheading

```

8756 \ExplSyntaxOff
8757 \newcommand\quizheading[1]{%
8758 \def\@tas{#1}%
8759 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8760 \ifx\@tas\@empty\else%
8761 \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8762 \fi%
8763 }
8764 \ExplSyntaxOn

```

(End definition for \quizheading. This function is documented on page ??.)

\testheading

```

8765
8766 \def\hwexamheader{\input{hwexam-default.header}}
8767
8768 \def\hwexamminutes{
8769 \tl_if_empty:NTF \testheading@duration {
8770 {\testheading@min}~\hwexam@minutes@kw
8771 }{
8772 \testheading@duration
8773 }
8774 }
8775
8776 \keys_define:nn { hwexam / testheading } {
8777 min .tl_set:N = \testheading@min,
8778 duration .tl_set:N = \testheading@duration,
8779 reqpts .tl_set:N = \testheading@reqpts,
8780 tools .tl_set:N = \testheading@tools
8781 }
8782 \cs_new_protected:Nn \_@@_testheading_args:n {
8783 \tl_clear:N \testheading@min
8784 \tl_clear:N \testheading@duration

```

```

8785 \tl_clear:N \testheading@reqpts
8786 \tl_clear:N \testheading@tools
8787 \keys_set:nn { hwexam / testheading }{ #1 }
8788 }
8789 \newenvironment{testheading}[1][]{
8790 \_@@_testheading_args:n{ #1 }
8791 \newcount\check@time\check@time=\testheading@min
8792 \advance\check@time by -\theassignment@totalmin
8793 \newif\if@bonuspoints
8794 \tl_if_empty:NTF \testheading@reqpts {
8795 \@bonuspointsfalse
8796 }{
8797 \newcount\bonus@pts
8798 \bonus@pts=\theassignment@totalpts
8799 \advance\bonus@pts by -\testheading@reqpts
8800 \edef\bonus@pts{\the\bonus@pts}
8801 \@bonuspointstrue
8802 }
8803 \edef\check@time{\the\check@time}
8804
8805 \makeatletter\hwexamheader\makeatother
8806 }{
8807 \newpage
8808 }

```

(End definition for \testheading. 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.

```

8809 <@@=problems>
8810 \renewcommand\@problem[3]{
8811 \stepcounter{assignment@probs}
8812 \def\__problemspts{#2}
8813 \ifx\__problemspts\empty\else
8814 \addtocounter{assignment@totalpts}{#2}
8815 \fi
8816 \def\__problemsmin{#3}\ifx\__problemsmin\empty\else\addtocounter{assignment@totalmin}{#3}\fi
8817 \xdef\correction@probs{\correction@probs & #1}%
8818 \xdef\correction@pts{\correction@pts & #2}
8819 \xdef\correction@reached{\correction@reached &}
8820 }
8821 <@@=hwexam>

```

(End definition for \@problem. This function is documented on page ??.)

\correction@table This macro generates the correction table

```

8822 \newcounter{assignment@probs}
8823 \newcounter{assignment@totalpts}
8824 \newcounter{assignment@totalmin}
8825 \def\correction@probs{\correction@probs@kw}
8826 \def\correction@pts{\correction@pts@kw}
8827 \def\correction@reached{\correction@reached@kw}
8828 \stepcounter{assignment@probs}
8829 \newcommand\correction@table{

```

```

8830 \resizebox{\textwidth}{!}{%
8831 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8832 &\multicolumn{\theassignment@probs}{c|}{%|
8833 {\footnotesize\correction@forgrading@kw} &\\ \hline
8834 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8835 \correction@pts & \theassignment@totalpts & \\ \hline
8836 \correction@reached & & \[.7cm]\hline
8837 \end{tabular}}
8838 \end{package}

```

(End definition for `\correction@table`. This function is documented on page ??.)

40.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

```

here we define the logos that characterize the assignment
\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```

Chapter 41

References

EdN:13

13

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T_EX distribution. The Comprehensive T_EX Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphicx.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

¹³EdNOTE: we need an un-numbered version sfragment*

- [SIa] *sLaTeX/sTeX-IDE*. URL: <https://github.com/slatex/sTeX-IDE> (visited on 04/22/2022).
- [SIb] *sLaTeX/stexls-vscode-plugin*. URL: <https://github.com/slatex/stexls-vscode-plugin> (visited on 04/22/2022).
- [SLS] *sLaTeX/stexls*. URL: <https://github.com/slatex/stexls> (visited on 04/22/2022).
- [ST] *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: <https://ctan.org/pkg/stex> (visited on 04/22/2022).
- [sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: <https://github.com/sLaTeX/sTeX> (visited on 05/11/2020).
- [Tana] Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: <http://ctan.org/pkg/beamer> (visited on 01/07/2014).
- [Tanb] Till Tantau. *User Guide to the Beamer Class*. URL: <http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.
- [TL] *TeX Live*. URL: <http://www.tug.org/texlive/> (visited on 12/11/2012).