

The sTeX3 Package Collection *

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

2022-08-25

Abstract

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

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

sTeX augments L^AT_EX with

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

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

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

*Version 3.2 (last revised 2022-08-25)

Contents

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

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

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

| | | |
|-----------|---|------------|
| 24 | <code>STeX</code>-Basics Implementation | 100 |
| 24.1 | The <code>STeXDocument</code> Class | 100 |
| 24.2 | Preliminaries | 101 |
| 24.3 | Messages and logging | 101 |
| 24.4 | HTML Annotations | 102 |
| 24.5 | Babel Languages | 104 |
| 24.6 | Persistence | 106 |
| 24.7 | Auxiliary Methods | 106 |
| 25 | <code>STeX</code>-MathHub Implementation | 110 |
| 25.1 | Generic Path Handling | 110 |
| 25.2 | PWD and <code>kpsewhich</code> | 112 |
| 25.3 | File Hooks and Tracking | 113 |
| 25.4 | MathHub Repositories | 114 |
| 25.5 | Using Content in Archives | 119 |
| 26 | <code>STeX</code>-References Implementation | 124 |
| 26.1 | Document URIs and URLs | 124 |
| 26.2 | Setting Reference Targets | 126 |
| 26.3 | Using References | 129 |
| 27 | <code>STeX</code>-Modules Implementation | 135 |
| 27.1 | The <code>smodule</code> environment | 139 |
| 27.2 | Invoking modules | 145 |
| 28 | <code>STeX</code>-Module Inheritance Implementation | 147 |
| 28.1 | SMS Mode | 147 |
| 28.2 | Inheritance | 151 |
| 29 | <code>STeX</code>-Symbols Implementation | 157 |
| 29.1 | Symbol Declarations | 157 |
| 29.2 | Notations | 165 |
| 29.3 | Variables | 175 |
| 30 | <code>STeX</code>-Terms Implementation | 184 |
| 30.1 | Symbol Invocations | 184 |
| 30.2 | Terms | 192 |
| 30.3 | Notation Components | 197 |
| 30.4 | Variables | 199 |
| 30.5 | Sequences | 202 |
| 31 | <code>STeX</code>-Structural Features Implementation | 203 |
| 31.1 | Imports with modification | 204 |
| 31.2 | The feature environment | 212 |
| 31.3 | Structure | 212 |
| 32 | <code>STeX</code>-Statements Implementation | 223 |
| 32.1 | Definitions | 223 |
| 32.2 | Assertions | 229 |
| 32.3 | Examples | 232 |
| 32.4 | Logical Paragraphs | 235 |

| | |
|--|------------|
| 33 The Implementation | 240 |
| 33.1 Proofs | 240 |
| 34 \TeX-Others Implementation | 249 |
| 35 \TeX-Metatheory Implementation | 251 |
| 36 Tikzinput Implementation | 254 |
| 37 document-structure.sty Implementation | 257 |
| 37.1 Package Options | 257 |
| 37.2 Document Structure | 258 |
| 37.3 Front and Backmatter | 262 |
| 37.4 Global Variables | 264 |
| 38 NotesSlides – Implementation | 265 |
| 38.1 Class and Package Options | 265 |
| 38.2 Notes and Slides | 267 |
| 38.3 Header and Footer Lines | 271 |
| 38.4 Frame Images | 273 |
| 38.5 Sectioning | 274 |
| 38.6 Excursions | 277 |
| 39 The Implementation | 279 |
| 39.1 Package Options | 279 |
| 39.2 Problems and Solutions | 280 |
| 39.3 Markup for Added Value Services | 287 |
| 39.4 Multiple Choice Blocks | 287 |
| 39.5 Filling in Concrete Solutions | 288 |
| 39.6 Including Problems | 288 |
| 39.7 Reporting Metadata | 290 |
| 40 Implementation: The hwexam Package | 292 |
| 40.1 Package Options | 292 |
| 40.2 Assignments | 293 |
| 40.3 Including Assignments | 296 |
| 40.4 Typesetting Exams | 297 |
| 40.5 Leftovers | 299 |
| 41 References | 300 |

Part I

Manual



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



Boxes like this one explain how some $\text{\texttt{gT\textsubscript{E}X}}$ concept relates to the $\text{\texttt{MMT/OMDoc}}$ system, philosophy or language; see [\[MMT; Koh06\]](#) for introductions.

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

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

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

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

2.1.1 Minimal Setup for the PDF-only Workflow

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.1.5 Manual Setup for Active Documents and Knowledge Management Services

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

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

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

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

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

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

2.2 A First \LaTeX Document

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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

Remark 2.2.1:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

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

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

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

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

```

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

```

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

Remark 2.2.2:

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

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

2.2.2 Mmt/OMDoc Conversion

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

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

Chapter 3

Creating sTeX Content

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

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

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

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

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

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

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

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

3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

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

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

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

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

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

3.2.1 The Local MathHub-Directory

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

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

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

3.2.2 The Structure of \TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

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

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

3.2.3 MANIFEST.MF-Files

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

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

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

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

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

source-base or

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

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

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

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

3.2.4 Using Files in \TeX Archives Directly

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

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

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

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

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

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

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

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

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

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

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

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

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

Remark 3.2.1:

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

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

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

Example 1

Input:

```

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

```

Output:

Hello World

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

For example:

Example 2

Input:

```

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

```

Output:

```

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

```

3.3.2 Declaring New Symbols and Notations

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

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

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

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

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

Example 3

Input:

```

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

```

Output:

```

Given a foo, we can...

```

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

Example 4

Input:

```

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

```

Output:

```

this is a symbol taking two arguments.

```

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

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

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

Example 5

Input:

```

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

```

Output:

```

First: a; Second: b

```

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

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

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

Example 6

Input:

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

Output:

First: a ; Second: b



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



Note that it is required that

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

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

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

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

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

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

Example 7

Input:

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

Output:

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

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

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

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

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

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

Operator Notations

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

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

Example 8

Input:

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

Output:

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

\hookrightarrow `\symbolname!` is translated to OMDoc/MMT as `<OMS name="...?symbolname"/>` 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:
 \hookrightarrow `\begin{smodule}[lang=<lang>]{Foo}` generates a theory `some/namespace?Foo` that only contains the “formal” part of the module – i.e. exactly the content that is exported when using `\importmodule`.
 \rightsquigarrow Additionally, MMT generates a *language theory* `some/namespace/Foo?<lang>` that includes `some/namespace?Foo` and contains all the other document content – variable declarations, includes for each `\usemodule`, etc.

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

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

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

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

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

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

3.4.2 Simple Inheritance and Namespaces

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

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

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



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

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

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

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



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

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



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

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

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

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

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



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



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

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

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

3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

```

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

Output:

A *monoid* is...

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

Example 25

Input:

```

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

Output:

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

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

Example 26

Input:

```

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

Output:

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

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

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

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

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

`\instantiate` generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned based on the respective key-value-list.

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

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

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

This allows us to do things like:

Example 27

Input:

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

Output:

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

.

and

Example 28

Input:

```

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

```

Output:

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

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

3.4.4 The copymodule Environment

TODO: explain

Given modules:

Example 29

Input:

```

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

```

Output:

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

Example 30

Input:

```

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

```

Output:

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

TODO: explain donotclone

3.4.5 The interpretmodule Environment

TODO: explain

Example 31

Input:

```

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

```

Output:

3.5 Primitive Symbols (The STEX Metatheory)

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

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

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

We make this theory part of the STEX collection due to the obiquity of the symbols involved. Note however, that the metatheory is for all practical purposes a “normal” STEX module, and the symbols contained “normal” STEX symbols.

Chapter 4

Using \TeX Symbols

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

4.1 `\symref` and its variants

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

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

Example 32

Input:

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

Output:

```
A natural number is...
```

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

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

Example 33

Input:

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

Output:

Natural numbers are...



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

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

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

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

4.2 Marking Up Text and On-the-Fly Notations

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

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

Example 34

Input:

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

Output:

The sum of n and m is...

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

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



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

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

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

Example 35

Input:

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

Output:

```
Addition is...
```

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

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

Example 36

Input:

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

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

Output:

adding k yields...

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

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

Example 37

Input:

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

Output:

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

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

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

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

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

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

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

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

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

Additionally, the document in which we want to reference a label needs 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}3 document]
```

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 ?? for details.

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

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

Example 38

Input:

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

Output:

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

`\definiendum` **sdefinition** (and **sparagraph** with `type=symdoc`) introduce three new macros: `\definiendum` behaves like `\symref` (and `\definame`/`\Definame` like `\symname`/`\Symname`, respectively), but 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  $\sum_{i=1}^{n-1} n^{2i-1} = n^2$  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.

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

\spfsketch For one-line proof sketches, we use the **\spfsketch** macro, which takes the same optional argument as **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>\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>\cmhtikzinput</code> | |

| | |
|---------------------------------|---|
| <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⟩}{⟨cctx⟩}</code> tests the content of the global variable <code>⟨vname⟩</code> , only if (after expansion) it is equal to <code>⟨val⟩</code> , the conditional text <code>⟨cctx⟩</code> is formatted. |
|-----------------------------------|---|

7.3 Slides and Course Notes

7.3.1 Introduction

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

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

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

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

7.3.2 Package Options

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

| | |
|---------------------------|--|
| <code>slides</code> | 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 \LaTeX becomes confused and throws error messages that are difficult to decipher.

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

```

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

```

```

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

```

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



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



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

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

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

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

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

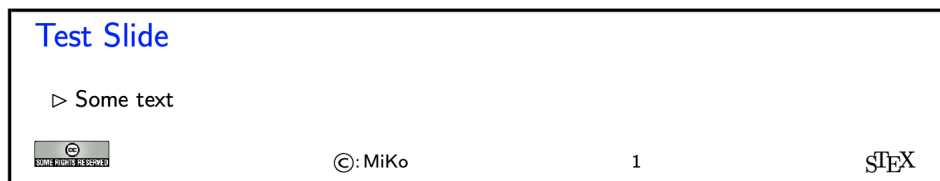
`nexample` (*env.*) level of source indenting). Similarly, we have the `nfragment`, `ndefinition`, `nexample`,

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

`nassertion` (*env.*)

7.3.4 Customizing Header and Footer Lines

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



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

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

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

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

7.3.5 Frame Images

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

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

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

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


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

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

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

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

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

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

7.3.6 Excursions

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

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

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

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

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

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

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

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

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



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

7.4 Representing Problems and Solutions

7.4.1 Introduction

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

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

7.4.2 Problems and Solutions

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

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

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

Example 40

Input:

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

```

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

```

Output:

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

Hint: Think positively, this is simple!

Note: Justify your answer

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

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

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

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

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

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

7.4.3 Markup for Added-Value Services

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

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

Multiple Choice Blocks

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

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

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

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

Example 41

Input:

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

Output:

Problem 7.4.2 (Functions)

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

☐ def

Correct!

☐ function

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

☐ fun

Wrong! *that is for Standard ML*

☐ public static void

Wrong! *that is for Java*

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

Example 42

Input:

```

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

```

Output:

Problem 7.4.3 (Functions)

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

☐ def

☐ function

☐ fun

☐ public static void

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

Filling-In Concrete Solutions

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

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

Example 43

Input:

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

Output:

Problem 7.4.4 (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle?

and the actual solution in solutions mode:

Example 44

Input:

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

Output:

Problem 7.4.5 (Fitting Elephants)

How many Elephants can you fit into a Volkswagen beetle?

4!

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

7.4.4 Including Problems

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

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

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

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

7.5 Homeworks, Quizzes and Exams

7.5.1 Introduction

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

7.5.2 Package Options

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

7.5.3 Assignments

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

7.5.4 Including Assignments

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

7.5.5 Typesetting Exams

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

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

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

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-08-25

You have one hour (sharp) for the test;

Write the solutions to the sheet.

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

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

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

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

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

good luck

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

Part II

Documentation

Chapter 8

STEX-Basics

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

8.1 Macros and Environments

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

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

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

8.1.1 HTML Annotations

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

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

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

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

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

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

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

STEX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

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

9.1.2 MathHub Archives

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`\stex_add_to_current_module:n`
`\STEXexport`

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

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

`\stex_add_constant_to_current_module:n`

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

`\stex_add_import_to_current_module:n`

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

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

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

\stex_modules_current_namespace:

Computes the current namespace as follows:

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

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

11.1.1 The smodule environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

| | |
|--------------------------------------|--|
| <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

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

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

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

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

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

Chapter 13

STEX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

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

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

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

| | |
|---|--|
| <hr/> <hr/> <code>\stex_symdecl_do:n</code> | <p>Implements the core functionality of <code>\symdecl</code>, and is called by <code>\symdecl</code> and <code>\symdef</code>. Ultimately stores the symbol $\langle URI \rangle$ in the property list <code>\l_stex_symdecl_<URI>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>name</code> (string), • <code>module</code> (string), • <code>notations</code> (sequence of strings; initially empty), • <code>local</code> (boolean), • <code>type</code> (token list), • <code>args</code> (string of <code>is</code>, <code>as</code> and <code>bs</code>), • <code>arity</code> (integer string), • <code>assoc</code>s (integer string; number of associative arguments), |
| <hr/> <hr/> <code>\stex_all_symbols:n</code> | <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>s (sequence of integer strings) |
| <hr/> <hr/> <code>\symdef</code> | <p><code>\symdef[<args>]{<symbol>}{<notations⁺>}</code> Combines <code>\symdecl</code> and <code>\notation</code> by introducing a new symbol and assigning a new notation for it.</p> |

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

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

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

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

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

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

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

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

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

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

Chapter 15

TeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` (*env.*) TODO

Chapter 16

TeX-Statements

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

16.1 Macros and Environments

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

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

Chapter 17

sTeX-Proofs: Structural Markup for Proofs

Chapter 18

sTeX-Metatheory

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput: Treating TIKZ code as images

19.1 Macros and Environments

Chapter 20

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

Chapter 21

NotesSlides – Slides and Course Notes

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

Chapter 23

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

Part IV

Implementation

Chapter 24

\TeX -Basics Implementation

24.1 The \TeX Document Class

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

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

```

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

```

24.2 Preliminaries

```

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

```

\stex The sTeX logo:

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

(End definition for `\stex` and `\sTeX`. These functions are documented on page 71.)

24.3 Messages and logging

```

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

24.4 HTML Annotations

```

107 <@@=stex_annotate>

```

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

```

108 \tl_new:N \l_stex_html_arg_tl

```

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

`_stex_html_checkempty:n`

```

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

```

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

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

`\stex_if_do_html:TF`

```

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

```

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

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

```

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

```

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

`\stex_annotate_html: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 72.)

24.5 Babel Languages

```

160 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 72.)

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

```

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

```

```

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

```

24.6 Persistence

```

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

```

24.7 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\ignorespacesandpars`

```

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

```



```

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

```

```

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

```

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

\MMTrule

```

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

```

```

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

```

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

Chapter 25

STEX -MathHub Implementation

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

25.1 Generic Path Handling

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

`\stex_path_from_string:Nn`

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

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

25.3 File Hooks and Tracking

516 `<@@=stex_files>`

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

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

`\g_stex_files_stack` keeps track of file changes

```

517 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

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

```

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

`\g_stex_currentfile_seq`

```

521 \seq_gclear_new:N\g_stex_currentfile_seq

```

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

`\stex_filestack_push:n`

```

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

```

```

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

```

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

`\stex_filestack_pop:`

```

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

```

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

Hooks for the current file:

```

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

```

25.4 MathHub Repositories

```

555 <@@=stex_mathhub>

```

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

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

```

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

```



```

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

```

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

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

```

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

```

```

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

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

621 \seq_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for \l_stex_mathhub_manifest_file_seq.)

__stex_mathhub_find_manifest:N

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

```

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

```

(End definition for __stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior

File variable used for MANIFEST-files

```

657 \ior_new:N \c__stex_mathhub_manifest_ior

```

(End definition for `\c__stex_mathhub_manifest_ior`.)

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

```

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

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

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

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

```

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

```

```

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

```

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

25.5 Using Content in Archives

`\mhpath`

```

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

```

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

`\inputref`

`\mhinput`

```

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

```

```

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

```

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

`\addmhbibresource`

```

838 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {

```

```

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

```

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

\libinput

```

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

```

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

\libusepackage

```

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

```

```

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

```

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

`\mhgraphics`
`\cmhgraphics`

```

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

```

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

`\lstinputmhlisting`
`\clstinputmhlisting`

```

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

```


933

934 `\end{package}`

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

Chapter 26

STEX -References Implementation

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

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

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

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 76.)

`\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 76.)

`\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 76.)

`\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 76.)

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}~\@cu
1093     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1094     \immediate\write\@auxout{\STEXInternalAuxAddDocRef{\l_stex_current_docns_str}{\l_tmpa_str
1095     %\str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1096     }
1097   }
1098   \NewDocumentCommand \slabel {m} {\stex_ref_new_doc_target:n {#1}}
```

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

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

```

1099 \cs_new_protected:Npn \STEXInternalAuxAddDocRef #1 #2 {
1100   \exp_args:Nnx \seq_if_in:NnTF \g_stex_ref_files_seq {\detokenize{#1}} {
1101     \exp_args:Nnx \seq_if_in:cnF{g_stex_ref_ #1 _seq}{\detokenize{#2}}{
1102       \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1103     }
1104   }{
1105     \exp_args:Nnx \seq_gput_right:Nn \g_stex_ref_files_seq {\detokenize{#1}}
1106     %\seq_if_exist:cF{g_stex_ref_ #1 _seq}{
1107       \seq_new:c{g_stex_ref_ #1 _seq} % <- seq_new throws errors??
1108     %}
1109     \exp_args:Nnx \seq_gput_left:cn{g_stex_ref_ #1 _seq}{\detokenize{#2}}
1110   }
1111
1112   %\str_set:Nn \l_tmpa_str {#1?#2}
1113   %\str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1114   %\seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1115     % \seq_new:c {g__stex_refs_labels_#2_seq}
1116     %}
1117   %\seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1118     % \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1119     %}
1120 }

```

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

```

1121 \AtEndDocument{
1122   \def\STEXInternalAuxAddDocRef#1 #2 {}{}
1123 }

```

\stex_ref_new_sym_target:n

```

1124 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1125
1126   % \stex_if_smsmode:TF {
1127   %   \str_if_exist:cF{sref_sym_#1_type}{
1128   %     \stex_get_document_url:
1129   %     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
1130   %     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1131   %   }
1132   % }{
1133   %   \str_if_empty:NF \l__stex_refs_curr_label_str {
1134   %     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1135   %     \immediate\write\@auxout{
1136   %       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label}
1137   %       \l__stex_refs_curr_label_str
1138   %     }
1139   %   }
1140   % }
1141 % }
1142 }

```

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

26.3 Using References

`\sref` Optional arguments:

```

1143
1144 \keys_define:nn { stex / sref / 1 } {
1145   archive .str_set_x:N = \l__stex_refs_repo_str,
1146   file     .str_set_x:N = \l__stex_refs_file_str,
1147   % TODO get rid of this
1148   fallback .code:n = {},
1149   pre      .code:n = {},
1150   post     .code:n = {}
1151 }
1152 \cs_new_protected:Nn \__stex_refs_args_i:n {
1153   \str_clear:N \l__stex_refs_repo_str
1154   \str_clear:N \l__stex_refs_file_str
1155   \keys_set:nn { stex / sref / 1 } { #1 }
1156 }
1157 \keys_define:nn { stex / sref / 2 } {
1158   in .str_set_x:N = \l__stex_refs_in_str,
1159   archive .str_set_x:N = \l__stex_refs_repob_str,
1160   title .tl_set:N = \l__stex_refs_title_tl
1161 }
1162 \cs_new_protected:Nn \__stex_refs_args_ii:n {
1163   \str_clear:N \l__stex_refs_in_str
1164   \tl_clear:N \l__stex_refs_title_tl
1165   \str_clear:N \l__stex_refs_repob_str
1166   \keys_set:nn { stex / sref / 2 } { #1 }
1167 }

```

The actual macro:

```

1168 \NewDocumentCommand \sref { 0{} m 0{} }{
1169   \__stex_refs_args_i:n{#1}
1170   \__stex_refs_args_ii:n{#3}
1171   \str_clear:N \l__stex_refs_uri_str
1172   \__stex_refs_find_uri:n{#2}
1173   \__stex_refs_do_sref:n{#2}
1174 }
1175 \NewDocumentCommand \extref { 0{} m m }{
1176   \__stex_refs_args_i:n{#1}
1177   \__stex_refs_args_ii:n{#3}
1178   \str_if_empty:NT \l__stex_refs_in_str {
1179     \msg_error:nn{stex}{error/extrefmissing}
1180   }
1181   \str_clear:N \l__stex_refs_uri_str
1182   \__stex_refs_find_uri:n{#2}
1183   \__stex_refs_do_sref_in:n{#2}
1184 }
1185
1186 \cs_new_protected:Nn \__stex_refs_find_uri:n {
1187   \stex_debug:nn{sref}{File: \l__stex_refs_file_str^^JRepo:\l__stex_refs_repo_str}
1188   \str_if_empty:NTF \l__stex_refs_file_str {
1189     \seq_if_exist:cT{g_stex_ref_\l_stex_current_docns_str_seq}{
1190       \seq_map_inline:cn{g_stex_ref_\l_stex_current_docns_str_seq}{
1191         \str_if_eq:nnT{#1}{##1}{

```

```

1192         \str_set_eq:NN \l__stex_refs_uri_str \l_stex_current_docns_str
1193         \seq_map_break:
1194     }
1195 }
1196 }
1197 \str_if_empty:NF \l__stex_refs_uri_str {
1198     \seq_map_inline:Nn \g_stex_ref_files_seq {
1199         \seq_map_inline:cn{g_stex_ref_###1_seq}{
1200             \str_if_eq:nnT{#1}{###1}{
1201                 \str_set:Nn \l__stex_refs_uri_str {##1}
1202                 \seq_map_break:n{\seq_map_break:}
1203             }
1204         }
1205     }
1206 }
1207 }{
1208     \str_if_empty:NTF \l__stex_refs_repo_str {
1209         \prop_if_exist:NTF \l_stex_current_repository_prop {
1210             \prop_get:NnN \l_stex_current_repository_prop { ns } \l__stex_refs_uri_str
1211             \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1212             \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1213             \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1214         }{
1215             \stex_path_from_string:Nn \l_tmpb_seq {
1216                 \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_file_str
1217             }
1218             \str_set:Nx \l__stex_refs_uri_str {file:/\stex_path_to_string:N \l_tmpb_seq}
1219         }
1220     }{
1221         \stex_require_repository:n \l__stex_refs_repo_str
1222         \prop_get:cnN { c_stex_mathhub_\l__stex_refs_repo_str _manifest_prop } { ns } \l__stex_refs_uri_str
1223         \str_set:Nx \l__stex_refs_uri_str {\l__stex_refs_uri_str / \l__stex_refs_file_str}
1224         \stex_path_from_string:Nn \l_tmpb_seq \l__stex_refs_uri_str
1225         \str_set:Nx \l__stex_refs_uri_str {\stex_path_to_string:N \l_tmpb_seq}
1226     }
1227 }
1228 }
1229
1230 \cs_new_protected:Nn \__stex_refs_do_autoref:n{
1231     \cs_if_exist:cTF{autoref}{
1232         \exp_args:Nx\autoref{sref_#1}
1233     }{
1234         \exp_args:Nx\ref{sref_#1}
1235     }
1236 }
1237
1238 \cs_new_protected:Nn \__stex_refs_do_sref:n {
1239     \str_if_empty:NTF \l__stex_refs_uri_str {
1240         \str_if_empty:NTF \l__stex_refs_in_str {
1241             \__stex_refs_do_autoref:n{#1}
1242         }{
1243             \__stex_refs_do_sref_in:n{#1}
1244         }
1245     }{

```



```

1246 \exp_args:NNo \seq_if_in:NnTF \g_stex_ref_files_seq \l__stex_refs_uri_str {
1247 \exp_args:Nnx \seq_if_in:cnTF{g_stex_ref_\l__stex_refs_uri_str _seq}{\detokenize{#1}}{
1248 \__stex_refs_do_autoref:n{\l__stex_refs_uri_str?#1}
1249 }{
1250 \str_if_empty:NTF \l__stex_refs_in_str {
1251 \__stex_refs_do_autoref:n{#1}
1252 }{
1253 \__stex_refs_do_sref_in:n{#1}
1254 }
1255 }
1256 }{
1257 \str_if_empty:NTF \l__stex_refs_in_str {
1258 \__stex_refs_do_autoref:n{#1}
1259 }{
1260 \__stex_refs_do_sref_in:n{#1}
1261 }
1262 }
1263 }
1264 }
1265
1266 \cs_new_protected:Nn \__stex_refs_restore_target:nnnnn {
1267 \str_if_empty:NTF \l__stex_refs_uri_str {
1268 \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1269 \tl_set:Nn \l__stex_refs_return_tl {
1270 \use:c{#3autorefname}~#4\tl_if_empty:nF{#5}{~(5)}~in~
1271 \tl_if_empty:nTF\l__stex_refs_title_tl{
1272 ???
1273 }\l__stex_refs_title_tl
1274 }
1275 }
1276 }{
1277 \stex_debug:nn{sref}{\l__stex_refs_uri_str}{~ == ~ #1 ~ ?}
1278 \exp_args:No \str_if_eq:nnT \l__stex_refs_uri_str {#1}{
1279 \stex_debug:nn{sref}{\l__stex_refs_id_str~ == ~ #2 ~ ?}
1280 \exp_args:No \str_if_eq:nnT \l__stex_refs_id_str {#2}{
1281 \stex_debug:nn{sref}{success!}
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 \endinput
1289 }
1290 }
1291 }
1292 }
1293
1294 \cs_new_protected:Nn \__stex_refs_do_sref_in:n {
1295 \stex_debug:nn{sref}{In: \l__stex_refs_in_str^^JRepo:\l__stex_refs_repo_str}
1296 \stex_debug:nn{sref}{URI: \l__stex_refs_uri_str?#1}
1297 %\msg_warning:nnn{stex}{warning/smsmissing}{<filename>}
1298 \begingroup\catcode13=9\relax\catcode10=9\relax
1299 \str_if_empty:NTF \l__stex_refs_repob_str {

```

```

1300 \prop_if_exist:NTF \l_stex_current_repository_prop {
1301   \str_set:Nx \l_tmpa_str {
1302     \c_stex_mathhub_str /
1303     \prop_item:Nn \l_stex_current_repository_prop { id }
1304     / source / \l__stex_refs_in_str .sref
1305   }
1306 }{
1307   \str_set:Nx \l_tmpa_str {
1308     \stex_path_to_string:N \g_stex_currentfile_seq/ .. / \l__stex_refs_in_str . sref
1309   }
1310 }
1311 }{
1312   \str_set:Nx \l_tmpa_str {
1313     \c_stex_mathhub_str / \l__stex_refs_repob_str
1314     / source / \l__stex_refs_in_str . sref
1315   }
1316 }
1317 \stex_path_from_string:Nn \l_tmpb_seq \l_tmpa_str
1318 \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
1319 \stex_debug:nn{sref}{File: \l_tmpa_str}
1320 \exp_args:No \IfFileExists \l_tmpa_str {
1321   \tl_clear:N \l__stex_refs_return_tl
1322   \str_set:Nn \l__stex_refs_id_str {#1}
1323   \let\STEXInternalSrefRestoreTarget\__stex_refs_restore_target:nnnnn
1324   \use:c{@ @ input}{\l_tmpa_str}
1325   \exp_args:No \tl_if_empty:nTF \l__stex_refs_return_tl {
1326     \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smslabelmissing}\l_tmpa_str{#1}
1327     \__stex_refs_do_autoref:n{
1328       \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1329     }
1330   }{
1331     \l__stex_refs_return_tl
1332   }
1333 }{
1334   \exp_args:Nnno \msg_warning:nnnn{stex}{warning/smsmissing}\l_tmpa_str
1335   \__stex_refs_do_autoref:n{
1336     \str_if_empty:NF\l__stex_refs_uri_str{\l__stex_refs_uri_str?}#1
1337   }
1338 }
1339 \endgroup
1340 }
1341
1342 % \__stex_refs_args:n { #1 }
1343 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1344 %   \str_set:Nx \l_tmpa_str { #2 }
1345 %   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1346 %   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1347 %     \seq_if_exist:CTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1348 %       \seq_get_left:cnF {g__stex_refs_labels_\l_tmpa_str_seq} \l_tmpa_str {
1349 %         \str_clear:N \l_tmpa_str
1350 %       }
1351 %     }{
1352 %       \str_clear:N \l_tmpa_str
1353 %     }

```

```

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

```

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

\srefsym

```

1403 \NewDocumentCommand \srefsym { 0{} m}{

```

```

1404 \stex_get_symbol:n { #2 }
1405 \__stex_refs_sym_aux:nn{#1}{\l_stex_get_symbol_uri_str}
1406 }
1407
1408 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1409
1410 % \str_if_exist:cTF {sref_sym_#2 _label_str }{
1411 % \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1412 % }{
1413 % \__stex_refs_args:n { #1 }
1414 % \str_if_empty:NTF \l__stex_refs_indocument_str {
1415 % \tl_if_exist:cTF{sref_sym_#2 _type}{
1416 % % doc uri in \l_tmpb_str
1417 % \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1418 % \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1419 % % reference
1420 % \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1421 % \cs_if_exist:cTF{autoref}{
1422 % \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1423 % }{
1424 % \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1425 % }
1426 % }{
1427 % \ltx@ifpackageloaded{hyperref}{
1428 % \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1429 % }{
1430 % \l__stex_refs_linktext_tl
1431 % }
1432 % }
1433 % }{
1434 % % URL
1435 % \ltx@ifpackageloaded{hyperref}{
1436 % \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl
1437 % }{
1438 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1439 % }
1440 % }
1441 % }{
1442 % \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_
1443 % }
1444 % }{
1445 % % TODO
1446 % }
1447 % }
1448 }

```

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

`\srefsymuri`

```

1449 \cs_new_protected:Npn \srefsymuri #1 #2 { % TODO
1450 #2%\__stex_refs_sym_aux:nn[linktext={#2}]{#1}
1451 }

```

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

```

1452 </package>

```

Chapter 27

STEX -Modules Implementation

```
1453 <*package>
1454
1455 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1456
1457 <@@=stex_modules>
1458
1459     Warnings and error messages
1458 \msg_new:nnn{stex}{error/unknownmodule}{
1459     No~module~#1~found
1460 }
1461 \msg_new:nnn{stex}{error/syntax}{
1462     Syntax~error:~#1
1463 }
1464 \msg_new:nnn{stex}{error/siglanguage}{
1465     Module~#1~declares~signature~#2,~but~does~not~
1466     declare~its~language
1467 }
1468 \msg_new:nnn{stex}{warning/deprecated}{
1469     #1~is~deprecated;~please~use~#2~instead!
1470 }
1471
1472 \msg_new:nnn{stex}{error/conflictingmodules}{
1473     Conflicting~imports~for~module~#1
1474 }
1475
\l_stex_current_module_str The current module:
1475 \str_new:N \l_stex_current_module_str
1476
(End definition for \l_stex_current_module_str. This variable is documented on page 79.)

\l_stex_all_modules_seq Stores all available modules
1476 \seq_new:N \l_stex_all_modules_seq
1477
(End definition for \l_stex_all_modules_seq. This variable is documented on page 79.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1477 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1478   \str_if_empty:NTF \l_stex_current_module_str
1479   \prg_return_false: \prg_return_true:
1480 }

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

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1481 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1482   \prop_if_exist:cTF { c_stex_module_#1_prop }
1483   \prg_return_true: \prg_return_false:
1484 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
Only allowed within modules:
1485 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1486   \stex_add_to_current_module:n { #1 }
1487   \stex_do_up_to_module:n { #1 }
1488 }}
1489 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1490
1491 \cs_new_protected:Nn \stex_add_to_current_module:n {
1492   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1493 }
1494 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1495 \cs_new_protected:Npn \STEXexport {
1496   \ExplSyntaxOn
1497   \__stex_modules_export:n
1498 }
1499 \cs_new_protected:Nn \__stex_modules_export:n {
1500   \ignorespacesandpars#1\ExplSyntaxOff
1501   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1502   \stex_smsmode_do:
1503 }
1504 \let \stex_module_export_helper:n \use:n
1505 \stex_deactivate_macro:Nn \STEXexport {module~environments}

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

```

```

\stex_add_constant_to_current_module:n
1506 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1507   \str_set:Nx \l_tmpa_str { #1 }
1508   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1509 }

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

```

```

\stex_add_import_to_current_module:n
1510 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1511   \str_set:Nx \l_tmpa_str { #1 }
1512   \exp_args:Nno

```

```

1513 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1514 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1515 }
1516 }

```

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

`\stex_collect_imports:n`

```

1517 \cs_new_protected:Nn \stex_collect_imports:n {
1518 \seq_clear:N \l_stex_collect_imports_seq
1519 \__stex_modules_collect_imports:n {#1}
1520 }
1521 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1522 \seq_map_inline:cn {c_stex_module_#1_imports} {
1523 \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1524 \__stex_modules_collect_imports:n { ##1 }
1525 }
1526 }
1527 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1528 \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1529 }
1530 }

```

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

`\stex_do_up_to_module:n`

```

1531 \int_new:N \l__stex_modules_group_depth_int
1532 \cs_new_protected:Nn \stex_do_up_to_module:n {
1533 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1534 #1
1535 }{
1536 #1
1537 \expandafter \tl_gset:Nn
1538 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1539 \expandafter\expandafter\expandafter\endcsname
1540 \expandafter\expandafter\expandafter { \csname
1541 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1542 \aftergroup\__stex_modules_aftergroup_do:
1543 }
1544 }
1545 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1546 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1547 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1548 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1549 }}}
1550 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1551 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1552 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1553 }{
1554 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1555 \aftergroup\__stex_modules_aftergroup_do:
1556 }
1557 }
1558 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1559 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1560 }

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

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

1561

(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.

```

1562 \str_new:N \l_stex_module_ns_str
1563 \str_new:N \l_stex_module_subpath_str
1564 \cs_new_protected:Nn __stex_modules_compute_namespace:nN {
1565   \seq_set_eq:NN \l_tmpa_seq #2
1566   % split off file extension
1567   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1568   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1569   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1570   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1571
1572   \bool_set_true:N \l_tmpa_bool
1573   \bool_while_do:Nn \l_tmpa_bool {
1574     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1575     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1576       {source} { \bool_set_false:N \l_tmpa_bool }
1577     }{}{
1578       \seq_if_empty:NT \l_tmpa_seq {
1579         \bool_set_false:N \l_tmpa_bool
1580       }
1581     }
1582   }
1583
1584   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1585   % \l_tmpa_seq <- sub-path relative to archive
1586   \str_if_empty:NTF \l_stex_module_subpath_str {
1587     \str_set:Nx \l_stex_module_ns_str {#1}
1588   }{
1589     \str_set:Nx \l_stex_module_ns_str {
1590       #1/\l_stex_module_subpath_str
1591     }
1592   }
1593 }
1594
1595 \cs_new_protected:Nn \stex_modules_current_namespace: {
1596   \str_clear:N \l_stex_module_subpath_str
1597   \prop_if_exist:NTF \l_stex_current_repository_prop {
1598     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1599     __stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1600   }{
1601     % split off file extension
1602     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1603     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str

```



```

1604 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1605 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1606 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1607 \str_set:Nx \l_stex_module_ns_str {
1608   file:/\stex_path_to_string:N \l_tmpa_seq
1609 }
1610 }
1611 }

```

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

27.1 The smodule environment

smodule arguments:

```

1612 \keys_define:nn { stex / module } {
1613   title      .tl_set:N      = \smodulename ,
1614   type       .str_set_x:N   = \smodulename ,
1615   id         .str_set_x:N   = \smoduleid ,
1616   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1617   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1618   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1619   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1620   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1621   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1622   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1623   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1624 }
1625
1626 \cs_new_protected:Nn \__stex_modules_args:n {
1627   \str_clear:N \smodulename
1628   \str_clear:N \smodulename
1629   \str_clear:N \smoduleid
1630   \str_clear:N \l_stex_module_ns_str
1631   \str_clear:N \l_stex_module_deprecate_str
1632   \str_clear:N \l_stex_module_lang_str
1633   \str_clear:N \l_stex_module_sig_str
1634   \str_clear:N \l_stex_module_creators_str
1635   \str_clear:N \l_stex_module_contributors_str
1636   \str_clear:N \l_stex_module_meta_str
1637   \str_clear:N \l_stex_module_srccite_str
1638   \keys_set:nn { stex / module } { #1 }
1639 }
1640
1641 % module parameters here? In the body?
1642

```

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

```

1643 \cs_new_protected:Nn \stex_module_setup:nn {
1644   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1645   \str_set:Nx \l_stex_module_name_str { #2 }
1646   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1647 \stex_if_in_module:TF {
1648   % Nested module
1649   \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1650   { ns } \l_stex_module_ns_str
1651   \str_set:Nx \l_stex_module_name_str {
1652     \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1653     { name } / \l_stex_module_name_str
1654   }
1655   \str_if_empty:NT \l_stex_module_lang_str {
1656     \str_set:Nx \l_stex_module_lang_str {
1657       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1658       { lang }
1659     }
1660   }
1661 }{
1662   % not nested:
1663   \str_if_empty:NT \l_stex_module_ns_str {
1664     \stex_modules_current_namespace:
1665     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1666       / {\l_stex_module_ns_str}
1667     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1668     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1669       \str_set:Nx \l_stex_module_ns_str {
1670         \stex_path_to_string:N \l_tmpa_seq
1671       }
1672     }
1673   }
1674 }

```

Next, we determine the language of the module:

```

1675 \str_if_empty:NT \l_stex_module_lang_str {
1676   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1677   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1678   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1679   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1680     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1681       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1682     }
1683   }
1684   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1685   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1686     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1687     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1688       inferred~from~file~name}
1689   }
1690 }
1691
1692 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1693   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1694 }}

```

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

```

1695 \str_if_empty:NTF \l_stex_module_sig_str {
1696   \exp_args:Nnx \prop_gset_from_keyval:cn {
1697     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1698   } {
1699     name      = \l_stex_module_name_str ,
1700     ns       = \l_stex_module_ns_str ,
1701     file     = \exp_not:o { \g_stex_currentfile_seq } ,
1702     lang     = \l_stex_module_lang_str ,
1703     sig      = \l_stex_module_sig_str ,
1704     deprecate = \l_stex_module_deprecate_str ,
1705     meta     = \l_stex_module_meta_str
1706   }
1707   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1708   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1709   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1710   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1711   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1712 \str_if_empty:NT \l_stex_module_meta_str {
1713   \str_set:Nx \l_stex_module_meta_str {
1714     \c_stex_metatheory_ns_str ? Metatheory
1715   }
1716 }
1717 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1718   \bool_set_true:N \l_stex_in_meta_bool
1719   \exp_args:Nx \stex_add_to_current_module:n {
1720     \bool_set_true:N \l_stex_in_meta_bool
1721     \stex_activate_module:n {\l_stex_module_meta_str}
1722     \bool_set_false:N \l_stex_in_meta_bool
1723   }
1724   \stex_activate_module:n {\l_stex_module_meta_str}
1725   \bool_set_false:N \l_stex_in_meta_bool
1726 }
1727 }{
1728   \str_if_empty:NT \l_stex_module_lang_str {
1729     \msg_error:nnxx{stex}{error/siglanguage}{
1730       \l_stex_module_ns_str?\l_stex_module_name_str
1731     }{\l_stex_module_sig_str}
1732   }
1733   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1734   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1735     \stex_debug:nn{modules}{(already exists)}
1736   }{
1737     \stex_debug:nn{modules}{(needs loading)}
1738     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1739     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1740     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1741     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1742     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1743     \str_set:Nx \l_tmpa_str {
1744       \stex_path_to_string:N \l_tmpa_seq /

```

```

1745     \l_tmpa_str . \l_stex_module_sig_str .tex
1746   }
1747   \IfFileExists \l_tmpa_str {
1748     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1749       \str_clear:N \l_stex_current_module_str
1750       \seq_clear:N \l_stex_all_modules_seq
1751       \stex_debug:nn{modules}{Loading~signature}
1752     }
1753   }{
1754     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1755   }
1756 }
1757 \stex_if_smsmode:F {
1758   \stex_activate_module:n {
1759     \l_stex_module_ns_str ? \l_stex_module_name_str
1760   }
1761 }
1762 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1763 }
1764 \str_if_empty:NF \l_stex_module_deprecate_str {
1765   \msg_warning:nnxx{stex}{warning/deprecated}{
1766     Module~\l_stex_current_module_str
1767   }{
1768     \l_stex_module_deprecate_str
1769   }
1770 }
1771 \seq_put_right:Nx \l_stex_all_modules_seq {
1772   \l_stex_module_ns_str ? \l_stex_module_name_str
1773 }
1774 \tl_clear:c{l__stex_modules_aftergroup_\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1775 }

```

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

smodule (*env.*) The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}
1776 \cs_new_protected:Nn \__stex_modules_begin_module: {
1777   \stex_reactivate_macro:N \STEXexport
1778   \stex_reactivate_macro:N \importmodule
1779   \stex_reactivate_macro:N \symdecl
1780   \stex_reactivate_macro:N \notation
1781   \stex_reactivate_macro:N \symdef
1782 }
1783 \stex_debug:nn{modules}{
1784   New~module:\\
1785   Namespace:~\l_stex_module_ns_str\\
1786   Name:~\l_stex_module_name_str\\
1787   Language:~\l_stex_module_lang_str\\
1788   Signature:~\l_stex_module_sig_str\\
1789   Metatheory:~\l_stex_module_meta_str\\
1790   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1791 }
1792

```

```

1793 \stex_if_do_html:T{
1794   \begin{stex_annotate_env} {theory} {
1795     \l_stex_module_ns_str ? \l_stex_module_name_str
1796   }
1797
1798   \stex_annotate_invisible:nnn{header}{} {
1799     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1800     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1801     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1802       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1803     }
1804     \str_if_empty:NF \smoduletype {
1805       \stex_annotate:nnn{type}{\smoduletype}{}
1806     }
1807   }
1808 }
1809 % TODO: Inherit metatheory for nested modules?
1810 }
1811 \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}`

```

1812 \cs_new_protected:Nn \_stex_modules_end_module: {
1813   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1814   \stex_reset_up_to_module:n \l_stex_current_module_str
1815   \stex_if_smsmode:T {
1816     \stex_persist:x {
1817       \prop_set_from_keyval:cn{c_stex_module\_l_stex_current_module_str _prop}{
1818         \exp_after:wN \prop_to_keyval:N \csname c_stex_module\_l_stex_current_module_str _pr
1819       }
1820       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _constants}{
1821         \seq_use:cn{c_stex_module\_l_stex_current_module_str _constants},
1822       }
1823       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _imports}{
1824         \seq_use:cn{c_stex_module\_l_stex_current_module_str _imports},
1825       }
1826       \tl_set:cn {c_stex_module\_l_stex_current_module_str _code}
1827     }
1828     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module\_l_stex_current_module
1829     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1830   }
1831 }

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1832 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1833 \NewDocumentEnvironment { smodule } { 0 } { m } {
1834   \stex_module_setup:nn{#1}{#2}
1835   %\par
1836   \stex_if_smsmode:F{
1837     \tl_if_empty:NF \smoduletitle {
1838       \exp_args:No \stex_document_title:n \smoduletitle
1839     }

```

```

1840 \tl_clear:N \l_tmpa_tl
1841 \clist_map_inline:Nn \smodulotype {
1842   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1843     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1844   }
1845 }
1846 \tl_if_empty:NTF \l_tmpa_tl {
1847   \__stex_modules_smodule_start:
1848 }{
1849   \l_tmpa_tl
1850 }
1851 }
1852 \__stex_modules_begin_module:
1853 \str_if_empty:NF \smoduleid {
1854   \stex_ref_new_doc_target:n \smoduleid
1855 }
1856 \stex_smsmode_do:
1857 } {
1858   \__stex_modules_end_module:
1859   \stex_if_smsmode:F {
1860     \end{stex_annotate_env}
1861     \clist_set:Nn \l_tmpa_clist \smodulotype
1862     \tl_clear:N \l_tmpa_tl
1863     \clist_map_inline:Nn \l_tmpa_clist {
1864       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1865         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1866       }
1867     }
1868     \tl_if_empty:NTF \l_tmpa_tl {
1869       \__stex_modules_smodule_end:
1870     }{
1871       \l_tmpa_tl
1872     }
1873   }
1874 }

```

\stexpatchmodule

```

1875 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1876 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1877
1878 \newcommand\stexpatchmodule[3] [] {
1879   \str_set:Nx \l_tmpa_str{ #1 }
1880   \str_if_empty:NTF \l_tmpa_str {
1881     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1882     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1883   }{
1884     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1885     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1886   }
1887 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1888 \NewDocumentCommand \STEXModule { m } {
1889   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1890   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1891   \tl_set:Nn \l_tmpa_tl {
1892     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1893   }
1894   \seq_map_inline:Nn \l_stex_all_modules_seq {
1895     \str_set:Nn \l_tmpb_str { ##1 }
1896     \str_if_eq:eeT { \l_tmpa_str } {
1897       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1898     } {
1899       \seq_map_break:n {
1900         \tl_set:Nn \l_tmpa_tl {
1901           \stex_invoke_module:n { ##1 }
1902         }
1903       }
1904     }
1905   }
1906   \l_tmpa_tl
1907 }
1908
1909 \cs_new_protected:Nn \stex_invoke_module:n {
1910   \stex_debug:nn{modules}{Invoking~module~#1}
1911   \peek_charcode_remove:NTF ! {
1912     \__stex_modules_invoke_uri:nN { #1 }
1913   } {
1914     \peek_charcode_remove:NTF ? {
1915       \__stex_modules_invoke_symbol:nn { #1 }
1916     } {
1917       \msg_error:nnx{stex}{error/syntax}{
1918         ?~or~!~expected~after~
1919         \c_backslash_str STEXModule{#1}
1920       }
1921     }
1922   }
1923 }
1924
1925 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1926   \str_set:Nn #2 { #1 }
1927 }
1928
1929 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1930   \stex_invoke_symbol:n{#1?#2}
1931 }

```

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

```

\stex_activate_module:n
1932 \bool_new:N \l_stex_in_meta_bool
1933 \bool_set_false:N \l_stex_in_meta_bool

```

```

1934 \cs_new_protected:Nn \stex_activate_module:n {
1935   \stex_debug:nn{modules}{Activating~module~#1}
1936   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1937     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1938     \use:c{ c_stex_module_#1_code }
1939   }
1940 }

```

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

`mmtinterface` (*env.*)

```

1941 \NewDocumentEnvironment { mmtinterface } { 0{} m m } {
1942   \begin{smodule}[#1]{#3}
1943   \str_set:Nx \l_stex_module_mmtfor_str {#2}
1944   \MMTinclude{#2}
1945   \stex_reactivate_macro:N \mmtdecl
1946   \stex_reactivate_macro:N \mmtdef
1947 }{
1948   \end{smodule}
1949 }

1950 \</package>

```


Chapter 28

STEX -Module Inheritance Implementation

```
1951 <*package>
1952
1953 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1954
```

28.1 SMS Mode

```
1955 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1956 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1957 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1958 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1959
1960 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1961   \makeatletter
1962   \makeatother
1963   \ExplSyntaxOn
1964   \ExplSyntaxOff
1965   \rustexBREAK
1966 }
1967
1968 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1969   \symdef
1970   \importmodule
1971   \notation
1972   \symdecl
1973   \STEXexport
1974   \inlineass
1975   \inlinedef
1976   \inlineex
1977   \endinput
1978   \setnotation
```

```

1979 \copynotation
1980 \assign
1981 \renamedekl
1982 \donotcopy
1983 \instantiate
1984 \textsymdecl
1985 }
1986
1987 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1988   \tl_to_str:n {
1989     smodule,
1990     copymodule,
1991     interpretmodule,
1992     realization,
1993     sdefinition,
1994     sexample,
1995     sassertion,
1996     sparagraph,
1997     mathstructure
1998   }
1999 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 82.)

`\stex_if_smsmode_p:`

```

\stex_if_smsmode:TF
2000 \bool_new:N \g__stex_smsmode_bool
2001 \bool_set_false:N \g__stex_smsmode_bool
2002 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
2003   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
2004 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

2005 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
2006   \vbox_set:Nn \l_tmpa_box {
2007     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
2008     \bool_gset_true:N \g__stex_smsmode_bool
2009     #2
2010     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
2011   }
2012   \box_clear:N \l_tmpa_box
2013 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

2014 \quark_new:N \q__stex_smsmode_break
2015
2016 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
2017   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
2018   \stex_smsmode_do:
2019 }
2020

```

```

2021 \cs_new_protected:Nn \__stex_smsmode_module:nn {
2022   \__stex_modules_args:n{#1}
2023   \stex_if_in_module:F {
2024     \str_if_empty:NF \l_stex_module_sig_str {
2025       \stex_modules_current_namespace:
2026       \str_set:Nx \l_stex_module_name_str { #2 }
2027       \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
2028         \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2029         \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2030         \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2031         \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
2032         \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
2033         \str_set:Nx \l_tmpa_str {
2034           \stex_path_to_string:N \l_tmpa_seq /
2035           \l_tmpa_str . \l_stex_module_sig_str .tex
2036         }
2037         \IfFileExists \l_tmpa_str {
2038           \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
2039         }{
2040           \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
2041         }
2042       }
2043     }
2044   }
2045 }
2046
2047 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
2048   %\stex_debug:nn{import-pair}{\detokenize{#1}~{#2}}
2049   \tl_if_empty:nTF{#1}{
2050     \prop_if_exist:NTF \l_stex_current_repository_prop
2051     {
2052       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
2053       \prg_return_true:
2054     } {
2055       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
2056       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2057       \tl_if_empty:NT \l_tmpa_tl {
2058         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
2059       }
2060       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
2061       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
2062       \prg_return_true: \prg_return_false:
2063     }
2064   }\prg_return_true:
2065 }
2066
2067 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
2068   \stex_filestack_push:n{#1}
2069   \seq_gclear:N \l__stex_smsmode_importmodules_seq
2070   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
2071   % ----- new -----
2072   \__stex_smsmode_in_smsmode:nn{#1}{
2073     \let\importmodule\__stex_smsmode_importmodule:
2074     \let\stex_module_setup:nn\__stex_smsmode_module:nn

```

```

2075 \let\__stex_modules_begin_module:\relax
2076 \let\__stex_modules_end_module:\relax
2077 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
2078 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
2079 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
2080 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
2081 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
2082 \everyeof{\q__stex_smsmode_break\noexpand}
2083 \expandafter\expandafter\expandafter
2084 \stex_smsmode_do:
2085 \csname @ @ input\endcsname "#1"\relax
2086
2087 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
2088   \stex_filestack_push:n{##1}
2089   \expandafter\expandafter\expandafter
2090   \stex_smsmode_do:
2091   \csname @ @ input\endcsname "##1"\relax
2092   \stex_filestack_pop:
2093 }
2094 }
2095 % ----- new -----
2096 \__stex_smsmode_in_smsmode:nn{#1} {
2097   #2
2098   % ----- new -----
2099   \begingroup
2100   \%stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
2101   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
2102     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
2103       \stex_import_module_uri:nn ##1
2104       \stex_import_require_module:nnnn
2105       \l_stex_import_ns_str
2106       \l_stex_import_archive_str
2107       \l_stex_import_path_str
2108       \l_stex_import_name_str \endgroup
2109     }
2110   }
2111   \endgroup
2112   \%stex_debug:nn{smsmode}{Actually~loading~file~#1}
2113   % ----- new -----
2114   \everyeof{\q__stex_smsmode_break\noexpand}
2115   \expandafter\expandafter\expandafter
2116   \stex_smsmode_do:
2117   \csname @ @ input\endcsname "#1"\relax
2118 }
2119 \stex_filestack_pop:
2120 }

```

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

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

```

2121 \cs_new_protected:Npn \stex_smsmode_do: {
2122   \stex_if_smsmode:T {
2123     \__stex_smsmode_do:w

```

```

2124 }
2125 }
2126 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
2127   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
2128     \expandafter\if\expandafter\relax\noexpand#1
2129     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
2130   \else\expandafter\__stex_smsmode_do:w\fi
2131 }{
2132   \__stex_smsmode_do:w % #1
2133 }
2134 }
2135 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
2136   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
2137     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
2138       #1\__stex_smsmode_do:w
2139     }{
2140       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
2141         #1
2142       }{
2143         \cs_if_eq:NNTF \begin #1 {
2144           \__stex_smsmode_check_begin:n
2145         }{
2146           \cs_if_eq:NNTF \end #1 {
2147             \__stex_smsmode_check_end:n
2148           }{
2149             \__stex_smsmode_do:w
2150           }
2151         }
2152       }
2153     }
2154   }
2155 }
2156
2157 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
2158   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2159     \begin{#1}
2160   }{
2161     \__stex_smsmode_do:w
2162   }
2163 }
2164 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
2165   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
2166     \end{#1}\__stex_smsmode_do:w
2167   }{
2168     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
2169   }
2170 }

```

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

28.2 Inheritance

```

2171 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```
2172 \cs_new_protected:Nn \stex_import_module_uri:nn {
2173   \str_set:Nx \l_stex_import_archive_str { #1 }
2174   \str_set:Nn \l_stex_import_path_str { #2 }
2175
2176   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
2177   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
2178   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
2179
2180   \stex_modules_current_namespace:
2181   \bool_lazy_all:nTF {
2182     {\str_if_empty_p:N \l_stex_import_archive_str}
2183     {\str_if_empty_p:N \l_stex_import_path_str}
2184     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
2185   }{
2186     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
2187     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
2188   }{
2189     \str_if_empty:NT \l_stex_import_archive_str {
2190       \prop_if_exist:NT \l_stex_current_repository_prop {
2191         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
2192       }
2193     }
2194     \str_if_empty:NTF \l_stex_import_archive_str {
2195       \str_if_empty:NF \l_stex_import_path_str {
2196         \stex_path_from_string:Nn \l_tmpb_seq {
2197           \l_stex_module_ns_str / .. / \l_stex_import_path_str
2198         }
2199         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
2200         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file:///}
2201       }
2202     }{
2203       \stex_require_repository:n \l_stex_import_archive_str
2204       \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str_manifest_prop } { ns }
2205       \l_stex_import_ns_str
2206       \str_if_empty:NF \l_stex_import_path_str {
2207         \str_set:Nx \l_stex_import_ns_str {
2208           \l_stex_import_ns_str / \l_stex_import_path_str
2209         }
2210       }
2211     }
2212   }
2213 }
```

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

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

Store the return values of `\stex_import_module_uri:nn`.

```
2214 \str_new:N \l_stex_import_name_str
2215 \str_new:N \l_stex_import_archive_str
2216 \str_new:N \l_stex_import_path_str
2217 \str_new:N \l_stex_import_ns_str
```

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

```

\stex_import_require_module:nnnnn {{\ns}} {{\archive-ID}} {{\path}} {{\name}}
2218 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
2219   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
2220
2221     \stex_debug:nn{requiremodule}{Here:\~1:\~2:\~3:\~4}
2222
2223     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
2224     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
2225
2226     %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
2227
2228     % archive
2229     \str_set:Nx \l_tmpa_str { #2 }
2230     \str_if_empty:NTF \l_tmpa_str {
2231       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2232       \seq_put_right:Nn \l_tmpa_seq {...}
2233     } {
2234       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
2235       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
2236       \seq_put_right:Nn \l_tmpa_seq { source }
2237     }
2238
2239     % path
2240     \str_set:Nx \l_tmpb_str { #3 }
2241     \str_if_empty:NTF \l_tmpb_str {
2242       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
2243
2244       \ltx@ifpackageloaded{babel} {
2245         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2246           { \language } \l_tmpb_str {
2247           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2248         }
2249       } {
2250         \str_clear:N \l_tmpb_str
2251       }
2252
2253       \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2254       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2255         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2256       }{
2257         \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2258         \IfFileExists{ \l_tmpa_str.tex }{
2259           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2260         }{
2261           % try english as default
2262           \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2263           \IfFileExists{ \l_tmpa_str.en.tex }{
2264             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2265           }{
2266             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2267           }
2268         }
2269       }
2270

```

```

2271 } {
2272   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
2273   \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2274
2275   \ltx@ifpackageloaded{babel} {
2276     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
2277       { \language } \l_tmpb_str {
2278       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2279     }
2280   } {
2281     \str_clear:N \l_tmpb_str
2282   }
2283
2284   \stex_path_canonicalize:N \l_tmpb_seq
2285   \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2286
2287   \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2288   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2289     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2290   }{
2291     \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2292     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2293       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2294     }{
2295       % try english as default
2296       \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2297       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2298         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2299       }{
2300         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2301         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2302           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2303         }{
2304           \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2305           \IfFileExists{ \l_tmpa_str.tex }{
2306             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2307           }{
2308             % try english as default
2309             \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2310             \IfFileExists{ \l_tmpa_str.en.tex }{
2311               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2312             }{
2313               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2314             }
2315           }
2316         }
2317       }
2318     }
2319   }
2320 }
2321
2322 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2323   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2324     \seq_clear:N \l_stex_all_modules_seq

```



```

2325     \str_clear:N \l_stex_current_module_str
2326     \str_set:Nx \l_tmpb_str { #2 }
2327     \str_if_empty:NF \l_tmpb_str {
2328       \stex_set_current_repository:n { #2 }
2329     }
2330     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2331   }
2332
2333   \stex_if_module_exists:nF { #1 ? #4 } {
2334     \msg_error:nnx{stex}{error/unknownmodule}{
2335       #1?#4~(in~file~\g__stex_importmodule_file_str)
2336     }
2337   }
2338 }
2339
2340 }
2341 \stex_activate_module:n { #1 ? #4 }
2342 }

```

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

`\importmodule`

```

2343 \NewDocumentCommand \importmodule { 0{} m } {
2344   \stex_import_module_uri:nn { #1 } { #2 }
2345   \stex_debug:nn{modules}{Importing~module:~
2346     \l_stex_import_ns_str ? \l_stex_import_name_str
2347   }
2348   \stex_import_require_module:nnnn
2349   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2350   { \l_stex_import_path_str } { \l_stex_import_name_str }
2351   \stex_if_smsmode:F {
2352     \stex_annotate_invisible:nnn
2353     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2354   }
2355   \exp_args:Nx \stex_add_to_current_module:n {
2356     \stex_import_require_module:nnnn
2357     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2358     { \l_stex_import_path_str } { \l_stex_import_name_str }
2359   }
2360   \exp_args:Nx \stex_add_import_to_current_module:n {
2361     \l_stex_import_ns_str ? \l_stex_import_name_str
2362   }
2363   \stex_smsmode_do:
2364   \ignorespacesandpars
2365 }
2366 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

`\usemodule`

```

2367 \NewDocumentCommand \usemodule { 0{} m } {
2368   \stex_if_smsmode:F {
2369     \stex_import_module_uri:nn { #1 } { #2 }
2370     \stex_import_require_module:nnnn
2371     { \l_stex_import_ns_str } { \l_stex_import_archive_str }

```

```

2372 { \l_stex_import_path_str } { \l_stex_import_name_str }
2373 \stex_annotate_invisible:nnn
2374 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2375 }
2376 \stex_smsmode_do:
2377 \ignorespacesandpars
2378 }

```

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

```

2379 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2380   \tl_if_empty:nF{#2}{
2381     \clist_set:Nn \l_tmpa_clist {#2}
2382     \clist_map_inline:Nn \l_tmpa_clist {
2383       \tl_if_head_eq_charcode:nNTF {##1} [{
2384         #1 ##1
2385       } {
2386         #1{##1}
2387       }
2388     }
2389   }
2390 }
2391 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2392
2393
2394 \end{package}

```

Chapter 29

STEX -Symbols Implementation

```
2395 <*package>
2396
2397 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
2398
2399 \msg_new:nnn{stex}{error/wrongargs}{
2400   args~value~in~symbol~declaration~for~#1~
2401   needs~to~be~i,~a,~b~or~B,~but~#2~given
2402 }
2403 \msg_new:nnn{stex}{error/unknownsymbol}{
2404   No~symbol~#1~found!
2405 }
2406 \msg_new:nnn{stex}{error/seqlength}{
2407   Expected~#1~arguments;~got~#2!
2408 }
2409 \msg_new:nnn{stex}{error/unknownnotation}{
2410   Unknown~notation~#1~for~#2!
2411 }
```

29.1 Symbol Declarations

```
2412 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2413 \cs_new_protected:Nn \stex_all_symbols:n {
2414   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2415   \seq_map_inline:Nn \l_stex_all_modules_seq {
2416     \seq_map_inline:cn{c_stex_module_##1_constants}{
2417       \__stex_symdecl_all_symbols_cs{##1?####1}
2418     }
2419   }
2420 }
```

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

\STEXsymbol

```
2421 \NewDocumentCommand \STEXsymbol { m } {  
2422   \stex_get_symbol:n { #1 }  
2423   \exp_args:No  
2424   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
2425 }
```

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

symdecl arguments:

```
2426 \keys_define:nn { stex / symdecl } {  
2427   name      .str_set_x:N = \l_stex_symdecl_name_str ,  
2428   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
2429   args      .str_set_x:N = \l_stex_symdecl_args_str ,  
2430   type      .tl_set:N = \l_stex_symdecl_type_tl ,  
2431   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,  
2432   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)  
2433   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)  
2434   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
2435   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,  
2436   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,  
2437   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,  
2438   assoc     .choices:nn =  
2439     {bin,binl,binr,pre,conj,pwconj}  
2440     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}  
2441 }  
2442  
2443 \bool_new:N \l_stex_symdecl_make_macro_bool  
2444  
2445 \cs_new_protected:Nn \__stex_symdecl_args:n {  
2446   \str_clear:N \l_stex_symdecl_name_str  
2447   \str_clear:N \l_stex_symdecl_args_str  
2448   \str_clear:N \l_stex_symdecl_deprecate_str  
2449   \str_clear:N \l_stex_symdecl_reorder_str  
2450   \str_clear:N \l_stex_symdecl_assoctype_str  
2451   \bool_set_false:N \l_stex_symdecl_local_bool  
2452   \tl_clear:N \l_stex_symdecl_type_tl  
2453   \tl_clear:N \l_stex_symdecl_definiens_tl  
2454   \clist_clear:N \l_stex_symdecl_argnames_clist  
2455  
2456   \keys_set:nn { stex / symdecl } { #1 }  
2457 }
```

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

```
2458  
2459 \NewDocumentCommand \symdecl { s m O{} } {  
2460   \__stex_symdecl_args:n { #3 }  
2461   \IfBooleanTF #1 {  
2462     \bool_set_false:N \l_stex_symdecl_make_macro_bool  
2463   } {  
2464     \bool_set_true:N \l_stex_symdecl_make_macro_bool  
2465   }  
2466   \stex_symdecl_do:n { #2 }
```

```

2467 \stex_smsmode_do:
2468 }
2469
2470 \cs_new_protected:Nn \stex_symdecl_do:nn {
2471   \__stex_symdecl_args:n{#1}
2472   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2473   \stex_symdecl_do:n{#2}
2474 }
2475
2476 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2477 \cs_new_protected:Nn \stex_symdecl_do:n {
2478   \stex_if_in_module:F {
2479     % TODO throw error? some default namespace?
2480   }
2481
2482   \str_if_empty:NT \l_stex_symdecl_name_str {
2483     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2484   }
2485
2486   \prop_if_exist:cT { \l_stex_symdecl_
2487     \l_stex_current_module_str ?
2488     \l_stex_symdecl_name_str
2489     _prop
2490   }{
2491     % TODO throw error (beware of circular dependencies)
2492   }
2493
2494   \prop_clear:N \l_tmpa_prop
2495   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2496   \seq_clear:N \l_tmpa_seq
2497   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2498   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2499
2500   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2501     \str_if_empty:NF \l_stex_module_deprecate_str {
2502       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2503     }
2504   }
2505   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2506
2507   \exp_args:No \stex_add_constant_to_current_module:n {
2508     \l_stex_symdecl_name_str
2509   }
2510
2511   % arity/args
2512   \int_zero:N \l_tmpb_int
2513
2514   \bool_set_true:N \l_tmpa_bool
2515   \str_map_inline:Nn \l_stex_symdecl_args_str {
2516     \token_case_meaning:NnF ##1 {

```

```

2517 0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2518 {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2519 {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2520 {\tl_to_str:n a} {
2521   \bool_set_false:N \l_tmpa_bool
2522   \int_incr:N \l_tmpb_int
2523 }
2524 {\tl_to_str:n B} {
2525   \bool_set_false:N \l_tmpa_bool
2526   \int_incr:N \l_tmpb_int
2527 }
2528 }{
2529   \msg_error:nnxx{stex}{error/wrongargs}{
2530     \l_stex_current_module_str ?
2531     \l_stex_symdecl_name_str
2532   }{##1}
2533 }
2534 }
2535
2536 \bool_if:NTF \l_tmpa_bool {
2537   % possibly numeric
2538   \str_if_empty:NTF \l_stex_symdecl_args_str {
2539     \prop_put:Nnn \l_tmpa_prop { args } {}
2540     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2541   }{
2542     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2543     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2544     \str_clear:N \l_tmpa_str
2545     \int_step_inline:nn \l_tmpa_int {
2546       \str_put_right:Nn \l_tmpa_str i
2547     }
2548     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2549   }
2550 } {
2551   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2552   \prop_put:Nnx \l_tmpa_prop { arity }
2553     { \str_count:N \l_stex_symdecl_args_str }
2554 }
2555 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2556
2557 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2558   \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2559 }{
2560   \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2561 }
2562
2563 % argnames
2564
2565 \clist_clear:N \l_tmpa_clist
2566 \int_step_inline:nn {\prop_item:Nn \l_tmpa_prop {arity}} {
2567   \clist_if_empty:NTF \l_stex_symdecl_argnames_clist {
2568     \clist_put_right:Nn \l_tmpa_clist {##1}
2569   }{
2570     \clist_pop:NN \l_stex_symdecl_argnames_clist \l_tmpa_tl

```

```

2571     \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist {\c_dollar_str\l_tmpa_tl}
2572   }
2573 }
2574 \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
2575
2576 % semantic macro
2577
2578 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2579   \exp_args:Nx \stex_do_up_to_module:n {
2580     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2581       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2582     }}
2583   }
2584 }
2585
2586 \stex_debug:nn{symbols}{New~symbol:~
2587   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2588   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2589   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2590   Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2591 }
2592
2593 % circular dependencies require this:
2594 \stex_if_do_html:T {
2595   \stex_annotate_invisible:nnn {symdecl} {
2596     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2597   } {
2598     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2599       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2600     }
2601     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{}
2602     \stex_annotate_invisible:nnn{macroname}{#1}{}
2603     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2604       \stex_annotate_invisible:nnn{definiens}{}
2605       {\l_stex_symdecl_definiens_tl$}
2606     }
2607     \str_if_empty:NF \l_stex_symdecl_assoctype_str {
2608       \stex_annotate_invisible:nnn{assoctype}{\l_stex_symdecl_assoctype_str}{}
2609     }
2610     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2611       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2612     }
2613   }
2614 }
2615 \prop_if_exist:cF {
2616   l_stex_symdecl_
2617   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2618   _prop
2619 } {
2620   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2621     \_stex_symdecl_restore_symbol:nnnnnnn
2622     {\l_stex_symdecl_name_str}
2623     { \prop_item:Nn \l_tmpa_prop {args} }
2624     { \prop_item:Nn \l_tmpa_prop {arity} }

```

```

2625     { \prop_item:Nn \l_tmpa_prop {assocs} }
2626     { \prop_item:Nn \l_tmpa_prop {defined} }
2627     {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2628     {\l_stex_current_module_str}
2629     { \prop_item:Nn \l_tmpa_prop {argnames} }
2630   }
2631 }
2632 }
2633 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnnn {
2634   \prop_clear:N \l_tmpa_prop
2635   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2636   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2637   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2638   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2639   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2640   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2641   \prop_put:Nnn \l_tmpa_prop { argnames } { #8 }
2642   \tl_if_empty:nF{#6}{
2643     \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2644   }
2645   \prop_set_eq:cN{\l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2646   \seq_clear:c{\l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2647 }

```

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

\textsymdecl

```

2648
2649 \keys_define:nn { stex / textsymdecl } {
2650   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2651   type      .tl_set:N    = \l__stex_symdecl_type_tl
2652 }
2653
2654 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2655   \str_clear:N \l__stex_symdecl_name_str
2656   \tl_clear:N \l__stex_symdecl_type_tl
2657   \clist_clear:N \l_stex_symdecl_argnames_clist
2658   \keys_set:nn { stex / textsymdecl } { #1 }
2659 }
2660
2661 \NewDocumentCommand \textsymdecl {m O{} m} {
2662   \_stex_textsymdecl_args:n { #2 }
2663   \str_if_empty:NTF \l__stex_symdecl_name_str {
2664     \__stex_symdecl_args:n{name=#1,#2}
2665   }{
2666     \__stex_symdecl_args:n{#2}
2667   }
2668   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2669   \stex_symdecl_do:n{#1-sym}
2670   \stex_execute_in_module:n{
2671     \cs_set_nopar:cpn{#1name}{
2672       \ifvmode\hbox_unpack:N\c_empty_box\fi
2673       \ifmmode\hbox{#3}\else#3\fi\xspace
2674     }

```



```

2675 \cs_set_nopar:cpn{#1}{
2676   \ifmmode\csname#1-sym\expandafter\endcsname\else
2677   \ifvmode\hbox_unpack:N\c_empty_box\fi
2678   \symref{#1-sym}{#3}\expandafter\xspace
2679   \fi
2680 }
2681 }
2682 \stex_execute_in_module:x{
2683   \__stex_notation_restore_notation:nnnnn
2684   {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl_name_str{#1}}{0}
2685   {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfpref}{\neginfpref}}
2686   \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2687 }
2688 }
2689 {}
2690 }
2691 \stex_smsmode_do:
2692 }

```

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

\stex_get_symbol:n

```

2693 \str_new:N \l_stex_get_symbol_uri_str
2694
2695 \cs_new_protected:Nn \stex_get_symbol:n {
2696   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2697     \tl_set:Nn \l_tmpa_tl { #1 }
2698     \__stex_symdecl_get_symbol_from_cs:
2699   }{
2700     % argument is a string
2701     % is it a command name?
2702     \cs_if_exist:cTF { #1 }{
2703       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2704       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2705       \str_if_empty:NTF \l_tmpa_str {
2706         \exp_args:Nx \cs_if_eq:NTF {
2707           \tl_head:N \l_tmpa_tl
2708         } \stex_invoke_symbol:n {
2709           \__stex_symdecl_get_symbol_from_cs:
2710         }{
2711           \__stex_symdecl_get_symbol_from_string:n { #1 }
2712         }
2713       } {
2714         \__stex_symdecl_get_symbol_from_string:n { #1 }
2715       }
2716     }{
2717       % argument is not a command name
2718       \__stex_symdecl_get_symbol_from_string:n { #1 }
2719       % \l_stex_all_symbols_seq
2720     }
2721   }
2722   \str_if_eq:eeF {
2723     \prop_item:cn {
2724       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop

```

```

2725     }{ deprecate }
2726   }{}{
2727     \msg_warning:nnxx{stex}{warning/deprecated}{
2728       Symbol~\l_stex_get_symbol_uri_str
2729     }{
2730       \prop_item:cn {l_stex_symdecl_l\l_stex_get_symbol_uri_str _prop}{ deprecate }
2731     }
2732   }
2733 }
2734
2735 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2736   \tl_set:Nn \l_tmpa_tl {
2737     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2738   }
2739   \str_set:Nn \l_tmpa_str { #1 }
2740
2741   %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2742
2743   \str_if_in:NnTF \l_tmpa_str ? {
2744     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2745     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2746     \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2747   }{
2748     \str_clear:N \l_tmpb_str
2749   }
2750   \str_if_empty:NNTF \l_tmpb_str {
2751     \seq_map_inline:Nn \l_stex_all_modules_seq {
2752       \seq_map_inline:cn{c_stex_module_##1_constants}{
2753         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2754           \seq_map_break:n{\seq_map_break:n{
2755             \tl_set:Nn \l_tmpa_tl {
2756               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2757             }
2758           }}
2759         }
2760       }
2761     }
2762   }{
2763     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2764     \seq_map_inline:Nn \l_stex_all_modules_seq {
2765       \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2766         \seq_map_inline:cn{c_stex_module_##1_constants}{
2767           \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2768             \seq_map_break:n{\seq_map_break:n{
2769               \tl_set:Nn \l_tmpa_tl {
2770                 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2771               }
2772             }}
2773           }
2774         }
2775       }
2776     }
2777   }
2778

```

```

2779 \l_tmpa_tl
2780 }
2781
2782 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2783   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2784     { \tl_tail:N \l_tmpa_tl }
2785   \tl_if_single:NTF \l_tmpa_tl {
2786     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2787       \exp_after:wN \str_set:Nn \exp_after:wN
2788         \l_stex_get_symbol_uri_str \l_tmpa_tl
2789     }{
2790       % TODO
2791       % tail is not a single group
2792     }
2793   }{
2794     % TODO
2795     % tail is not a single group
2796   }
2797 }

```

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

29.2 Notations

```

2798 <@@=stex_notation>
      notation arguments:
2799 \keys_define:nn { stex / notation } {
2800   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2801   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2802   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2803   op       .tl_set:N = \l__stex_notation_op_tl ,
2804   primary .bool_set:N = \l__stex_notation_primary_bool ,
2805   primary .default:n = {true} ,
2806   hints    .str_set_x:N = \l__stex_notation_hints_str,
2807   unknown .code:n = \str_set:Nx
2808     \l__stex_notation_variant_str \l_keys_key_str
2809 }
2810
2811 \cs_new_protected:Nn \_stex_notation_args:n {
2812   % \str_clear:N \l__stex_notation_lang_str
2813   \str_clear:N \l__stex_notation_variant_str
2814   \str_clear:N \l__stex_notation_prec_str
2815   \str_clear:N \l__stex_notation_hints_str
2816   \tl_clear:N \l__stex_notation_op_tl
2817   \bool_set_false:N \l__stex_notation_primary_bool
2818
2819   \keys_set:nn { stex / notation } { #1 }
2820 }

```

\notation

```

2821 \NewDocumentCommand \notation { s m O{}} {
2822   \_stex_notation_args:n { #3 }
2823   \tl_clear:N \l_stex_symdecl_definiens_tl

```

```

2824 \stex_get_symbol:n { #2 }
2825 \tl_set:Nn \l_stex_notation_after_do_tl {
2826   \__stex_notation_final:
2827   \IfBooleanTF#1{
2828     \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2829   }{}
2830   \stex_smsmode_do:\ignorespacesandpars
2831 }
2832 \stex_notation_do:nnnnn
2833 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2834 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2835 { \l__stex_notation_variant_str }
2836 { \l__stex_notation_prec_str }
2837 }
2838 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nnnnn

```

2839 \seq_new:N \l__stex_notation_precedences_seq
2840 \tl_new:N \l__stex_notation_opprec_tl
2841 \int_new:N \l__stex_notation_currarg_int
2842 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2843
2844 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2845   \let\STEXInternalCurrentSymbolStr\relax
2846   \seq_clear:N \l__stex_notation_precedences_seq
2847   \tl_clear:N \l__stex_notation_opprec_tl
2848   \str_set:Nx \l__stex_notation_args_str { #1 }
2849   \str_set:Nx \l__stex_notation_arity_str { #2 }
2850   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2851   \str_set:Nx \l__stex_notation_prec_str { #4 }
2852
2853   % precedences
2854   \str_if_empty:NTF \l__stex_notation_prec_str {
2855     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2856       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2857     }{
2858       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2859     }
2860   } {
2861     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2862       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2863       \int_step_inline:nn { \l__stex_notation_arity_str } {
2864         \exp_args:NNo
2865         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2866       }
2867     }{
2868       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2869       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2870         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2871         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2872           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2873             \l_tmpa_seq {\tl_to_str:n{x}} } { \l_tmpa_str }

```

```

2874         \seq_map_inline:Nn \l_tmpa_seq {
2875             \seq_put_right:Nn \l__stex_notation_precedences_seq { ##1 }
2876         }
2877     }
2878 }{
2879     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2880         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2881     }{
2882         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2883     }
2884 }
2885 }
2886 }
2887
2888 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2889 \int_step_inline:nn { \l__stex_notation_arity_str } {
2890     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_seq {
2891         \exp_args:NNo
2892         \seq_put_right:No \l__stex_notation_precedences_seq {
2893             \l__stex_notation_opprec_tl
2894         }
2895     }
2896 }
2897 \tl_clear:N \l_stex_notation_dummyargs_tl
2898
2899 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2900     \exp_args:NNe
2901     \cs_set:Npn \l_stex_notation_macrocode_cs {
2902         \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2903         { \l__stex_notation_suffix_str }
2904         { \l__stex_notation_opprec_tl }
2905         { \exp_not:n { #5 } }
2906     }
2907     \l_stex_notation_after_do_tl
2908 }{
2909     \str_if_in:NnTF \l__stex_notation_args_str b {
2910         \exp_args:Nne \use:nn
2911         {
2912             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2913             \cs_set:Npn \l__stex_notation_arity_str } { {
2914                 \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2915                 { \l__stex_notation_suffix_str }
2916                 { \l__stex_notation_opprec_tl }
2917                 { \exp_not:n { #5 } }
2918             }}
2919     }{
2920         \str_if_in:NnTF \l__stex_notation_args_str B {
2921             \exp_args:Nne \use:nn
2922             {
2923                 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2924                 \cs_set:Npn \l__stex_notation_arity_str } { {
2925                     \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2926                     { \l__stex_notation_suffix_str }
2927                     { \l__stex_notation_opprec_tl }

```

```

2928         { \exp_not:n { #5 } }
2929     } }
2930 }{
2931     \exp_args:Nne \use:nn
2932     {
2933         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2934         \cs_set:Npn \l__stex_notation_arity_str } { {
2935             \STEXInternalTermMathOMAiiai { \STEXInternalCurrentSymbolStr }
2936             { \l__stex_notation_suffix_str }
2937             { \l__stex_notation_opprec_tl }
2938             { \exp_not:n { #5 } }
2939         } }
2940     }
2941 }
2942
2943 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2944 \int_zero:N \l__stex_notation_currarg_int
2945 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2946 \__stex_notation_arguments:
2947 }
2948 }

```

(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

```

2949 \cs_new_protected:Nn \__stex_notation_arguments: {
2950     \int_incr:N \l__stex_notation_currarg_int
2951     \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2952         \l_stex_notation_after_do_tl
2953     }{
2954         \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2955         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2956         \str_if_eq:NnTF \l_tmpa_str a {
2957             \__stex_notation_argument_assoc:nn{a}
2958         }{
2959             \str_if_eq:NnTF \l_tmpa_str B {
2960                 \__stex_notation_argument_assoc:nn{B}
2961             }{
2962                 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2963                 \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2964                     { \STEXInternalTermMathArgiii
2965                       { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2966                       { \l_tmpb_str }
2967                       { ###\int_use:N \l__stex_notation_currarg_int }
2968                     }
2969                 }
2970                 \__stex_notation_arguments:
2971             }
2972         }
2973     }
2974 }

```

(End definition for `__stex_notation_arguments:.`)

_stex_notation_argument_assoc:nn

```

2975 \cs_new_protected:Nn \_stex_notation_argument_assoc:nn {
2976
2977   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2978     {\l_stex_notation_arity_str}{
2979       #2
2980     }
2981   \int_zero:N \l_tmpa_int
2982   \tl_clear:N \l_tmpa_tl
2983   \str_map_inline:Nn \l_stex_notation_args_str {
2984     \int_incr:N \l_tmpa_int
2985     \tl_put_right:Nx \l_tmpa_tl {
2986       \str_if_eq:nnTF {##1}{a}{ {} }{
2987         \str_if_eq:nnTF {##1}{B}{ {} }{
2988           {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_int}
2989         }
2990       }
2991     }
2992   }
2993   \exp_after:wN\exp_after:wN\exp_after:wN \def
2994   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2995   \exp_after:wN\exp_after:wN\exp_after:wN ##
2996   \exp_after:wN\exp_after:wN\exp_after:wN 1
2997   \exp_after:wN\exp_after:wN\exp_after:wN ##
2998   \exp_after:wN\exp_after:wN\exp_after:wN 2
2999   \exp_after:wN\exp_after:wN\exp_after:wN {
3000     \exp_after:wN \exp_after:wN \exp_after:wN
3001     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
3002       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3003     }
3004   }
3005
3006   \seq_pop_left:NN \l_stex_notation_remaining_precs_seq \l_tmpa_str
3007   \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
3008     \STEXInternalTermMathAssocArgiiii
3009     { \int_use:N \l_stex_notation_currarg_int }
3010     { \l_tmpa_str }
3011     { ####\int_use:N \l_stex_notation_currarg_int }
3012     { \l_tmpa_cs {####1} {####2} }
3013     {#1}
3014   } }
3015   \_stex_notation_arguments:
3016 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

3017 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
3018   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}
3019   \cs_set_nopar:Npn {#3}{#4}
3020   \tl_if_empty:nF {#5}{
3021     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
3022   }
3023   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{

```

```

3024 \seq_put_right:cx { \l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
3025 }
3026 }
3027
3028 \cs_new_protected:Nn \__stex_notation_final: {
3029
3030 \stex_execute_in_module:x {
3031 \__stex_notation_restore_notation:nnnnn
3032 {\l_stex_get_symbol_uri_str}
3033 {\l__stex_notation_suffix_str}
3034 {\l__stex_notation_arity_str}
3035 {
3036 \exp_after:wN \exp_after:wN \exp_after:wN
3037 \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3038 { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
3039 }
3040 {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
3041 }
3042
3043 \stex_debug:nn{symbols}{
3044 Notation~\l__stex_notation_suffix_str
3045 ~for~\l_stex_get_symbol_uri_str^^J
3046 Operator~precedence:~\l__stex_notation_opprec_tl^^J
3047 Argument~precedences:~
3048 \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
3049 Notation: \cs_meaning:c {
3050 stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
3051 \l__stex_notation_suffix_str
3052 _cs
3053 }
3054 }
3055 % HTML annotations
3056 \stex_if_do_html:T {
3057 \stex_annotate_invisible:nnn { notation }
3058 { \l_stex_get_symbol_uri_str } {
3059 \stex_annotate_invisible:nnn { notationfragment }
3060 { \l__stex_notation_suffix_str }{}
3061 \stex_annotate_invisible:nnn { precedence }
3062 { \l__stex_notation_prec_str }{}
3063
3064 \int_zero:N \l_tmpa_int
3065 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
3066 \tl_clear:N \l_tmpa_tl
3067 \int_step_inline:nn { \l__stex_notation_arity_str }{
3068 \int_incr:N \l_tmpa_int
3069 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
3070 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
3071 \str_if_eq:VnTF \l_tmpb_str a {
3072 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3073 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{},
3074 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{},
3075 } }
3076 }{
3077 \str_if_eq:VnTF \l_tmpb_str B {

```



```

3078         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3079             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
3080             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
3081         } }
3082     }{
3083         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3084             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{ }
3085         } }
3086     }
3087 }
3088 }
3089 \stex_annotate_invisible:nnn { notationcomp }{ }{
3090     \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
3091     $ \exp_args:Nno \use:nn { \use:c {
3092         stex_notation_ \STEXInternalCurrentSymbolStr
3093         \c_hash_str \l__stex_notation_suffix_str _cs
3094     } } { \l_tmpa_tl } $
3095 }
3096 \tl_if_empty:NF \l__stex_notation_op_tl {
3097     \stex_annotate_invisible:nnn { notationopcomp }{ }{
3098         $\l__stex_notation_op_tl$
3099     }
3100 }
3101 }
3102 }
3103 }

```

(End definition for `_stex_notation_final:.`)

`\setnotation`

```

3104 \keys_define:nn { stex / setnotation } {
3105     % lang .tl_set_x:N = \l__stex_notation_lang_str ,
3106     variant .tl_set_x:N = \l__stex_notation_variant_str ,
3107     unknown .code:n = \str_set:Nx
3108         \l__stex_notation_variant_str \l_keys_key_str
3109 }
3110
3111 \cs_new_protected:Nn \_stex_setnotation_args:n {
3112     % \str_clear:N \l__stex_notation_lang_str
3113     \str_clear:N \l__stex_notation_variant_str
3114     \keys_set:nn { stex / setnotation } { #1 }
3115 }
3116
3117 \cs_new_protected:Nn \_stex_notation_setnotation:nn {
3118     \seq_if_exist:CT{l_stex_symdecl_#1_notations}{
3119         \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
3120         \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
3121     }
3122 }
3123
3124 \cs_new_protected:Nn \stex_setnotation:n {
3125     \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
3126     { \l__stex_notation_variant_str }{
3127         \stex_execute_in_module:x{ \_stex_notation_setnotation:nn {#1}{\l__stex_notation_vari

```

```

3128 \stex_debug:nn {notations}{
3129   Setting~default~notation~
3130   {\l_stex_notation_variant_str }~for~
3131   #1 \\
3132   \expandafter\meaning\csname
3133   l_stex_symdecl_#1 _notations\endcsname
3134 }
3135 }{
3136   \msg_error:nnxx{stex}{unknownnotation}{\l_stex_notation_variant_str}{#1}
3137 }
3138 }
3139
3140 \NewDocumentCommand \setnotation {m m} {
3141   \stex_get_symbol:n { #1 }
3142   \_stex_setnotation_args:n { #2 }
3143   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
3144   \stex_smsmode_do:\ignorespacesandpars
3145 }
3146
3147 \cs_new_protected:Nn \stex_copy_notations:nn {
3148   \stex_debug:nn {notations}{
3149     Copying~notations~from~#2~to~#1\\
3150     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
3151   }
3152   \tl_clear:N \l_tmpa_tl
3153   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
3154     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
3155   }
3156   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
3157     \stex_debug:nn{Here}{Here:~##1}
3158     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
3159     \edef \l_tmpa_tl {
3160       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
3161       \exp_after:wN\exp_after:wN\exp_after:wN {
3162         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
3163       }
3164     }
3165
3166     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
3167     \exp_after:wN ####\exp_after:wN 1 \exp_after:wN ####\exp_after:wN 2
3168     \exp_after:wN { \l_tmpa_tl }
3169
3170     \edef \l_tmpa_tl {
3171       \exp_after:wN \exp_not:n \exp_after:wN {
3172         \l_tmpa_tl {##### 1}{##### 2}
3173       }
3174     }
3175
3176     \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
3177
3178   \stex_execute_in_module:x {
3179     \_stex_notation_restore_notation:nnnnn
3180     {#1}{##1}
3181     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }

```

```

3182     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
3183     {
3184         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
3185             \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
3186             }
3187         }
3188     }\endgroup
3189 }
3190 }
3191
3192 \NewDocumentCommand \copynotation {m m} {
3193     \stex_get_symbol:n { #1 }
3194     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
3195     \stex_get_symbol:n { #2 }
3196     \exp_args:Noo
3197     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
3198     \stex_smsmode_do:\ignorespacesandpars
3199 }
3200

```

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

\symdef

```

3201 \keys_define:nn { stex / symdef } {
3202     name .str_set_x:N = \l_stex_symdecl_name_str ,
3203     local .bool_set:N = \l_stex_symdecl_local_bool ,
3204     args .str_set_x:N = \l_stex_symdecl_args_str ,
3205     type .tl_set:N = \l_stex_symdecl_type_tl ,
3206     def .tl_set:N = \l_stex_symdecl_definiens_tl ,
3207     reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3208     op .tl_set:N = \l__stex_notation_op_tl ,
3209     % lang .str_set_x:N = \l__stex_notation_lang_str ,
3210     variant .str_set_x:N = \l__stex_notation_variant_str ,
3211     prec .str_set_x:N = \l__stex_notation_prec_str ,
3212     argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3213     assoc .choices:nn =
3214         {bin,binl,binr,pre,conj,pwconj}
3215         {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3216     unknown .code:n = \str_set:Nx
3217         \l__stex_notation_variant_str \l_keys_key_str
3218 }
3219
3220 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
3221     \str_clear:N \l_stex_symdecl_name_str
3222     \str_clear:N \l_stex_symdecl_args_str
3223     \str_clear:N \l_stex_symdecl_assoctype_str
3224     \str_clear:N \l_stex_symdecl_reorder_str
3225     \bool_set_false:N \l_stex_symdecl_local_bool
3226     \tl_clear:N \l_stex_symdecl_type_tl
3227     \tl_clear:N \l_stex_symdecl_definiens_tl
3228     \clist_clear:N \l_stex_symdecl_argnames_clist
3229     % \str_clear:N \l__stex_notation_lang_str
3230     \str_clear:N \l__stex_notation_variant_str
3231     \str_clear:N \l__stex_notation_prec_str

```

```

3232 \tl_clear:N \l__stex_notation_op_tl
3233
3234 \keys_set:nn { stex / symdef } { #1 }
3235 }
3236
3237 \NewDocumentCommand \symdef { m O{} } {
3238   \__stex_notation_symdef_args:n { #2 }
3239   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3240   \stex_symdecl_do:n { #1 }
3241   \tl_set:Nn \l_stex_notation_after_do_tl {
3242     \__stex_notation_final:
3243     \stex_smsmode_do:\ignorespacesandpars
3244   }
3245   \str_set:Nx \l_stex_get_symbol_uri_str {
3246     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3247   }
3248   \exp_args:Nx \stex_notation_do:nnnnn
3249     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { args } }
3250     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { arity } }
3251     { \l__stex_notation_variant_str }
3252     { \l__stex_notation_prec_str }
3253 }
3254 \stex_deactivate_macro:Nn \symdef {module~environments}
3255
3256 \keys_define:nn { stex / mmtdef } {
3257   name .str_set_x:N = \l_stex_symdecl_name_str ,
3258   args .str_set_x:N = \l_stex_symdecl_args_str ,
3259   reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
3260   op .tl_set:N = \l__stex_notation_op_tl ,
3261   % lang .str_set_x:N = \l__stex_notation_lang_str ,
3262   variant .str_set_x:N = \l__stex_notation_variant_str ,
3263   prec .str_set_x:N = \l__stex_notation_prec_str ,
3264   argnames .clist_set:N = \l_stex_symdecl_argnames_clist ,
3265   assoc .choices:nn =
3266     {bin,binl,binr,pre,conj,pwconj}
3267     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
3268   unknown .code:n = \str_set:Nx
3269     \l__stex_notation_variant_str \l_keys_key_str
3270 }
3271 \cs_new_protected:Nn \stex_mmtdef_args:n {
3272   \str_clear:N \l_stex_symdecl_name_str
3273   \str_clear:N \l_stex_symdecl_args_str
3274   \str_clear:N \l_stex_symdecl_assoctype_str
3275   \str_clear:N \l_stex_symdecl_reorder_str
3276   \clist_clear:N \l_stex_symdecl_argnames_clist
3277   % \str_clear:N \l__stex_notation_lang_str
3278   \str_clear:N \l__stex_notation_variant_str
3279   \str_clear:N \l__stex_notation_prec_str
3280   \tl_clear:N \l__stex_notation_op_tl
3281
3282   \keys_set:nn { stex / mmtdef } { #1 }
3283 }
3284
3285 \NewDocumentCommand \mmtdef {m O{} }{

```

```

3286 \_stex_mmtdef_args:n{ #2 }
3287 \bool_set_true:N \l_stex_symdecl_make_macro_bool
3288 \str_if_empty:NT \l_stex_symdecl_name_str {
3289   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3290 }
3291 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3292 % \stex_annotate:nnn{ OMID }{
3293 %   \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3294 % }{}
3295 %}
3296 \stex_symdecl_do:n { #1 }
3297 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3298   \stex_annotate:nnn{ OMID }{
3299     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3300   }{},
3301   \stex_annotate:nnn{ OMID }{
3302     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3303   }{}
3304 }
3305 \tl_set:Nn \l_stex_notation_after_do_tl {
3306   \__stex_notation_final:
3307   \stex_smsmode_do:\ignorespacesandpars
3308 }
3309 \str_set:Nx \l_stex_get_symbol_uri_str {
3310   \l_stex_current_module_str ? \l_stex_symdecl_name_str
3311 }
3312 \exp_args:Nx \stex_notation_do:nnnnn
3313 { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
3314 { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
3315 { \l__stex_notation_variant_str }
3316 { \l__stex_notation_prec_str}
3317 }

```

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

29.3 Variables

```

3318 <@@=stex_variables>
3319
3320 \keys_define:nn { stex / vardef } {
3321   name .str_set_x:N = \l__stex_variables_name_str ,
3322   args .str_set_x:N = \l__stex_variables_args_str ,
3323   type .tl_set:N = \l__stex_variables_type_tl ,
3324   def .tl_set:N = \l__stex_variables_def_tl ,
3325   op .tl_set:N = \l__stex_variables_op_tl ,
3326   prec .str_set_x:N = \l__stex_variables_prec_str ,
3327   reorder .str_set_x:N = \l__stex_variables_reorder_str ,
3328   argnames .clist_set:N = \l__stex_variables_argnames_clist ,
3329   assoc .choices:nn =
3330     {bin,binl,binr,pre,conj,pwconj}
3331     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3332   bind .choices:nn =
3333     {forall,exists}
3334     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}

```

```

3335 }
3336
3337 \cs_new_protected:Nn \__stex_variables_args:n {
3338   \str_clear:N \l__stex_variables_name_str
3339   \str_clear:N \l__stex_variables_args_str
3340   \str_clear:N \l__stex_variables_prec_str
3341   \str_clear:N \l__stex_variables_assoctype_str
3342   \str_clear:N \l__stex_variables_reorder_str
3343   \str_clear:N \l__stex_variables_bind_str
3344   \tl_clear:N \l__stex_variables_type_tl
3345   \tl_clear:N \l__stex_variables_def_tl
3346   \tl_clear:N \l__stex_variables_op_tl
3347   \clist_clear:N \l__stex_variables_argnames_clist
3348
3349   \keys_set:nn { stex / vardef } { #1 }
3350 }
3351
3352 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
3353   \__stex_variables_args:n {#2}
3354   \str_if_empty:NT \l__stex_variables_name_str {
3355     \str_set:Nx \l__stex_variables_name_str { #1 }
3356   }
3357   \prop_clear:N \l_tmpa_prop
3358   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3359
3360   \int_zero:N \l_tmpb_int
3361   \bool_set_true:N \l_tmpa_bool
3362   \str_map_inline:Nn \l__stex_variables_args_str {
3363     \token_case_meaning:NnF ##1 {
3364       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3365       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3366       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3367       {\tl_to_str:n a} {
3368         \bool_set_false:N \l_tmpa_bool
3369         \int_incr:N \l_tmpb_int
3370       }
3371       {\tl_to_str:n B} {
3372         \bool_set_false:N \l_tmpa_bool
3373         \int_incr:N \l_tmpb_int
3374       }
3375     }{
3376       \msg_error:nnxx{stex}{error/wrongargs}{
3377         variable~\l__stex_variables_name_str
3378       }{##1}
3379     }
3380   }
3381   \bool_if:NTF \l_tmpa_bool {
3382     % possibly numeric
3383     \str_if_empty:NTF \l__stex_variables_args_str {
3384       \prop_put:Nnn \l_tmpa_prop { args } {}
3385       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3386     }{
3387       \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3388       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }

```

```

3389 \str_clear:N \l_tmpa_str
3390 \int_step_inline:nn \l_tmpa_int {
3391   \str_put_right:Nn \l_tmpa_str i
3392 }
3393 \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3394 \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3395 }
3396 } {
3397   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3398   \prop_put:Nnx \l_tmpa_prop { arity }
3399   { \str_count:N \l__stex_variables_args_str }
3400 }
3401 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3402 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
3403
3404 % argnames
3405
3406 \clist_clear:N \l_tmpa_clist
3407 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {
3408   \clist_if_empty:NTF \l__stex_variables_argnames_clist {
3409     \clist_put_right:Nn \l_tmpa_clist { ##1 }
3410   } {
3411     \clist_pop:NN \l__stex_variables_argnames_clist \l_tmpa_tl
3412     \exp_args:NNx \clist_put_right:Nn \l_tmpa_clist { \c_dollar_str \l_tmpa_tl }
3413   }
3414 }
3415 \prop_put:Nnx \l_tmpa_prop { argnames } { \clist_use:Nn \l_tmpa_clist , }
3416
3417
3418 \prop_set_eq:cN { l_stex_symdecl_var://\l__stex_variables_name_str _prop } \l_tmpa_prop
3419
3420 \tl_if_empty:NF \l__stex_variables_op_tl {
3421   \cs_set:cpx {
3422     stex_var_op_notation_ \l__stex_variables_name_str _cs
3423   } { \exp_not:N \comp { \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3424 }
3425
3426 \tl_set:Nn \l_stex_notation_after_do_tl {
3427   \exp_args:Nne \use:nn {
3428     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3429     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3430   } { {
3431     \exp_after:wN \exp_after:wN \exp_after:wN
3432     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3433     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3434   } }
3435
3436 \stex_if_do_html:T {
3437   \stex_annotate_invisible:nnn { vardecl } { \l__stex_variables_name_str } {
3438     \stex_annotate_invisible:nnn { precedence }
3439     { \l__stex_variables_prec_str } { }
3440     \tl_if_empty:NF \l__stex_variables_type_tl { \stex_annotate_invisible:nnn { type } { } { $\l
3441     \stex_annotate_invisible:nnn { args } { \l__stex_variables_args_str } { }
3442     \stex_annotate_invisible:nnn { macroname } { #1 } { }
3443     \tl_if_empty:NF \l__stex_variables_def_tl {

```

```

3443     \stex_annotate_invisible:nnn{definiens}{}
3444     {${\l__stex_variables_def_tl$}
3445   }
3446   \str_if_empty:NF \l__stex_variables_assoctype_str {
3447     \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3448   }
3449   \str_if_empty:NF \l__stex_variables_reorder_str {
3450     \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3451   }
3452   \int_zero:N \l_tmpa_int
3453   \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3454   \tl_clear:N \l_tmpa_tl
3455   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3456     \int_incr:N \l_tmpa_int
3457     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3458     \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3459     \str_if_eq:VnTF \l_tmpb_str a {
3460       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3461         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3462         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3463       } }
3464     }{
3465       \str_if_eq:VnTF \l_tmpb_str B {
3466         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3467           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3468           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3469         } }
3470       }{
3471         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3472           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3473         } }
3474       }
3475     }
3476   }
3477   \stex_annotate_invisible:nnn { notationcomp }{}{
3478     \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3479     $ \exp_args:Nno \use:nn { \use:c {
3480       stex_var_notation_\l__stex_variables_name_str _cs
3481     } } { \l_tmpa_tl } $
3482   }
3483   \tl_if_empty:NF \l__stex_variables_op_tl {
3484     \stex_annotate_invisible:nnn { notationopcomp }{}{
3485       ${\l__stex_variables_op_tl$}
3486     }
3487   }
3488 }
3489 \str_if_empty:NF \l__stex_variables_bind_str {
3490   \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variabl
3491 }
3492 }\ignorespacesandpars
3493 }
3494
3495 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3496 }

```



```

3497
3498 \cs_new:Nn \_stex_reset:N {
3499   \tl_if_exist:NTF #1 {
3500     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3501   }{
3502     \let \exp_not:N #1 \exp_not:N \undefined
3503   }
3504 }
3505
3506 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3507   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3508   \exp_args:Nnx \use:nn {
3509     % TODO
3510     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3511       #2
3512     }
3513   }{
3514     \_stex_reset:N \varnot
3515     \_stex_reset:N \vartype
3516     \_stex_reset:N \vardefi
3517   }
3518 }
3519
3520 \NewDocumentCommand \vardef { s } {
3521   \IfBooleanTF#1 {
3522     \__stex_variables_do_complex:nn
3523   }{
3524     \__stex_variables_do_simple:nnn
3525   }
3526 }
3527
3528 \NewDocumentCommand \svar { 0{} m }{
3529   \tl_if_empty:nTF {#1}{
3530     \str_set:Nn \l_tmpa_str { #2 }
3531   }{
3532     \str_set:Nn \l_tmpa_str { #1 }
3533   }
3534   \_stex_term_omv:nn {
3535     var://\l_tmpa_str
3536   }{
3537     \exp_args:Nnx \use:nn {
3538       \def\comp{\_varcomp}
3539       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3540       \comp{ #2 }
3541     }{
3542       \_stex_reset:N \comp
3543       \_stex_reset:N \STEXInternalCurrentSymbolStr
3544     }
3545   }
3546 }
3547
3548
3549
3550 \keys_define:nn { stex / varseq } {

```

```

3551 name      .str_set_x:N = \l__stex_variables_name_str ,
3552 args      .int_set:N   = \l__stex_variables_args_int ,
3553 type      .tl_set:N    = \l__stex_variables_type_tl ,
3554 mid       .tl_set:N    = \l__stex_variables_mid_tl ,
3555 bind      .choices:nn =
3556           {forall,exists}
3557           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3558 }
3559
3560 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3561   \str_clear:N \l__stex_variables_name_str
3562   \int_set:Nn \l__stex_variables_args_int 1
3563   \tl_clear:N \l__stex_variables_type_tl
3564   \str_clear:N \l__stex_variables_bind_str
3565
3566   \keys_set:nn { stex / varseq } { #1 }
3567 }
3568
3569 \NewDocumentCommand \varseq {m O{} m m m}{
3570   \__stex_variables_seq_args:n { #2 }
3571   \str_if_empty:NT \l__stex_variables_name_str {
3572     \str_set:Nx \l__stex_variables_name_str { #1 }
3573   }
3574   \prop_clear:N \l_tmpa_prop
3575   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3576
3577   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3578   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3579     \msg_error:nnxx{stex}{error/seqlength}
3580     {\int_use:N \l__stex_variables_args_int}
3581     {\seq_count:N \l_tmpa_seq}
3582   }
3583   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3584   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3585     \msg_error:nnxx{stex}{error/seqlength}
3586     {\int_use:N \l__stex_variables_args_int}
3587     {\seq_count:N \l_tmpb_seq}
3588   }
3589   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3590   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3591
3592   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str_cs}
3593   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3594
3595   % argnames
3596
3597   \clist_clear:N \l_tmpa_clist
3598   \int_step_inline:nn {\l__stex_variables_args_int} {
3599     \clist_put_right:Nn \l_tmpa_clist {##1}
3600   }
3601   \prop_put:Nnx \l_tmpa_prop {argnames} {\clist_use:Nn \l_tmpa_clist ,}
3602
3603
3604

```

```

3605 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str_cs}}
3606 \int_step_inline:nn \l__stex_variables_args_int {
3607   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3608 }
3609 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3610 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3611 \tl_if_empty:NF \l__stex_variables_mid_tl {
3612   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3613   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3614 }
3615 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str_cs}}
3616 \int_step_inline:nn \l__stex_variables_args_int {
3617   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3618 }
3619 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3620 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3621
3622
3623
3624 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3625
3626 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3627
3628 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str_cs}}
3629
3630 \int_step_inline:nn \l__stex_variables_args_int {
3631   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3632     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3633   }}
3634 }
3635
3636 \tl_set:Nx \l_tmpa_tl {
3637   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3638     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3639   }
3640 }
3641
3642 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3643
3644 \exp_args:Nno \use:nn {
3645   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str_cs}
3646   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3647
3648   \stex_debug:nn{sequences}{New~Sequence:~
3649     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str_cs\endcsname\\~\\
3650     \prop_to_keyval:N \l_tmpa_prop
3651   }
3652   \prop_set_eq:cN {\l_stex_symdecl_varseq://\l__stex_variables_name_str_prop}\l_tmpa_prop
3653
3654   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3655     \tl_if_empty:NF \l__stex_variables_type_tl {
3656       \stex_annotate:nnn {type}{\l__stex_variables_type_tl}
3657     }
3658     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{

```

```

3659 \str_if_empty:NF \l__stex_variables_bind_str {
3660   \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{ }
3661 }
3662 \stex_annotate:nnn{startindex}{ }{ $3$ }
3663 \stex_annotate:nnn{endindex}{ }{ $4$ }
3664
3665 \tl_clear:N \l_tmpa_tl
3666 \int_step_inline:nn \l__stex_variables_args_int {
3667   \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3668     \stex_annotate:nnn{argmarker}{##1}{ }
3669   } }
3670 }
3671 \stex_annotate_invisible:nnn { notationcomp }{ }{
3672   \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3673   $ \exp_args:Nno \use:nn { \use:c {
3674     stex_varseq\l__stex_variables_name_str _cs
3675   } } { \l_tmpa_tl } $
3676 }
3677 \stex_annotate_invisible:nnn { notationopcomp }{ }{
3678   $ \prop_item:Nn \l_tmpa_prop { notation } $
3679 }
3680
3681 }}
3682
3683 \ignorespacesandpars
3684 }
3685
3686
3687 \keys_define:nn { stex / mmtdecl } {
3688   name      .str_set_x:N = \l_stex_symdecl_name_str ,
3689   args      .str_set_x:N = \l_stex_symdecl_args_str ,
3690   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
3691   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
3692   argnames  .clist_set:N = \l_stex_symdecl_argnames_clist ,
3693   assoc     .choices:nn =
3694     {bin,binl,binr,pre,conj,pwconj}
3695     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}}
3696 }
3697
3698 \cs_new_protected:Nn \_stex_mmtdecl_args:n {
3699   \str_clear:N \l_stex_symdecl_name_str
3700   \str_clear:N \l_stex_symdecl_args_str
3701   \str_clear:N \l_stex_symdecl_deprecate_str
3702   \str_clear:N \l_stex_symdecl_reorder_str
3703   \str_clear:N \l_stex_symdecl_assoc_type_str
3704   \bool_set_false:N \l_stex_symdecl_local_bool
3705   \clist_clear:N \l_stex_symdecl_argnames_clist
3706
3707   \keys_set:nn { stex / symdecl } { #1 }
3708 }
3709
3710 \NewDocumentCommand \mmtdecl { s m O{ } } {
3711   \_stex_mmtdecl_args:n{#3}
3712   \IfBooleanTF #1 {

```

```

3713   \bool_set_false:N \l_stex_symdecl_make_macro_bool
3714 } {
3715   \bool_set_true:N \l_stex_symdecl_make_macro_bool
3716 }
3717 \str_if_empty:NT \l_stex_symdecl_name_str {
3718   \str_set:Nx \l_stex_symdecl_name_str { #1 }
3719 }
3720 %\tl_set:Nx \l_stex_symdecl_definiens_tl {
3721 %   \stex_annotate:nnn{ OMID }{
3722 %     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3723 %   }{}
3724 %}
3725 \stex_symdecl_do:n{#2}
3726 \MMTrule{rules.stex.mmt.kwarc.info?SubstitutionRule}{
3727   \stex_annotate:nnn{ OMID }{
3728     \l_stex_current_module_str ? \l_stex_symdecl_name_str
3729   }{},
3730   \stex_annotate:nnn{ OMID }{
3731     \l_stex_module_mmtfor_str?\l_stex_symdecl_name_str
3732   }{}
3733 }
3734 \stex_smsmode_do:
3735 }
3736
3737 \stex_deactivate_macro:Nn \mmtdecl {mmtinterface~environments}
3738 \stex_deactivate_macro:Nn \mmtdef {mmtinterface~environments}
3739
3740 </package>

```

Chapter 30

STEX -Terms Implementation

```
3741 <*package>
3742
3743 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3744
3745 <@@=stex_terms>
3746
3747 Warnings and error messages
3748 \msg_new:nnn{stex}{error/nonotation}{
3749   Symbol~#1~invoked,~but~has~no~notation#2!
3750 }
3751 \msg_new:nnn{stex}{error/notationarg}{
3752   Error~in~parsing~notation~#1
3753 }
3754 \msg_new:nnn{stex}{error/noop}{
3755   Symbol~#1~has~no~operator~notation~for~notation~#2
3756 }
3757 \msg_new:nnn{stex}{error/notallowed}{
3758   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3759 }
3760 \msg_new:nnn{stex}{error/doubleargument}{
3761   Argument~#1~of~symbol~#2~already~assigned
3762 }
3763 \msg_new:nnn{stex}{error/overarity}{
3764   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3765 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3765
3766
3767 \bool_new:N \l_stex_allow_semantic_bool
3768 \bool_set_true:N \l_stex_allow_semantic_bool
3769
```

```

3770 \cs_new_protected:Nn \stex_invoke_symbol:n {
3771   \ifvmode\indent\fi
3772   \bool_if:NTF \l_stex_allow_semantic_bool {
3773     \str_if_eq:eeF {
3774       \prop_item:cn {
3775         l_stex_symdecl_#1_prop
3776       }{ deprecate }
3777     }{}{
3778       \msg_warning:nxxx{stex}{warning/deprecated}{
3779         Symbol~#1
3780       }{
3781         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3782       }
3783     }
3784     \if_mode_math:
3785       \exp_after:wN \__stex_terms_invoke_math:n
3786     \else:
3787       \exp_after:wN \__stex_terms_invoke_text:n
3788     \fi: { #1 }
3789   }{
3790     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3791   }
3792 }
3793
3794 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3795   \peek_charcode_remove:NTF ! {
3796     \__stex_terms_invoke_op_custom:nn {#1}
3797   }{
3798     \__stex_terms_invoke_custom:nn {#1}
3799   }
3800 }
3801
3802 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3803   \peek_charcode_remove:NTF ! {
3804     % operator
3805     \peek_charcode_remove:NTF * {
3806       % custom op
3807       \__stex_terms_invoke_op_custom:nn {#1}
3808     }{
3809       % op notation
3810       \peek_charcode:NTF [ {
3811         \__stex_terms_invoke_op_notation:nw {#1}
3812       }{
3813         \__stex_terms_invoke_op_notation:nw {#1}[]
3814       }
3815     }
3816   }{
3817     \peek_charcode_remove:NTF * {
3818       \__stex_terms_invoke_custom:nn {#1}
3819       % custom
3820     }{
3821       % normal
3822       \peek_charcode:NTF [ {
3823         \__stex_terms_invoke_notation:nw {#1}

```

```

3824     }{
3825         \__stex_terms_invoke_notation:nw {#1}[]
3826     }
3827 }
3828 }
3829 }
3830
3831
3832 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3833     \exp_args:Nnx \use:nn {
3834         \def\comp{\_comp}
3835         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3836         \bool_set_false:N \l_stex_allow_semantic_bool
3837         \stex_mathml_intent:nn{#1}{
3838             \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3839                 \comp{ #2 }
3840             }
3841         }
3842     }{
3843         \stex_reset:N \comp
3844         \stex_reset:N \STEXInternalCurrentSymbolStr
3845         \bool_set_true:N \l_stex_allow_semantic_bool
3846     }
3847 }
3848
3849 \keys_define:nn { stex / terms } {
3850     % lang      .tl_set_x:N = \l_stex_notation_lang_str ,
3851     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3852     unknown .code:n      = \str_set:Nx
3853         \l_stex_notation_variant_str \l_keys_key_str
3854 }
3855
3856 \cs_new_protected:Nn \__stex_terms_args:n {
3857     % \str_clear:N \l_stex_notation_lang_str
3858     \str_clear:N \l_stex_notation_variant_str
3859
3860     \keys_set:nn { stex / terms } { #1 }
3861 }
3862
3863 \cs_new_protected:Nn \stex_find_notation:nn {
3864     \__stex_terms_args:n { #2 }
3865     \seq_if_empty:cTF {
3866         l_stex_symdecl_ #1 _notations
3867     } {
3868         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3869     } {
3870         \str_if_empty:NTF \l_stex_notation_variant_str {
3871             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3872         }{
3873             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3874                 \l_stex_notation_variant_str
3875             }{
3876                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3877             }{

```



```

3878         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3879         ~\l_stex_notation_variant_str
3880     }
3881 }
3882 }
3883 }
3884 }
3885
3886 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3887     \exp_args:Nnx \use:nn {
3888         \def\comp{\_comp}
3889         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3890         \stex_find_notation:nn { #1 }{ #2 }
3891         \bool_set_false:N \l_stex_allow_semantic_bool
3892         \cs_if_exist:cTF {
3893             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3894         }{
3895             \_stex_term_oms:nnn { #1 }{
3896                 #1 \c_hash_str \l_stex_notation_variant_str
3897             }{
3898                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3899             }
3900         }{
3901             \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3902                 \cs_if_exist:cTF {
3903                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3904                 }{
3905                     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3906                         \_stex_reset:N \comp
3907                         \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3908                         \_stex_reset:N \STEXInternalCurrentSymbolStr
3909                         \bool_set_true:N \l_stex_allow_semantic_bool
3910                     }
3911                     \def\comp{\_comp}
3912                     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3913                     \bool_set_false:N \l_stex_allow_semantic_bool
3914                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3915                 }{
3916                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3917                     ~\l_stex_notation_variant_str
3918                     }
3919                 }
3920             }{
3921                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3922             }
3923         }
3924     }{
3925         \_stex_reset:N \comp
3926         \_stex_reset:N \STEXInternalCurrentSymbolStr
3927         \bool_set_true:N \l_stex_allow_semantic_bool
3928     }
3929 }
3930
3931 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```

```

3932 \stex_find_notation:nn { #1 }{ #2 }
3933 \cs_if_exist:cTF {
3934   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3935 }{
3936   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3937     \_stex_reset:N \comp
3938     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3939     \_stex_reset:N \STEXInternalCurrentSymbolStr
3940     \bool_set_true:N \l_stex_allow_semantic_bool
3941   }
3942   \def\comp{\_comp}
3943   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3944   \bool_set_false:N \l_stex_allow_semantic_bool
3945   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3946 }{
3947   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3948     ~\l_stex_notation_variant_str
3949   }
3950 }
3951 }
3952
3953 \prop_new:N \l__stex_terms_custom_args_prop
3954 \clist_new:N \l_stex_argnames_seq
3955 \seq_new:N \l__stex_terms_tmp_seq
3956
3957 \cs_new_protected:Nn \__stex_terms_custom_comp:n{ \bool_set_false:N \l_stex_allow_semantic_bo
3958
3959 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3960   \exp_args:Nnx \use:nn {
3961     \def\comp{\__stex_terms_custom_comp:n}
3962     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3963     \prop_clear:N \l__stex_terms_custom_args_prop
3964     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3965     \prop_get:cnN {
3966       l_stex_symdecl_#1 _prop
3967     }{ args } \l_tmpa_str
3968     \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
3969       \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
3970     }
3971     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3972     \tl_set:Nn \arg { \__stex_terms_arg: }
3973     \str_if_empty:NTF \l_tmpa_str {
3974       \stex_mathml_intent:nn{#1}{
3975         \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3976       }
3977     }{
3978       \seq_clear:N \l__stex_terms_tmp_seq
3979       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
3980         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
3981         \bool_lazy_or:nnT{
3982           \str_if_eq_p:nn{a}{\str_item:Nn \l_tmpa_str{##1}}
3983         }{
3984           \str_if_eq_p:nn{B}{\str_item:Nn \l_tmpa_str{##1}}
3985         }{

```

```

3986         \tl_put_right:Nn \l__stex_terms_tmp_tl +
3987     }
3988     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
3989 }
3990 \stex_mathml_intent:nn{
3991     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}]{ args }(
3992     \seq_use:Nn \l__stex_terms_tmp_seq ,
3993 )
3994 }{
3995     \str_if_in:NnTF \l_tmpa_str b {
3996         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3997     }{
3998         \str_if_in:NnTF \l_tmpa_str B {
3999             \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4000         }{
4001             \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
4002         }
4003     }
4004 }
4005 }
4006 % TODO check that all arguments exist
4007 }{
4008     \_stex_reset:N \l_stex_argnames_seq
4009     \_stex_reset:N \STEXInternalCurrentSymbolStr
4010     \_stex_reset:N \arg
4011     \_stex_reset:N \comp
4012     \_stex_reset:N \l__stex_terms_custom_args_prop
4013     %\bool_set_true:N \l_stex_allow_semantic_bool
4014 }
4015 }
4016
4017 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
4018     \tl_if_empty:nTF {#2}{
4019         \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
4020         \bool_set_true:N \l_tmpa_bool
4021         \bool_do_while:Nn \l_tmpa_bool {
4022             \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
4023             \int_incr:N \l_tmpa_int
4024         }{
4025             \bool_set_false:N \l_tmpa_bool
4026         }
4027     }
4028     }{
4029         \int_set:Nn \l_tmpa_int { #2 }
4030     }
4031     \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
4032     \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
4033         \msg_error:nnxxx{stex}{error/overarity}
4034         {\int_use:N \l_tmpa_int}
4035         {\STEXInternalCurrentSymbolStr}
4036         {\str_count:N \l_tmpa_str}
4037     }
4038     \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
4039     \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {

```

```

4040 \bool_lazy_any:nF {
4041   {\str_if_eq_p:Vn \l_tmpa_str {a}}
4042   {\str_if_eq_p:Vn \l_tmpa_str {B}}
4043 }{
4044   \msg_error:nnxx{stex}{error/doubleargument}
4045   {\int_use:N \l_tmpa_int}
4046   {\STEXInternalCurrentSymbolStr}
4047 }
4048 }
4049 \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
4050 \bool_if:NTF \l_stex_allow_semantic_bool \use_i:nn {
4051   \bool_set_true:N \l_stex_allow_semantic_bool
4052   \use:nn
4053 }
4054 {
4055 \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4056   \IfBooleanTF#1{
4057     \stex_annotate_invisible:n { %TODO
4058       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4059     }
4060   }{ %TODO
4061     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
4062   }
4063 }}
4064 {\bool_set_false:N \l_stex_allow_semantic_bool}
4065 }
4066
4067
4068 \cs_new_protected:Nn \_stex_term_arg:nn {
4069   \bool_set_true:N \l_stex_allow_semantic_bool
4070   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
4071   \bool_set_false:N \l_stex_allow_semantic_bool
4072 }
4073
4074 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
4075   \exp_args:Nnx \use:nn
4076   { \int_set:Nn \l__stex_terms_downprec { #2 }
4077     \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq \l_tmpa_int}{
4078       \_stex_term_arg:nn { #1 }{ #3 }
4079     }
4080   }
4081   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4082 }

```

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

`\STEXInternalTermMathAssocArgiiii`

```

4083 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiiii #1#2#3#4#5 {
4084   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
4085   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#5#1}{#2}}
4086   \tl_if_empty:nTF { #3 }{
4087     \STEXInternalTermMathArgiii{#5#1}{#2}{#3}
4088   }{
4089     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{

```

```

4090 \expandafter\if\expandafter\relax\noexpand#3
4091 \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_maybe_sequence:Nnn#3{#1}{#5}}
4092 \else
4093 \tl_set:Nn \l_tmpa_tl {\_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}}
4094 \fi
4095 \l_tmpa_tl
4096 }{
4097 \_stex_terms_math_assoc_arg_simple:nnn{#1}{#3}{#5}
4098 }
4099 }
4100 }
4101
4102 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_maybe_sequence:Nnn {
4103 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
4104 \str_if_empty:NTF \l_tmpa_str {
4105 \exp_args:Nx \cs_if_eq:NNTF {
4106 \tl_head:N #1
4107 } \stex_invoke_sequence:n {
4108 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
4109 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
4110 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {l_stex_symdecl_varseq://\l_tmpa_str _prop}{notat
4111 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
4112 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
4113 \exp_not:n{\exp_args:Nnx \use:nn} {
4114 \exp_not:n {
4115 \def\comp{\_varcomp}
4116 \str_set:Nn \STEXInternalCurrentSymbolStr
4117 } {varseq://\l_tmpa_str}
4118 \exp_not:n{ ##1 }
4119 }{
4120 \exp_not:n {
4121 \_stex_reset:N \comp
4122 \_stex_reset:N \STEXInternalCurrentSymbolStr
4123 }
4124 }
4125 }}}
4126 \exp_args:Nno \use:n {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
4127 \seq_reverse:N \l_tmpa_seq
4128 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
4129 \seq_map_inline:Nn \l_tmpa_seq {
4130 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4131 \exp_args:Nno
4132 \l_tmpa_cs { ##1 } \l_tmpa_tl
4133 }
4134 }
4135 \tl_set:Nx \l_tmpa_tl {
4136 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
4137 \exp_args:No \exp_not:n \l_tmpa_tl
4138 }
4139 }
4140 \exp_args:No\l_tmpb_tl\l_tmpa_tl
4141 }{
4142 \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4143 }

```

```

4144 } {
4145   \_stex_terms_math_assoc_arg_simple:nnn{#2} { #1 }{#3}
4146 }
4147
4148 }
4149
4150 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_simple:nnn {
4151   \clist_set:Nn \l_tmpa_clist{ #2 }
4152   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
4153     \tl_set:Nn \l_tmpa_tl {
4154       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4155         \_stex_term_arg:nn{A#3#1}{ #2 } }
4156     }
4157   }{
4158     \clist_reverse:N \l_tmpa_clist
4159     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
4160     \tl_set:Nx \l_tmpa_tl {
4161       \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4162         \_stex_term_arg:nn{A#3#1}{
4163           \exp_args:No \exp_not:n \l_tmpa_tl
4164         }
4165       }
4166     }
4167     \clist_map_inline:Nn \l_tmpa_clist {
4168       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
4169         \exp_args:Nno
4170         \l_tmpa_cs {
4171           \stex_mathml_arg:nn{\seq_item:Nn \l_stex_argnames_seq #1}{
4172             \_stex_term_arg:nn{A#3#1}{##1}
4173           }
4174         } \l_tmpa_tl
4175       }
4176     }
4177     \exp_args:No\l_tmpb_tl\l_tmpa_tl
4178   }

```

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

30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
4179 \tl_const:Nx \infprec {\int_use:N \c_max_int}
4180 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
4181 \int_new:N \l__stex_terms_downprec
4182 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

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

Bracketing:

```

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
4183 \tl_set:Nn \l__stex_terms_left_bracket_str (
4184 \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

```

4185 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
4186   \bool_if:NTF \l__stex_terms_brackets_done_bool {
4187     \bool_set_false:N \l__stex_terms_brackets_done_bool
4188     #2
4189   } {
4190     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
4191       \bool_if:NTF \l__stex_inarray_bool { #2 } {
4192         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
4193         \dobrackets { #2 }
4194       }
4195     }{ #2 }
4196   }
4197 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

4198 \bool_new:N \l__stex_terms_brackets_done_bool
4199 %\RequirePackage{scalerel}
4200 \cs_new_protected:Npn \dobrackets #1 {
4201   %\ThisStyle{\if D\m@switch
4202   %   \exp_args:Nnx \use:nn
4203   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
4204   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
4205   %   \else
4206   \exp_args:Nnx \use:nn
4207   {
4208     \bool_set_true:N \l__stex_terms_brackets_done_bool
4209     \int_set:Nn \l__stex_terms_downprec \infprec
4210     \l__stex_terms_left_bracket_str
4211     #1
4212   }
4213   {
4214     \bool_set_false:N \l__stex_terms_brackets_done_bool
4215     \l__stex_terms_right_bracket_str
4216     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
4217   }
4218   %\fi}
4219 }
```

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

`\withbrackets`

```

4220 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
4221   \exp_args:Nnx \use:nn
4222   {
4223     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
4224     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
4225     #3
4226   }
4227 }
```

```

4228 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
4229 {\l__stex_terms_left_bracket_str}
4230 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
4231 {\l__stex_terms_right_bracket_str}
4232 }
4233 }

```

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

`\STEXinvisible`

```

4234 \cs_new_protected:Npn \STEXinvisible #1 {
4235   \stex_annotate_invisible:n { #1 }
4236 }

```

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

`\STEXInternalTermMathOMSiiii`

```

4237 \cs_new_protected:Nn \_stex_term_oms:nnn {
4238   \stex_annotate:nnn{ OMID }{ #2 }{
4239     #3
4240   }
4241 }
4242
4243 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
4244   \__stex_terms_maybe_brackets:nn { #3 }{
4245     \stex_mathml_intent:nn{#1} {
4246       \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4247     }
4248   }
4249 }

```

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

`_stex_term_math_omv:nn`

```

4250 \cs_new_protected:Nn \_stex_term_omv:nn {
4251   \stex_annotate:nnn{ OMV }{ #1 }{
4252     #2
4253   }
4254 }

```

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

`\STEXInternalTermMathOMAiiai`

```

4255 \cs_new_protected:Nn \_stex_term_oma:nnn {
4256   \stex_annotate:nnn{ OMA }{ #2 }{
4257     #3
4258   }
4259 }
4260
4261 \cs_new_protected:Npn \STEXInternalTermMathOMAiiai #1#2#3#4 {
4262   \exp_args:Nnx \use:nn {
4263     \seq_clear:N \l__stex_terms_tmp_seq
4264     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4265       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {

```



```

4266     \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4267   }
4268   \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4269     \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4270     \bool_lazy_or:nnT{
4271       \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4272     }{
4273       \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4274     }{
4275       \tl_put_right:Nn \l__stex_terms_tmp_tl +
4276     }
4277     \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl
4278   }
4279 }
4280 \__stex_terms_maybe_brackets:nn { #3 }{
4281   \stex_mathml_intent:nn{
4282     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4283       \seq_use:Nn \l__stex_terms_tmp_seq ,
4284     )
4285   }{
4286     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4287   }
4288 }
4289 }{
4290   \_stex_reset:N \l_stex_argnames_seq
4291 }
4292 }

```

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

`\STEXInternalTermMathOMBi`

```

4293 \cs_new_protected:Nn \_stex_term_ombind:nnn {
4294   \stex_annotate:nnn{ OMBIND }{ #2 }{
4295     #3
4296   }
4297 }
4298
4299 \cs_new_protected:Npn \STEXInternalTermMathOMBi #1#2#3#4 {
4300   \exp_args:Nnx \use:nn {
4301     \seq_clear:N \l__stex_terms_tmp_seq
4302     \prop_if_exist:cT{l_stex_symdecl_#1 _prop}{
4303       \exp_args:NNx \seq_set_from_clist:Nn \l_stex_argnames_seq {
4304         \prop_item:cn {l_stex_symdecl_#1 _prop}{argnames}
4305       }
4306       \exp_args:Nx\int_step_inline:nn{\prop_item:cn{l_stex_symdecl_#1 _prop}{arity}}{
4307         \tl_set:Nx \l__stex_terms_tmp_tl {\seq_item:Nn \l_stex_argnames_seq {##1}}
4308         \bool_lazy_or:nnT{
4309           \str_if_eq_p:nn{a}{\str_item:Nn\l_tmpa_str{##1}}
4310         }{
4311           \str_if_eq_p:nn{B}{\str_item:Nn\l_tmpa_str{##1}}
4312         }{
4313           \tl_put_right:Nn \l__stex_terms_tmp_tl +
4314         }
4315         \seq_put_right:No \l__stex_terms_tmp_seq \l__stex_terms_tmp_tl

```

```

4316   }
4317 }
4318 \__stex_terms_maybe_brackets:nn { #3 }{
4319   \stex_mathml_intent:nn{
4320     #1[\prop_item:cn {l_stex_symdecl_#1 _prop}{ args }](
4321       \seq_use:Nn \l__stex_terms_tmp_seq ,
4322     )
4323   }{
4324     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
4325   }
4326 }
4327 }{
4328   \stex_reset:N \l_stex_argnames_seq
4329 }
4330 }

```

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

`\symref`
`\symname`

```

4331 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4332
4333 \keys_define:nn { stex / symname } {
4334   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
4335   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
4336   root     .tl_set_x:N      = \l__stex_terms_root_tl
4337 }
4338
4339 \cs_new_protected:Nn \stex_symname_args:n {
4340   \tl_clear:N \l__stex_terms_post_tl
4341   \tl_clear:N \l__stex_terms_pre_tl
4342   \tl_clear:N \l__stex_terms_root_str
4343   \keys_set:nn { stex / symname } { #1 }
4344 }
4345
4346 \NewDocumentCommand \symref { m m }{
4347   \let\compemph_uri_prev:\compemph@uri
4348   \let\compemph@uri\symrefemph@uri
4349   \STEXsymbol{#1}!\{ #2 }
4350   \let\compemph@uri\compemph_uri_prev:
4351 }
4352
4353 \NewDocumentCommand \synonym { 0{} m m }{
4354   \stex_symname_args:n { #1 }
4355   \let\compemph_uri_prev:\compemph@uri
4356   \let\compemph@uri\symrefemph@uri
4357   % TODO
4358   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
4359   \let\compemph@uri\compemph_uri_prev:
4360 }
4361
4362 \NewDocumentCommand \symname { 0{} m }{
4363   \stex_symname_args:n { #1 }
4364   \stex_get_symbol:n { #2 }
4365   \str_set:Nx \l_tmpa_str {

```

```

4366 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4367 }
4368 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4369
4370 \let\compemph_uri_prev:\compemph@uri
4371 \let\compemph@uri\symrefemph@uri
4372 \exp_args:NNx \use:nn
4373 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
4374 \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
4375 } }
4376 \let\compemph@uri\compemph_uri_prev:
4377 }
4378
4379 \NewDocumentCommand \Symname { 0{} m }{
4380 \stex_symname_args:n { #1 }
4381 \stex_get_symbol:n { #2 }
4382 \str_set:Nx \l_tmpa_str {
4383 \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4384 }
4385 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4386 \let\compemph_uri_prev:\compemph@uri
4387 \let\compemph@uri\symrefemph@uri
4388 \exp_args:NNx \use:nn
4389 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
4390 \exp_after:wN \stex_capitalize:n \l_tmpa_str
4391 \l__stex_terms_post_tl
4392 } }
4393 \let\compemph@uri\compemph_uri_prev:
4394 }

```

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

30.3 Notation Components

```

4395 <@@=stex_notationcomps>
4396
4397 \comp
4398 \compemph@uri
4399 \compemph
4400 \defemph
4401 \defemph@uri
4402 \symrefemph
4403 \symrefemph@uri
4404 \varemp
4405 \varemp@uri
4396 \cs_new_protected:Npn \_comp #1 {
4397 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4398 \stex_html_backend:TF {
4399 \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4400 }{
4401 \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
4402 }
4403 }
4404 }
4405
4406 \cs_new_protected:Npn \_varcomp #1 {
4407 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
4408 \stex_html_backend:TF {
4409 \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
4410 }{
4411 \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
4412 }

```

```

4413 }
4414 }
4415
4416 \def\comp{\_comp}
4417
4418 \cs_new_protected:Npn \compemph@uri #1 #2 {
4419   \compemph{ #1 }
4420 }
4421
4422
4423 \cs_new_protected:Npn \compemph #1 {
4424   #1
4425 }
4426
4427 \cs_new_protected:Npn \defemph@uri #1 #2 {
4428   \defemph{#1}
4429 }
4430
4431 \cs_new_protected:Npn \defemph #1 {
4432   \textbf{#1}
4433 }
4434
4435 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
4436   \symrefemph{#1}
4437 }
4438
4439 \cs_new_protected:Npn \symrefemph #1 {
4440   \emph{#1}
4441 }
4442
4443 \cs_new_protected:Npn \varemp@uri #1 #2 {
4444   \varemp{#1}
4445 }
4446
4447 \cs_new_protected:Npn \varemp #1 {
4448   #1
4449 }

```

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

`\ellipses`

```

4450 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
4451 \bool_new:N \l_stex_inparray_bool
4452 \bool_set_false:N \l_stex_inparray_bool
4453 \NewDocumentCommand \parray { m m } {
4454   \begin{group}
4455     \bool_set_true:N \l_stex_inparray_bool
4456     \begin{array}{#1}
4457       #2
4458     \end{array}
4459   \end{group}

```

```

4460 }
4461
4462 \NewDocumentCommand \prmatrix { m } {
4463   \begingroup
4464   \bool_set_true:N \l_stex_inarray_bool
4465   \begin{matrix}
4466     #1
4467   \end{matrix}
4468   \endgroup
4469 }
4470
4471 \def \maybepline {
4472   \bool_if:NT \l_stex_inarray_bool {\hline}
4473 }
4474
4475 \def \parrayline #1 #2 {
4476   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
4477 }
4478
4479 \def \pmrow #1 { \parrayline{}{ #1 } }
4480
4481 \def \parraylineh #1 #2 {
4482   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
4483 }
4484
4485 \def \parraycell #1 {
4486   #1 \bool_if:NT \l_stex_inarray_bool {&}
4487 }

```

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

30.4 Variables

4488 <@=stex_variables>

\stex_invoke_variable:n Invokes a variable

```

4489 \cs_new_protected:Nn \stex_invoke_variable:n {
4490   \if_mode_math:
4491     \exp_after:wN \__stex_variables_invoke_math:n
4492   \else:
4493     \exp_after:wN \__stex_variables_invoke_text:n
4494   \fi: {#1}
4495 }
4496
4497 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
4498   \peek_charcode_remove:NTF ! {
4499     \__stex_variables_invoke_op_custom:nn {#1}
4500   }{
4501     \__stex_variables_invoke_custom:nn {#1}
4502   }
4503 }
4504
4505
4506 \cs_new_protected:Nn \__stex_variables_invoke_math:n {

```

```

4507 \peek_charcode_remove:NTF ! {
4508   \peek_charcode_remove:NTF ! {
4509     \peek_charcode:NTF [ {
4510       % TODO throw error
4511     }{
4512       \__stex_variables_invoke_op_custom:nn
4513     }
4514   }{
4515     \__stex_variables_invoke_op:n { #1 }
4516   }
4517 }{
4518   \peek_charcode_remove:NTF * {
4519     \__stex_variables_invoke_custom:nn { #1 }
4520   }{
4521     \__stex_variables_invoke_math_ii:n { #1 }
4522   }
4523 }
4524 }
4525
4526 \cs_new_protected:Nn \__stex_variables_invoke_op_custom:nn {
4527   \exp_args:Nnx \use:nn {
4528     \def\comp{\_varcomp}
4529     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4530     \bool_set_false:N \l_stex_allow_semantic_bool
4531     \_stex_term_omv:nn {var://#1}{
4532       \comp{ #2 }
4533     }
4534   }{
4535     \_stex_reset:N \comp
4536     \_stex_reset:N \STEXInternalCurrentSymbolStr
4537     \bool_set_true:N \l_stex_allow_semantic_bool
4538   }
4539 }
4540
4541 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
4542   \cs_if_exist:cTF {
4543     stex_var_op_notation_ #1 _cs
4544   }{
4545     \exp_args:Nnx \use:nn {
4546       \def\comp{\_varcomp}
4547       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4548       \_stex_term_omv:nn { var://#1 }{
4549         \use:c{stex_var_op_notation_ #1 _cs }
4550       }
4551     }{
4552       \_stex_reset:N \comp
4553       \_stex_reset:N \STEXInternalCurrentSymbolStr
4554     }
4555   }{
4556     \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_var://#1_prop}{arity}} = 0{
4557       \__stex_variables_invoke_math_ii:n {#1}
4558     }{
4559       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
4560     }

```

```

4561 }
4562 }
4563
4564 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4565   \cs_if_exist:cTF {
4566     stex_var_notation_#1_cs
4567   }{
4568     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4569       \_stex_reset:N \comp
4570       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4571       \_stex_reset:N \STEXInternalCurrentSymbolStr
4572       \bool_set_true:N \l_stex_allow_semantic_bool
4573     }
4574     \def\comp{\_varcomp}
4575     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4576     \bool_set_false:N \l_stex_allow_semantic_bool
4577     \use:c{stex_var_notation_#1_cs}
4578   }{
4579     \msg_error:nnxx{stex}{error/nonotation}{variable-#1}{s}
4580   }
4581 }
4582
4583 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4584   \exp_args:Nnx \use:nn {
4585     \def\comp{\_varcomp}
4586     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4587     \prop_clear:N \l__stex_terms_custom_args_prop
4588     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4589     \prop_get:cnN {
4590       l_stex_symdecl_var://#1 _prop
4591     }{ args } \l_tmpa_str
4592     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4593     \tl_set:Nn \arg { \_stex_terms_arg: }
4594     \str_if_empty:NTF \l_tmpa_str {
4595       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4596     }{
4597       \str_if_in:NnTF \l_tmpa_str b {
4598         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4599       }{
4600         \str_if_in:NnTF \l_tmpa_str B {
4601           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4602         }{
4603           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4604         }
4605       }
4606     }
4607     % TODO check that all arguments exist
4608   }{
4609     \_stex_reset:N \STEXInternalCurrentSymbolStr
4610     \_stex_reset:N \arg
4611     \_stex_reset:N \comp
4612     \_stex_reset:N \l__stex_terms_custom_args_prop
4613     %\bool_set_true:N \l_stex_allow_semantic_bool
4614   }

```

```
4615 }
```

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

30.5 Sequences

```
4616 <@@=stex_sequences>
4617
4618 \cs_new_protected:Nn \stex_invoke_sequence:n {
4619   \peek_charcode_remove:NTF ! {
4620     \stex_term_omv:nn {varseq://#1}{
4621       \exp_args:Nnx \use:nn {
4622         \def\comp{\_varcomp}
4623         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4624         \prop_item:cn{l_stex_symdecl_varseq://#1_prop}{notation}
4625       }{
4626         \_stex_reset:N \comp
4627         \_stex_reset:N \STEXInternalCurrentSymbolStr
4628       }
4629     }
4630   }{
4631     \bool_set_false:N \l_stex_allow_semantic_bool
4632     \def\comp{\_varcomp}
4633     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4634     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4635       \_stex_reset:N \comp
4636       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4637       \_stex_reset:N \STEXInternalCurrentSymbolStr
4638       \bool_set_true:N \l_stex_allow_semantic_bool
4639     }
4640     \use:c { stex_varseq_#1_cs }
4641   }
4642 }
4643 </package>
```


Chapter 31

STEX -Structural Features Implementation

```
4644 ⟨*package⟩
4645
4646 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4647
      Warnings and error messages
4648 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4649   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4650 }
4651 \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
4652   Symbol~#1~not~assigned~in~interpretmodule~#2
4653 }
4654
4655 \msg_new:nnn{stex}{error/unknownstructure}{
4656   No~structure~#1~found!
4657 }
4658
4659 \msg_new:nnn{stex}{error/unknownfield}{
4660   No~field~#1~in~instance~#2~found!\\#3
4661 }
4662
4663 \msg_new:nnn{stex}{error/keyval}{
4664   Invalid~key=value~pair~#1
4665 }
4666 \msg_new:nnn{stex}{error/instantiate/missing}{
4667   Assignments~missing~in~instantiate:~#1
4668 }
4669 \msg_new:nnn{stex}{error/incompatible}{
4670   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4671 }
4672
```

31.1 Imports with modification

```

4673 <@@=stex_copymodule>
4674 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4675   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4676     \tl_set:Nn \l_tmpa_tl { #1 }
4677     \__stex_copymodule_get_symbol_from_cs:
4678   }{
4679     % argument is a string
4680     % is it a command name?
4681     \cs_if_exist:cTF { #1 }{
4682       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4683       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4684       \str_if_empty:NTF \l_tmpa_str {
4685         \exp_args:Nx \cs_if_eq:NNTF {
4686           \tl_head:N \l_tmpa_tl
4687         } \stex_invoke_symbol:n {
4688           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4689         }{
4690           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4691         }
4692       } {
4693         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4694       }
4695     }{
4696       % argument is not a command name
4697       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4698       % \l_stex_all_symbols_seq
4699     }
4700   }
4701 }
4702
4703 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4704   \str_set:Nn \l_tmpa_str { #1 }
4705   \bool_set_false:N \l_tmpa_bool
4706   \bool_if:NF \l_tmpa_bool {
4707     \tl_set:Nn \l_tmpa_tl {
4708       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4709     }
4710     \str_set:Nn \l_tmpa_str { #1 }
4711     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4712     \seq_map_inline:Nn #2 {
4713       \str_set:Nn \l_tmpb_str { ##1 }
4714       \str_if_eq:eeT { \l_tmpa_str } {
4715         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4716       } {
4717         \seq_map_break:n {
4718           \tl_set:Nn \l_tmpa_tl {
4719             \str_set:Nn \l_stex_get_symbol_uri_str {
4720               ##1
4721             }
4722           }
4723         }
4724       }

```

```

4725     }
4726     \l_tmpa_tl
4727   }
4728 }
4729
4730 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4731   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4732     { \tl_tail:N \l_tmpa_tl }
4733   \tl_if_single:NTF \l_tmpa_tl {
4734     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4735       \exp_after:wN \str_set:Nn \exp_after:wN
4736         \l_stex_get_symbol_uri_str \l_tmpa_tl
4737       \__stex_copymodule_get_symbol_check:n { #1 }
4738     }{
4739       % TODO
4740       % tail is not a single group
4741     }
4742   }{
4743     % TODO
4744     % tail is not a single group
4745   }
4746 }
4747
4748 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4749   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4750     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4751       :~\seq_use:Nn #1 {,~}
4752     }
4753   }
4754 }
4755
4756 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4757   % import module
4758   \stex_import_module_uri:nn { #1 } { #2 }
4759   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4760   \stex_import_require_module:nnnn
4761     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4762     { \l_stex_import_path_str } { \l_stex_import_name_str }
4763
4764   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4765   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4766
4767   % fields
4768   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4769   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4770     \seq_map_inline:cn {c_stex_module_###1_constants}{
4771       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4772         ###1 ? ####1
4773       }
4774     }
4775   }
4776
4777   % setup prop
4778   \seq_clear:N \l_tmpa_seq

```

```

4779 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4780   name      = \l_stex_current_copymodule_name_str ,
4781   module    = \l_stex_current_module_str ,
4782   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4783   includes  = \l_tmpa_seq %,
4784 % fields    = \l_tmpa_seq
4785 }
4786 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4787   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4788 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4789 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4790
4791 \stex_if_do_html:T {
4792   \begin{stex_annotate_env} {#4} {
4793     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4794   }
4795   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
4796 }
4797 }
4798
4799 \cs_new_protected:Nn \stex_copymodule_end:n {
4800   % apply to every field
4801   \def \l_tmpa_cs ##1 ##2 {#1}
4802
4803   \tl_clear:N \__stex_copymodule_module_tl
4804   \tl_clear:N \__stex_copymodule_exec_tl
4805
4806   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4807   \seq_clear:N \__stex_copymodule_fields_seq
4808
4809   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4810     \seq_map_inline:cn {c_stex_module_##1_constants}{
4811
4812       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4813       \l_tmpa_cs{##1}{####1}
4814
4815       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4816         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4817         \stex_if_do_html:T {
4818           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4819             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4820           }
4821         }
4822       }{
4823         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / ##1 }
4824       }
4825
4826       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
4827       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4828       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4829
4830       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4831         \stex_if_do_html:T {
4832           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4833         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4834     }
4835 }
4836 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4837 }
4838
4839 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4840 \tl_put_right:Nx \__stex_copymodule_module_tl {
4841     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4842     \prop_set_from_keyval:cn {
4843         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4844     }{
4845         \prop_to_keyval:N \l_tmpa_prop
4846     }
4847 }
4848
4849 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4850     \stex_if_do_html:T {
4851         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4852             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4853         }
4854     }
4855     \tl_put_right:Nx \__stex_copymodule_module_tl {
4856         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4857             \stex_invoke_symbol:n {
4858                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4859             }
4860         }
4861     }
4862 }
4863
4864 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4865
4866 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4867     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4868 }
4869
4870 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4871     \stex_if_do_html:TF{
4872         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4873     }{
4874         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4875     }
4876 }
4877 }
4878 }
4879
4880
4881 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4882 \tl_put_left:Nx \__stex_copymodule_module_tl {
4883     \prop_set_from_keyval:cn {
4884         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4885     }{
4886         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4887   }
4888 }
4889
4890 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4891   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4892 }
4893
4894 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4895 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4896 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4897
4898 \__stex_copymodule_exec_tl
4899 \stex_if_do_html:T {
4900   \end{stex_annotate_env}
4901 }
4902 }
4903
4904 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4905   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4906   \stex_deactivate_macro:Nn \symdecl {module~environments}
4907   \stex_deactivate_macro:Nn \symdef {module~environments}
4908   \stex_deactivate_macro:Nn \notation {module~environments}
4909   \stex_reactivate_macro:N \assign
4910   \stex_reactivate_macro:N \renamedekl
4911   \stex_reactivate_macro:N \donotcopy
4912   \stex_smsmode_do:
4913 }{
4914   \stex_copymodule_end:n {}
4915 }
4916
4917 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4918   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4919   \stex_deactivate_macro:Nn \symdecl {module~environments}
4920   \stex_deactivate_macro:Nn \symdef {module~environments}
4921   \stex_deactivate_macro:Nn \notation {module~environments}
4922   \stex_reactivate_macro:N \assign
4923   \stex_reactivate_macro:N \renamedekl
4924   \stex_reactivate_macro:N \donotcopy
4925   \stex_smsmode_do:
4926 }{
4927   \stex_copymodule_end:n {
4928     \tl_if_exist:cF {
4929       l__stex_copymodule_copymodule_##1?##2_def_tl
4930     }{
4931       \str_if_eq:eeF {
4932         \prop_item:cn{
4933           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4934         }{ true }{
4935           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4936             ##1?##2
4937           }{\l_stex_current_copymodule_name_str}
4938         }
4939       }
4940     }

```

```

4941 }
4942
4943 \iffalse \begin{stex_annotate_env} \fi
4944 \NewDocumentEnvironment {realization} { 0 } { m } {
4945   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4946   \stex_deactivate_macro:Nn \symdecl {module~environments}
4947   \stex_deactivate_macro:Nn \symdef {module~environments}
4948   \stex_deactivate_macro:Nn \notation {module~environments}
4949   \stex_reactivate_macro:N \donotcopy
4950   \stex_reactivate_macro:N \assign
4951   \stex_smsmode_do:
4952 } {
4953   \stex_import_module_uri:nn { #1 } { #2 }
4954   \tl_clear:N \__stex_copymodule_exec_tl
4955   \tl_set:Nx \__stex_copymodule_module_tl {
4956     \stex_import_require_module:nnnn
4957     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4958     { \l_stex_import_path_str } { \l_stex_import_name_str }
4959   }
4960
4961   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4962     \seq_map_inline:cn {c_stex_module_##1_constants}{
4963       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4964       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4965         \stex_if_do_html:T {
4966           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4967             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4968               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4969             }
4970           }
4971         }
4972         \tl_put_right:Nx \__stex_copymodule_module_tl {
4973           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined } { true }
4974         }
4975       }
4976     }
4977
4978   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4979
4980   \__stex_copymodule_exec_tl
4981   \stex_if_do_html:T {\end{stex_annotate_env}}
4982 }
4983
4984 \NewDocumentCommand \donotcopy { m } {
4985   \str_clear:N \l_stex_import_name_str
4986   \str_set:Nn \l_tmpa_str { #1 }
4987   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4988   \seq_map_inline:Nn \l_stex_all_modules_seq {
4989     \str_set:Nn \l_tmpb_str { ##1 }
4990     \str_if_eq:eeT { \l_tmpa_str } {
4991       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4992     } {
4993       \seq_map_break:n {
4994         \stex_if_do_html:T {

```

```

4995         \stex_if_smsmode:F {
4996             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4997                 \stex_annotate:nnn{domain}{##1}{}}
4998         }
4999     }
5000 }
5001 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
5002 }
5003 }
5004 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
5005     \str_set:Nn \l_tmpb_str { #####1 }
5006     \str_if_eq:eeT { \l_tmpa_str } {
5007         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
5008     } {
5009         \seq_map_break:n {\seq_map_break:n {
5010             \stex_if_do_html:T {
5011                 \stex_if_smsmode:F {
5012                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
5013                         \stex_annotate:nnn{domain}{
5014                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
5015                         }{}
5016                     }
5017                 }
5018             }
5019             \str_set:Nx \l_stex_import_name_str {
5020                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
5021             }
5022         }}
5023     }
5024 }
5025 }
5026 \str_if_empty:NTF \l_stex_import_name_str {
5027     % TODO throw error
5028 }{
5029     \stex_collect_imports:n {\l_stex_import_name_str }
5030     \seq_map_inline:Nn \l_stex_collect_imports_seq {
5031         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
5032         \seq_map_inline:cn {c_stex_module_###1_constants}{
5033             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
5034             \bool_lazy_any:nT {
5035                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
5036                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
5037                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
5038             }{
5039                 % TODO throw error
5040             }
5041         }
5042     }
5043     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
5044     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
5045     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
5046 }
5047 \stex_smsmode_do:
5048 }

```



```

5049
5050 \NewDocumentCommand \assign { m m }{
5051   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
5052   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
5053   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
5054   \stex_smsmode_do:
5055 }
5056
5057 \keys_define:nn { stex / renamedecl } {
5058   name          .str_set_x:N = \l_stex_renamedecl_name_str
5059 }
5060 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
5061   \str_clear:N \l_stex_renamedecl_name_str
5062   \keys_set:nn { stex / renamedecl } { #1 }
5063 }
5064
5065 \NewDocumentCommand \renamedecl { O{} m m }{
5066   \__stex_copymodule_renamedecl_args:n { #1 }
5067   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
5068   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
5069   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
5070   \str_if_empty:NTF \l_stex_renamedecl_name_str {
5071     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5072       \l_stex_get_symbol_uri_str
5073     } }
5074   } {
5075     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
5076       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
5077       \prop_set_eq:cc {l_stex_symdecl_
5078         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5079         _prop
5080       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
5081       \seq_set_eq:cc {l_stex_symdecl_
5082         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5083         _notations
5084       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
5085       \prop_put:cnx {l_stex_symdecl_
5086         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5087         _prop
5088       }{ name }{ \l_stex_renamedecl_name_str }
5089       \prop_put:cnx {l_stex_symdecl_
5090         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5091         _prop
5092       }{ module }{ \l_stex_current_module_str }
5093       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
5094         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5095       }
5096       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
5097         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
5098       } }
5099   }
5100   \stex_smsmode_do:
5101 }
5102

```

```

5103 \stex_deactivate_macro:Nn \assign {copymodules}
5104 \stex_deactivate_macro:Nn \renamedekl {copymodules}
5105 \stex_deactivate_macro:Nn \donotcopy {copymodules}
5106
5107

```

31.2 The feature environment

`structural@feature (env.)`

```

5108 <@@=stex_features>
5109
5110 \NewDocumentEnvironment{structural_feature_module}{m m m}{
5111   \stex_if_in_module:F {
5112     \msg_set:nnn{stex}{error/nomodule}{
5113       Structural~Feature~has~to~occur~in~a~module:\\
5114       Feature~#2~of~type~#1\\
5115       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
5116     }
5117     \msg_error:nn{stex}{error/nomodule}
5118   }
5119
5120   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
5121
5122   \stex_module_setup:nn{meta=NONE}{#2 - #1}
5123
5124   \stex_if_do_html:T {
5125     \begin{stex_annotate_env}{feature:#1}{\l_stex_feature_parent_str ? #2 - #1}
5126     \stex_annotate_invisible:nnn{header}{#3}
5127   }
5128   {
5129     \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
5130     \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
5131     \stex_debug:nn{features}{
5132       Feature: \l_stex_last_feature_str
5133     }
5134     \stex_if_do_html:T {
5135       \end{stex_annotate_env}
5136     }
5137   }

```

31.3 Structure

`structure (env.)`

```

5138 <@@=stex_structures>
5139 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
5140   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
5141     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
5142   }
5143   \prop_gput:cxn {c_stex_module_ \l_stex_current_module_str _structures}
5144   {#1}{#2}
5145 }
5146

```

```

5147 \keys_define:nn { stex / features / structure } {
5148   name          .str_set_x:N = \l__stex_structures_name_str ,
5149 }
5150
5151 \cs_new_protected:Nn \__stex_structures_structure_args:n {
5152   \str_clear:N \l__stex_structures_name_str
5153   \keys_set:nn { stex / features / structure } { #1 }
5154 }
5155
5156 \NewDocumentEnvironment{mathstructure}{m O{}}{
5157   \__stex_structures_structure_args:n { #2 }
5158   \str_if_empty:NT \l__stex_structures_name_str {
5159     \str_set:Nx \l__stex_structures_name_str { #1 }
5160   }
5161   \stex_suppress_html:n {
5162     \bool_set_true:N \l_stex_symdecl_make_macro_bool
5163     \exp_args:Nx \stex_symdecl_do:nn {
5164       name = \l__stex_structures_name_str ,
5165       def = {\STEXsymbol{module-type}}{
5166         \STEXInternalTermMathOMSiiii {
5167           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5168             { ns } ?
5169           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
5170             { name } / \l__stex_structures_name_str - structure
5171         }{}{0}{}
5172       }}
5173     }{ #1 }
5174   }
5175   \exp_args:Nnnx
5176   \begin{structural_feature_module}{ structure }
5177     { \l__stex_structures_name_str }{}
5178   \stex_smsmode_do:
5179 }{
5180   \end{structural_feature_module}
5181   \stex_reset_up_to_module:n \l_stex_last_feature_str
5182   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
5183   \seq_clear:N \l_tmpa_seq
5184   \seq_map_inline:Nn \l_stex_collect_imports_seq {
5185     \seq_map_inline:cn{c_stex_module_##1_constants}{
5186       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
5187     }
5188   }
5189   \exp_args:Nnno
5190   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
5191   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
5192   \stex_add_structure_to_current_module:nn
5193     \l__stex_structures_name_str
5194     \l_stex_last_feature_str
5195
5196   \stex_execute_in_module:x {
5197     \tl_set:cn { #1 }{
5198       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
5199     }
5200   }

```

```

5201 }
5202
5203 \cs_new:Nn \stex_invoke_structure:nn {
5204   \stex_invoke_symbol:n { #1?#2 }
5205 }
5206
5207 \cs_new_protected:Nn \stex_get_structure:n {
5208   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
5209     \tl_set:Nn \l_tmpa_tl { #1 }
5210     \__stex_structures_get_from_cs:
5211   }{
5212     \cs_if_exist:cTF { #1 }{
5213       \cs_set_eq:Nc \l_tmpa_cs { #1 }
5214       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
5215       \str_if_empty:NTF \l_tmpa_str {
5216         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
5217           \__stex_structures_get_from_cs:
5218         }{
5219           \__stex_structures_get_from_string:n { #1 }
5220         }
5221       }{
5222         \__stex_structures_get_from_string:n { #1 }
5223       }
5224     }{
5225       \__stex_structures_get_from_string:n { #1 }
5226     }
5227   }
5228 }
5229
5230 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
5231   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
5232     { \tl_tail:N \l_tmpa_tl }
5233   \str_set:Nx \l_tmpa_str {
5234     \exp_after:wN \use_i:nn \l_tmpa_tl
5235   }
5236   \str_set:Nx \l_tmpb_str {
5237     \exp_after:wN \use_ii:nn \l_tmpa_tl
5238   }
5239   \str_set:Nx \l_stex_get_structure_str {
5240     \l_tmpa_str ? \l_tmpb_str
5241   }
5242   \str_set:Nx \l_stex_get_structure_module_str {
5243     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
5244   }
5245 }
5246
5247 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
5248   \tl_set:Nn \l_tmpa_tl {
5249     \msg_error:nnn{stex}{error/unknownstructure}{#1}
5250   }
5251   \str_set:Nn \l_tmpa_str { #1 }
5252   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
5253
5254   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

5255 \prop_if_exist:cT {c_stex_module_##1_structures} {
5256 \prop_map_inline:cn {c_stex_module_##1_structures} {
5257 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
5258 \prop_map_break:n{\seq_map_break:n{
5259 \tl_set:Nn \l_tmpa_tl {
5260 \str_set:Nn \l_stex_get_structure_str {##1?####1}
5261 \str_set:Nn \l_stex_get_structure_module_str {####2}
5262 }
5263 }}
5264 }
5265 }
5266 }
5267 }
5268 \l_tmpa_tl
5269 }

```

\instantiate

```

5270
5271 \keys_define:nn { stex / instantiate } {
5272 name .str_set_x:N = \l__stex_structures_name_str
5273 }
5274 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
5275 \str_clear:N \l__stex_structures_name_str
5276 \keys_set:nn { stex / instantiate } { #1 }
5277 }
5278
5279 \NewDocumentCommand \instantiate {m O{} m m O{}}{
5280 \beginingroup
5281 \stex_get_structure:n {#3}
5282 \__stex_structures_instantiate_args:n { #2 }
5283 \str_if_empty:NT \l__stex_structures_name_str {
5284 \str_set:Nn \l__stex_structures_name_str { #1 }
5285 }
5286 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5287 \seq_clear:N \l__stex_structures_fields_seq
5288 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5289 \seq_map_inline:Nn \l_stex_collect_imports_seq {
5290 \seq_map_inline:cn {c_stex_module_##1_constants}{
5291 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
5292 }
5293 }
5294
5295 \tl_if_empty:nF{#5}{
5296 \seq_set_split:Nnn \l_tmpa_seq , {#5}
5297 \prop_clear:N \l_tmpa_prop
5298 \seq_map_inline:Nn \l_tmpa_seq {
5299 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5300 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5301 \msg_error:nnn{stex}{error/keyval}{##1}
5302 }
5303 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
5304 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5305 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
5306 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```

```

5307     \exp_args:Nxx \str_if_eq:nnF
5308     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5309     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
5310     \msg_error:nnxxxx{stex}{error/incompatible}
5311     {\l__stex_structures_dom_str}
5312     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
5313     {\l_stex_get_symbol_uri_str}
5314     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
5315   }
5316   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
5317 }
5318 }
5319
5320 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5321   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
5322   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
5323
5324   \stex_add_constant_to_current_module:n {\l_tmpa_str}
5325   \stex_execute_in_module:x {
5326     \prop_set_from_keyval:cn { l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _p
5327     name = \l_tmpa_str ,
5328     args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5329     arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5330     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5331     argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}}
5332   }
5333   \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
5334 }
5335
5336 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5337   \stex_find_notation:nn{##1}{}
5338   \stex_execute_in_module:x {
5339     \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
5340   }
5341
5342   \stex_copy_control_sequence_ii:ccN
5343   {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
5344   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5345   \l_tmpa_tl
5346   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
5347
5348
5349   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5350     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
5351     \stex_execute_in_module:x {
5352       \tl_set:cn
5353       {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
5354       { \exp_args:No \exp_not:n \l_tmpa_cs}
5355     }
5356   }
5357
5358 }
5359
5360 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur

```

```

5361 }
5362
5363 \stex_execute_in_module:x {
5364   \prop_set_from_keyval:cn {l_stex_instance_ \l_stex_current_module_str? \l__stex_structur
5365     domain = \l_stex_get_structure_module_str ,
5366     \prop_to_keyval:N \l_tmpa_prop
5367   }
5368   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str? \l__stex_structur
5369 }
5370 \stex_debug:nn{instantiate}{
5371   Instance~\l_stex_current_module_str? \l__stex_structures_name_str \
5372   \prop_to_keyval:N \l_tmpa_prop
5373 }
5374 \exp_args:Nxx \stex_symdecl_do:nn {
5375   type={\STEXsymbol{module-type}{
5376     \STEXInternalTermMathOMSiiii {
5377       \l_stex_get_structure_module_str
5378     }{}{0}{}}
5379   }}
5380 }{\l__stex_structures_name_str}
5381 % {
5382   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str? \l__stex_structures
5383   \tl_set:Nn \l_stex_notation_after_do_tl {\_stex_notation_final:}
5384   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
5385 % }
5386 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
5387 \endgroup
5388 \stex_smsmode_do:\ignorespacesandpars
5389 }
5390
5391 \cs_new_protected:Nn \stex_symbol_or_var:n {
5392   \cs_if_exist:cTF{#1}{
5393     \cs_set_eq:Nc \l_tmpa_tl { #1 }
5394     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
5395     \str_if_empty:NTF \l_tmpa_str {
5396       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
5397       \stex_invoke_variable:n {
5398         \bool_set_true:N \l_stex_symbol_or_var_bool
5399         \bool_set_false:N \l_stex_instance_or_symbol_bool
5400         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5401         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5402         \str_set:Nx \l_stex_get_symbol_uri_str {
5403           \exp_after:wN \use:n \l_tmpa_tl
5404         }
5405       }{ % TODO \stex_invoke_varinstance:n
5406         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
5407           \bool_set_true:N \l_stex_symbol_or_var_bool
5408           \bool_set_true:N \l_stex_instance_or_symbol_bool
5409           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
5410           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
5411           \str_set:Nx \l_stex_get_symbol_uri_str {
5412             \exp_after:wN \use:n \l_tmpa_tl
5413           }
5414         }{

```

```

5415         \bool_set_false:N \l_stex_symbol_or_var_bool
5416         \stex_get_symbol:n{#1}
5417     }
5418 }
5419 }{
5420     \__stex_structures_symbolorvar_from_string:n{ #1 }
5421 }
5422 }{
5423     \__stex_structures_symbolorvar_from_string:n{ #1 }
5424 }
5425 }
5426
5427 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
5428     \prop_if_exist:cTF {l_stex_symdecl_var://#1 _prop}{
5429         \bool_set_true:N \l_stex_symbol_or_var_bool
5430         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
5431     }{
5432         \bool_set_false:N \l_stex_symbol_or_var_bool
5433         \stex_get_symbol:n{#1}
5434     }
5435 }
5436
5437 \keys_define:nn { stex / varinstantiate } {
5438     name .str_set_x:N = \l__stex_structures_name_str,
5439     bind .choices:nn =
5440         {forall,exists}
5441         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
5442 }
5443
5444 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
5445     \str_clear:N \l__stex_structures_name_str
5446     \str_clear:N \l__stex_structures_bind_str
5447     \keys_set:nn { stex / varinstantiate } { #1 }
5448 }
5449
5450 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
5451     \beginingroup
5452         \stex_get_structure:n {#3}
5453         \__stex_structures_varinstantiate_args:n { #2 }
5454         \str_if_empty:NT \l__stex_structures_name_str {
5455             \str_set:Nn \l__stex_structures_name_str { #1 }
5456         }
5457         \stex_if_do_html:TF{
5458             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
5459         }{\use:n}
5460         {
5461             \stex_if_do_html:T{
5462                 \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
5463             }
5464             \seq_clear:N \l__stex_structures_fields_seq
5465             \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
5466             \seq_map_inline:Nn \l_stex_collect_imports_seq {
5467                 \seq_map_inline:cn {c_stex_module_##1_constants}{
5468                     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }

```



```

5469     }
5470 }
5471 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
5472 \prop_clear:N \l_tmpa_prop
5473 \tl_if_empty:nF {#5} {
5474     \seq_set_split:Nnn \l_tmpa_seq , {#5}
5475     \seq_map_inline:Nn \l_tmpa_seq {
5476         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
5477         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
5478             \msg_error:nnn{stex}{error/keyval}{##1}
5479         }
5480         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
5481         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
5482         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq:nn
5483         \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
5484         \stex_if_do_html:T{
5485             \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
5486                 \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{\l_stex_get_symbol_uri_str}{}}
5487         }
5488         \bool_if:NTF \l_stex_symbol_or_var_bool {
5489             \exp_args:Nxx \str_if_eq:nnF
5490                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5491                 {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}{
5492                 \msg_error:nnxxx{stex}{error/incompatible}
5493                 {\l__stex_structures_dom_str}
5494                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5495                 {\l_stex_get_symbol_uri_str}
5496                 {\prop_item:cn{l_stex_symdecl_var://\l_stex_get_symbol_uri_str _prop}{args}}
5497             }
5498             \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
5499             }}{
5500             \exp_args:Nxx \str_if_eq:nnF
5501                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5502                 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
5503                 \msg_error:nnxxx{stex}{error/incompatible}
5504                 {\l__stex_structures_dom_str}
5505                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
5506                 {\l_stex_get_symbol_uri_str}
5507                 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
5508             }
5509             \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
5510             }}
5511         }
5512     }
5513     \tl_gclear:N \g__stex_structures_aftergroup_tl
5514     \seq_map_inline:Nn \l__stex_structures_fields_seq {
5515         \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l__stex_structures_name_str} {\stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}}
5516         \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5517         \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5518             \stex_find_notation:nn{##1}{
5519                 \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
5520                     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5521                 \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
5522                 \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{

```

```

5523         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5524         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
5525         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
5526     }
5527 }
5528
5529 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5530     \prop_set_from_keyval:cn {l_stex_symdecl_ var://\l_tmpa_str_prop}{
5531         name = \l_tmpa_str ,
5532         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5533         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5534         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs} ,
5535         argnames = {\prop_item:cn {l_stex_symdecl_##1_prop}{argnames}} ,
5536     }
5537     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
5538     {g__stex_structures_tmpa_\l_tmpa_str_cs}
5539     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
5540     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
5541 }
5542 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5543 }
5544 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5545     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
5546         domain = \l_stex_get_structure_module_str ,
5547         \prop_to_keyval:N \l_tmpa_prop
5548     }
5549     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5550     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
5551         \exp_args:NNx \exp_not:N \use:nn {
5552             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5553             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5554                 \exp_not:n{
5555                     \_varcomp{#4}
5556                 }
5557             }
5558         }}{
5559             \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5560         }
5561     }
5562 }
5563 }
5564 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{g__stex_structures_a
5565 \aftergroup\g__stex_structures_aftergroup_tl
5566 \endgroup
5567 \stex_smsmode_do:\ignorespacesandpars
5568 }
5569
5570 \cs_new_protected:Nn \stex_invoke_instance:n {
5571     \peek_charcode_remove:NTF ! {
5572         \stex_invoke_symbol:n{#1}
5573     }{
5574         \_stex_invoke_instance:nn {#1}
5575     }
5576 }

```

```

5577
5578
5579 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5580   \peek_charcode_remove:NTF ! {
5581     \exp_args:Nnx \use:nn {
5582       \def\comp{\_varcomp}
5583       \use:c{l_stex_varinstance_#1_op_tl}
5584     }{
5585       \_stex_reset:N \comp
5586     }
5587   }{
5588     \_stex_invoke_varinstance:nn {#1}
5589   }
5590 }
5591
5592 \cs_new_protected:Nn \stex_invoke_instance:nn {
5593   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5594     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5595   }{
5596     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5597     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5598       \prop_to_keyval:N \l_tmpa_prop
5599     }
5600   }
5601 }
5602
5603 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
5604   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5605     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5606     \l_tmpa_tl
5607   }{
5608     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
5609     }
5610   }

```

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

\stex_invoke_structure:nnn

```

5611 % #1: URI of the instance
5612 % #2: URI of the instantiated module
5613 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5614   \tl_if_empty:nTF{ #3 }{
5615     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5616       c_stex_feature_ #2 _prop
5617     }
5618     \tl_clear:N \l_tmpa_tl
5619     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5620     \seq_map_inline:Nn \l_tmpa_seq {
5621       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5622       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5623       \cs_if_exist:cT {
5624         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5625       }{
5626         \tl_if_empty:NF \l_tmpa_tl {

```

```

5627         \tl_put_right:Nn \l_tmpa_tl {,}
5628     }
5629     \tl_put_right:Nx \l_tmpa_tl {
5630         \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5631     }
5632 }
5633 }
5634 \exp_args:No \mathstruct \l_tmpa_tl
5635 }{
5636     \stex_invoke_symbol:n{#1/#3}
5637 }
5638 }

```

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

```

5639 </package>

```

Chapter 32

STEX -Statements Implementation

```
5640 <*package>
5641
5642 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5643
5644 <@@=stex_statements>
    Warnings and error messages
5645
\titleemph
5646 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

32.1 Definitions

definiendum

```
5647 \keys_define:nn {stex / definiendum }{
5648   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5649   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5650   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5651   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5652 }
5653 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5654   \str_clear:N \l__stex_statements_definiendum_root_str
5655   \tl_clear:N \l__stex_statements_definiendum_post_tl
5656   \str_clear:N \l__stex_statements_definiendum_gfa_str
5657   \keys_set:nn { stex / definiendum }{ #1 }
5658 }
5659 \NewDocumentCommand \definiendum { O{} m m } {
5660   \__stex_statements_definiendum_args:n { #1 }
5661   \stex_get_symbol:n { #2 }
5662   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5663   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5664     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5665     \tl_set:Nn \l_tmpa_tl { #3 }
5666   } {
5667     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5668     \tl_set:Nn \l_tmpa_tl {
5669       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5670     }
5671   }
5672 } {
5673   \tl_set:Nn \l_tmpa_tl { #3 }
5674 }
5675
5676 % TODO root
5677 \stex_html_backend:TF {
5678   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5679 } {
5680   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5681 }
5682 }
5683 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

5684
5685 \NewDocumentCommand \definame { 0{ } m } {
5686   \__stex_statements_definiendum_args:n { #1 }
5687   % TODO: root
5688   \stex_get_symbol:n { #2 }
5689   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5690   \str_set:Nx \l_tmpa_str {
5691     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5692   }
5693   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5694   \stex_html_backend:TF {
5695     \stex_if_do_html:T {
5696       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5697         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5698       }
5699     }
5700   } {
5701     \exp_args:Nnx \defemph@uri {
5702       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5703     } { \l_stex_get_symbol_uri_str }
5704   }
5705 }
5706 \stex_deactivate_macro:Nn \definame {definition~environments}
5707
5708 \NewDocumentCommand \Definame { 0{ } m } {
5709   \__stex_statements_definiendum_args:n { #1 }
5710   \stex_get_symbol:n { #2 }
5711   \str_set:Nx \l_tmpa_str {
5712     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5713   }
5714   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5715 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5716 \stex_html_backend:TF {
5717   \stex_if_do_html:T {
5718     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5719       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5720     }
5721   }
5722 } {
5723   \exp_args:Nnx \defemph@uri {
5724     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5725   } { \l_stex_get_symbol_uri_str }
5726 }
5727 }
5728 \stex_deactivate_macro:Nn \Definame {definition-environments}
5729
5730 \NewDocumentCommand \premise { m }{
5731   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5732 }
5733 \NewDocumentCommand \conclusion { m }{
5734   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5735 }
5736 \NewDocumentCommand \definiens { 0{} m }{
5737   \str_clear:N \l_stex_get_symbol_uri_str
5738   \tl_if_empty:nF {#1} {
5739     \stex_get_symbol:n { #1 }
5740   }
5741   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5742     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5743       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5744     }{
5745       % TODO throw error
5746     }
5747   }
5748   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5749   {\l_stex_current_module_str}{
5750     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5751   }{true}{
5752     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5753     \exp_args:Nx \stex_add_to_current_module:n {
5754       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5755     }
5756   }
5757 }
5758 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5759 }
5760
5761 \NewDocumentCommand \varbindforall {m}{
5762   \stex_symbol_or_var:n {#1}
5763   \bool_if:NTF\l_stex_symbol_or_var_bool{
5764     \stex_if_do_html:T {
5765       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5766     }
5767   }{
5768     % todo throw error

```

```

5769 }
5770 }
5771
5772 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5773 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5774 \stex_deactivate_macro:Nn \definiens {definition~environments}
5775 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5776

```

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

sdefinition (env.)

```

5777
5778 \keys_define:nn {stex / sdefinition }{
5779   type      .str_set_x:N = \sdefinitiontype,
5780   id        .str_set_x:N = \sdefinitionid,
5781   name      .str_set_x:N = \sdefinitionname,
5782   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5783   title     .tl_set:N    = \sdefinitiontitle
5784 }
5785 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5786   \str_clear:N \sdefinitiontype
5787   \str_clear:N \sdefinitionid
5788   \str_clear:N \sdefinitionname
5789   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5790   \tl_clear:N \sdefinitiontitle
5791   \keys_set:nn { stex / sdefinition }{ #1 }
5792 }
5793
5794 \NewDocumentEnvironment{sdefinition}{0{}}{
5795   \__stex_statements_sdefinition_args:n{ #1 }
5796   \stex_reactivate_macro:N \definiendum
5797   \stex_reactivate_macro:N \definame
5798   \stex_reactivate_macro:N \Definame
5799   \stex_reactivate_macro:N \premise
5800   \stex_reactivate_macro:N \definiens
5801   \stex_reactivate_macro:N \varbindforall
5802   \stex_if_smsmode:F{
5803     \seq_clear:N \l_tmpb_seq
5804     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5805       \tl_if_empty:nF{ ##1 }{
5806         \stex_get_symbol:n { ##1 }
5807         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5808           \l_stex_get_symbol_uri_str
5809         }
5810       }
5811     }
5812     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5813     \exp_args:Nnnx
5814     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5815     \str_if_empty:NF \sdefinitiontype {
5816       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5817     }
5818     \str_if_empty:NF \sdefinitionname {

```



```

5819     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5820   }
5821   \clist_set:No \l_tmpa_clist \sdefinitiontype
5822   \tl_clear:N \l_tmpa_tl
5823   \clist_map_inline:Nn \l_tmpa_clist {
5824     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5825       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5826     }
5827   }
5828   \tl_if_empty:NTF \l_tmpa_tl {
5829     \__stex_statements_sdefinition_start:
5830   }{
5831     \l_tmpa_tl
5832   }
5833 }
5834 \stex_ref_new_doc_target:n \sdefinitionid
5835 \stex_smsmode_do:
5836 }{
5837   \stex_suppress_html:n {
5838     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5839   }
5840   \stex_if_smsmode:F {
5841     \clist_set:No \l_tmpa_clist \sdefinitiontype
5842     \tl_clear:N \l_tmpa_tl
5843     \clist_map_inline:Nn \l_tmpa_clist {
5844       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5845         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5846       }
5847     }
5848     \tl_if_empty:NTF \l_tmpa_tl {
5849       \__stex_statements_sdefinition_end:
5850     }{
5851       \l_tmpa_tl
5852     }
5853     \end{stex_annotate_env}
5854   }
5855 }

```

\stexpatchdefinition

```

5856 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5857   \stex_par:\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5858     ~(\sdefinitiontitle)
5859   }~}
5860 }
5861 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5862
5863 \newcommand\stexpatchdefinition[3]{} {
5864   \str_set:Nx \l_tmpa_str{ #1 }
5865   \str_if_empty:NTF \l_tmpa_str {
5866     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5867     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5868   }{
5869     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5870     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5871     }
5872 }

```

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

`\inlinedef` inline:

```

5873 \keys_define:nn {stex / inlinedef }{
5874   type      .str_set_x:N = \sdefinitiontype,
5875   id        .str_set_x:N = \sdefinitionid,
5876   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5877   name      .str_set_x:N = \sdefinitionname
5878 }
5879 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5880   \str_clear:N \sdefinitiontype
5881   \str_clear:N \sdefinitionid
5882   \str_clear:N \sdefinitionname
5883   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5884   \keys_set:nn { stex / inlinedef }{ #1 }
5885 }
5886 \NewDocumentCommand \inlinedef { 0{} m } {
5887   \begin{group}
5888     \__stex_statements_inlinedef_args:n{ #1 }
5889     \stex_reactivate_macro:N \definiendum
5890     \stex_reactivate_macro:N \definame
5891     \stex_reactivate_macro:N \Definame
5892     \stex_reactivate_macro:N \premise
5893     \stex_reactivate_macro:N \definiens
5894     \stex_reactivate_macro:N \varbindforall
5895     \stex_ref_new_doc_target:n \sdefinitionid
5896     \stex_if_smsmode:TF{\stex_suppress_html:n {
5897       \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5898     }}{
5899       \seq_clear:N \l_tmpb_seq
5900       \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5901         \tl_if_empty:nF{ ##1 }{
5902           \stex_get_symbol:n { ##1 }
5903           \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5904             \l_stex_get_symbol_uri_str
5905           }
5906         }
5907       }
5908       \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5909       \exp_args:Nnx
5910       \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5911         \str_if_empty:NF \sdefinitiontype {
5912           \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5913         }
5914         #2
5915         \str_if_empty:NF \sdefinitionname {
5916           \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5917           \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5918         }
5919       }
5920     }

```

```

5921 \endgroup
5922 \stex_smsmode_do:
5923 }

```

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

32.2 Assertions

`sassertion (env.)`

```

5924
5925 \keys_define:nn {stex / sassertion }{
5926   type      .str_set_x:N = \sassertiontype,
5927   id         .str_set_x:N = \sassertionid,
5928   title      .tl_set:N   = \sassertiontitle ,
5929   for        .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5930   name       .str_set_x:N = \sassertionname
5931 }
5932 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5933   \str_clear:N \sassertiontype
5934   \str_clear:N \sassertionid
5935   \str_clear:N \sassertionname
5936   \clist_clear:N \l__stex_statements_sassertion_for_clist
5937   \tl_clear:N \sassertiontitle
5938   \keys_set:nn { stex / sassertion }{ #1 }
5939 }
5940
5941 %\tl_new:N \g__stex_statements_aftergroup_tl
5942
5943 \NewDocumentEnvironment{sassertion}{0{}}{
5944   \__stex_statements_sassertion_args:n{ #1 }
5945   \stex_reactivate_macro:N \premise
5946   \stex_reactivate_macro:N \conclusion
5947   \stex_reactivate_macro:N \varbindforall
5948   \stex_if_smsmode:F {
5949     \seq_clear:N \l_tmpb_seq
5950     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5951       \tl_if_empty:nF{ ##1 }{
5952         \stex_get_symbol:n { ##1 }
5953         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5954           \l_stex_get_symbol_uri_str
5955         }
5956       }
5957     }
5958     \exp_args:Nnnx
5959     \begin{sassertion}{\seq_use:Nn \l_tmpb_seq {,}}
5960     \str_if_empty:NF \sassertiontype {
5961       \stex_annotate_invisible:nnn{type}{\sassertiontype}{\sassertiontype}{}
5962     }
5963     \str_if_empty:NF \sassertionname {
5964       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{\sassertionname}{}
5965     }
5966     \clist_set:N \l_tmpa_clist \sassertiontype
5967     \tl_clear:N \l_tmpa_tl

```

```

5968 \clist_map_inline:Nn \l_tmpa_clist {
5969   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5970     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5971   }
5972 }
5973 \tl_if_empty:NTF \l_tmpa_tl {
5974   \__stex_statements_sassertion_start:
5975 }{
5976   \l_tmpa_tl
5977 }
5978 }
5979 \str_if_empty:NTF \sassertionid {
5980   \str_if_empty:NF \sassertionname {
5981     \stex_ref_new_doc_target:n {}
5982   }
5983 } {
5984   \stex_ref_new_doc_target:n \sassertionid
5985 }
5986 \stex_smsmode_do:
5987 ){
5988   \str_if_empty:NF \sassertionname {
5989     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sassertionname}}
5990     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5991   }
5992   \stex_if_smsmode:F {
5993     \clist_set:No \l_tmpa_clist \sassertiontype
5994     \tl_clear:N \l_tmpa_tl
5995     \clist_map_inline:Nn \l_tmpa_clist {
5996       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5997         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5998       }
5999     }
6000     \tl_if_empty:NTF \l_tmpa_tl {
6001       \__stex_statements_sassertion_end:
6002     }{
6003       \l_tmpa_tl
6004     }
6005     \end{stex_annotate_env}
6006   }
6007 }

```

\stexpatchassertion

```

6008
6009 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
6010   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
6011     (\sassertiontitle)
6012   }~}
6013 }
6014 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
6015
6016 \newcommand\stexpatchassertion[3] [] {
6017   \str_set:Nx \l_tmpa_str{ #1 }
6018   \str_if_empty:NTF \l_tmpa_str {
6019     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

6020     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
6021   }{
6022     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
6023     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
6024   }
6025 }

```

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

`\inlineass` inline:

```

6026 \keys_define:nn {stex / inlineass }{
6027   type      .str_set_x:N = \sassertiontype,
6028   id        .str_set_x:N = \sassertionid,
6029   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
6030   name      .str_set_x:N = \sassertionname
6031 }
6032 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
6033   \str_clear:N \sassertiontype
6034   \str_clear:N \sassertionid
6035   \str_clear:N \sassertionname
6036   \clist_clear:N \l__stex_statements_sassertion_for_clist
6037   \keys_set:nn { stex / inlineass }{ #1 }
6038 }
6039 \NewDocumentCommand \inlineass { 0{} m } {
6040   \beginngroup
6041   \stex_reactivate_macro:N \premise
6042   \stex_reactivate_macro:N \conclusion
6043   \stex_reactivate_macro:N \varbindforall
6044   \__stex_statements_inlineass_args:n{ #1 }
6045   \str_if_empty:NTF \sassertionid {
6046     \str_if_empty:NF \sassertionname {
6047       \stex_ref_new_doc_target:n {}
6048     }
6049   } {
6050     \stex_ref_new_doc_target:n \sassertionid
6051   }
6052
6053   \stex_if_smsmode:TF{
6054     \str_if_empty:NF \sassertionname {
6055       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
6056       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
6057     }
6058   }{
6059     \seq_clear:N \l_tmpb_seq
6060     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
6061       \tl_if_empty:nF{ ##1 }{
6062         \stex_get_symbol:n { ##1 }
6063         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6064           \l_stex_get_symbol_uri_str
6065         }
6066       }
6067     }
6068     \exp_args:Nnx
6069     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{

```

```

6070     \str_if_empty:NF \sassassertiontype {
6071       \stex_annotate_invisible:nnn{typestrings}{\sassassertiontype}{ }
6072     }
6073     #2
6074     \str_if_empty:NF \sassassertionname {
6075       \stex_suppress_html:n{\stex_symdecl_do:nn{ }{\sassassertionname}}
6076       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassassertionname}
6077       \stex_annotate_invisible:nnn{statementname}{\sassassertionname}{ }
6078     }
6079   }
6080 }
6081 \endgroup
6082 \stex_smsmode_do:
6083 }

```

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

32.3 Examples

`sexample (env.)`

```

6084
6085 \keys_define:nn {stex / sexample }{
6086   type      .str_set_x:N = \exampletype,
6087   id        .str_set_x:N = \sexampleid,
6088   title     .tl_set:N    = \sexamplename,
6089   name      .str_set_x:N = \sexamplename ,
6090   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
6091 }
6092 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
6093   \str_clear:N \sexampletype
6094   \str_clear:N \sexampleid
6095   \str_clear:N \sexamplename
6096   \tl_clear:N \sexamplename
6097   \clist_clear:N \l__stex_statements_sexample_for_clist
6098   \keys_set:nn { stex / sexample }{ #1 }
6099 }
6100
6101 \NewDocumentEnvironment{sexample}{0{}}{
6102   \__stex_statements_sexample_args:n{ #1 }
6103   \stex_reactivate_macro:N \premise
6104   \stex_reactivate_macro:N \conclusion
6105   \stex_if_smsmode:F {
6106     \seq_clear:N \l_tmpb_seq
6107     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6108       \tl_if_empty:nF{ ##1 }{
6109         \stex_get_symbol:n { ##1 }
6110         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6111           \l_stex_get_symbol_uri_str
6112         }
6113       }
6114     }
6115     \exp_args:Nnnx
6116     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```

```

6117 \str_if_empty:NF \sexamplotype {
6118 \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}}
6119 }
6120 \str_if_empty:NF \sexamplename {
6121 \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}}
6122 }
6123 \clist_set:No \l_tmpa_clist \sexamplotype
6124 \tl_clear:N \l_tmpa_tl
6125 \clist_map_inline:Nn \l_tmpa_clist {
6126 \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
6127 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
6128 }
6129 }
6130 \tl_if_empty:NTF \l_tmpa_tl {
6131 \__stex_statements_sexample_start:
6132 }{
6133 \l_tmpa_tl
6134 }
6135 }
6136 \str_if_empty:NF \sexampleid {
6137 \stex_ref_new_doc_target:n \sexampleid
6138 }
6139 \stex_smsmode_do:
6140 }{
6141 \str_if_empty:NF \sexamplename {
6142 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6143 }
6144 \stex_if_smsmode:F {
6145 \clist_set:No \l_tmpa_clist \sexamplotype
6146 \tl_clear:N \l_tmpa_tl
6147 \clist_map_inline:Nn \l_tmpa_clist {
6148 \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
6149 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
6150 }
6151 }
6152 \tl_if_empty:NTF \l_tmpa_tl {
6153 \__stex_statements_sexample_end:
6154 }{
6155 \l_tmpa_tl
6156 }
6157 \end{stex_annotate_env}
6158 }
6159 }

```

\stexpatchexample

```

6160
6161 \cs_new_protected:Nn \__stex_statements_sexample_start: {
6162 \stex_par:\noindent\titleemph{Example~\tl_if_empty:NF \sexampltitle {
6163 (\sexampltitle)
6164 }~}
6165 }
6166 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
6167
6168 \newcommand\stexpatchexample[3]{} {

```

```

6169 \str_set:Nx \l_tmpa_str{ #1 }
6170 \str_if_empty:NTF \l_tmpa_str {
6171   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
6172   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
6173 }{
6174   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
6175   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
6176 }
6177 }

```

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

`\inlineex` inline:

```

6178 \keys_define:nn {stex / inlineex }{
6179   type      .str_set_x:N = \sexamplotype,
6180   id        .str_set_x:N = \sexampleid,
6181   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
6182   name      .str_set_x:N = \sexamplename
6183 }
6184 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
6185   \str_clear:N \sexamplotype
6186   \str_clear:N \sexampleid
6187   \str_clear:N \sexamplename
6188   \clist_clear:N \l__stex_statements_sexample_for_clist
6189   \keys_set:nn { stex / inlineex }{ #1 }
6190 }
6191 \NewDocumentCommand \inlineex { 0{} m } {
6192   \begingroup
6193   \stex_reactivate_macro:N \premise
6194   \stex_reactivate_macro:N \conclusion
6195   \__stex_statements_inlineex_args:n{ #1 }
6196   \str_if_empty:NF \sexampleid {
6197     \stex_ref_new_doc_target:n \sexampleid
6198   }
6199   \stex_if_smsmode:TF{
6200     \str_if_empty:NF \sexamplename {
6201       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
6202   }
6203 }{
6204   \seq_clear:N \l_tmpb_seq
6205   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
6206     \tl_if_empty:nF{ ##1 }{
6207       \stex_get_symbol:n { ##1 }
6208       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6209         \l_stex_get_symbol_uri_str
6210       }
6211     }
6212   }
6213   \exp_args:Nnx
6214   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
6215     \str_if_empty:NF \sexamplotype {
6216       \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
6217     }
6218   #2

```



```

6219     \str_if_empty:NF \sexamplename {
6220       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
6221       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
6222     }
6223   }
6224 }
6225 \endgroup
6226 \stex_smsmode_do:
6227 }

```

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

32.4 Logical Paragraphs

`sparagraph` (*env.*)

```

6228 \keys_define:nn { stex / sparagraph } {
6229   id      .str_set_x:N = \sparagraphid ,
6230   title   .tl_set:N   = \l_stex_sparagraph_title_tl ,
6231   type    .str_set_x:N = \sparagraphtype ,
6232   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6233   from    .tl_set:N   = \sparagraphfrom ,
6234   to      .tl_set:N   = \sparagraphto ,
6235   start   .tl_set:N   = \l_stex_sparagraph_start_tl ,
6236   name    .str_set:N   = \sparagraphname ,
6237   imports .tl_set:N   = \l__stex_statements_sparagraph_imports_tl
6238 }
6239
6240 \cs_new_protected:Nn \stex_sparagraph_args:n {
6241   \tl_clear:N \l_stex_sparagraph_title_tl
6242   \tl_clear:N \sparagraphfrom
6243   \tl_clear:N \sparagraphto
6244   \tl_clear:N \l_stex_sparagraph_start_tl
6245   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
6246   \str_clear:N \sparagraphid
6247   \str_clear:N \sparagraphtype
6248   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6249   \str_clear:N \sparagraphname
6250   \keys_set:nn { stex / sparagraph } { #1 }
6251 }
6252 \newif\if@in@omtext\@in@omtextfalse
6253
6254 \NewDocumentEnvironment {sparagraph} { 0{ } } {
6255   \stex_sparagraph_args:n { #1 }
6256   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6257     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
6258   }{
6259     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
6260   }
6261   \@in@omtexttrue
6262   \stex_if_smsmode:F {
6263     \seq_clear:N \l_tmpb_seq
6264     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6265       \tl_if_empty:nF{ ##1 }{

```

```

6266     \stex_get_symbol:n { ##1 }
6267     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6268         \l_stex_get_symbol_uri_str
6269     }
6270 }
6271 }
6272 \exp_args:Nnnx
6273 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
6274 \str_if_empty:NF \sparagraphtype {
6275     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
6276 }
6277 \str_if_empty:NF \sparagraphfrom {
6278     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6279 }
6280 \str_if_empty:NF \sparagraphto {
6281     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6282 }
6283 \str_if_empty:NF \sparagraphname {
6284     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6285 }
6286 \clist_set:No \l_tmpa_clist \sparagraphtype
6287 \tl_clear:N \l_tmpa_tl
6288 \clist_map_inline:Nn \sparagraphtype {
6289     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
6290         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
6291     }
6292 }
6293 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
6294 \tl_if_empty:NTF \l_tmpa_tl {
6295     \__stex_statements_sparagraph_start:
6296 }{
6297     \l_tmpa_tl
6298 }
6299 }
6300 \clist_set:No \l_tmpa_clist \sparagraphtype
6301 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6302     {
6303         \stex_reactivate_macro:N \definiendum
6304         \stex_reactivate_macro:N \definame
6305         \stex_reactivate_macro:N \Definame
6306         \stex_reactivate_macro:N \premise
6307         \stex_reactivate_macro:N \definiens
6308     }
6309 \str_if_empty:NTF \sparagraphid {
6310     \str_if_empty:NTF \sparagraphname {
6311         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6312             \stex_ref_new_doc_target:n {}
6313         }
6314     } {
6315         \stex_ref_new_doc_target:n {}
6316     }
6317 } {
6318     \stex_ref_new_doc_target:n \sparagraphid
6319 }

```

```

6320 \exp_args:NNx
6321 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6322   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6323     \tl_if_empty:nF{ ##1 }{
6324       \stex_get_symbol:n { ##1 }
6325       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
6326     }
6327   }
6328 }
6329 \stex_smsmode_do:
6330 \ignorespacesandpars
6331 }{
6332   \str_if_empty:NF \sparagraphname {
6333     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6334     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6335   }
6336   \stex_if_smsmode:F {
6337     \clist_set:No \l_tmpa_clist \sparagraphtype
6338     \tl_clear:N \l_tmpa_tl
6339     \clist_map_inline:Nn \l_tmpa_clist {
6340       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
6341         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
6342       }
6343     }
6344     \tl_if_empty:NTF \l_tmpa_tl {
6345       \__stex_statements_sparagraph_end:
6346     }{
6347       \l_tmpa_tl
6348     }
6349     \end{stex_annotate_env}
6350   }
6351 }

```

\stexpatchparagraph

```

6352
6353 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
6354   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
6355     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
6356       \titleemph{\l_stex_sparagraph_title_tl}:~
6357     }
6358   }{
6359     \titleemph{\l_stex_sparagraph_start_tl}~
6360   }
6361 }
6362 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
6363
6364 \newcommand\stexpatchparagraph[3] [] {
6365   \str_set:Nx \l_tmpa_str{ #1 }
6366   \str_if_empty:NTF \l_tmpa_str {
6367     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
6368     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
6369   }{
6370     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
6371     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

6372     }
6373 }
6374
6375 \keys_define:nn { stex / inlinepara } {
6376   id      .str_set:N = \sparagraphid ,
6377   type    .str_set:N = \sparagraphtype ,
6378   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
6379   from    .tl_set:N   = \sparagraphfrom ,
6380   to      .tl_set:N   = \sparagraphto ,
6381   name    .str_set:N   = \sparagraphname
6382 }
6383 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
6384   \tl_clear:N \sparagraphfrom
6385   \tl_clear:N \sparagraphto
6386   \str_clear:N \sparagraphid
6387   \str_clear:N \sparagraphtype
6388   \clist_clear:N \l__stex_statements_sparagraph_for_clist
6389   \str_clear:N \sparagraphname
6390   \keys_set:nn { stex / inlinepara }{ #1 }
6391 }
6392 \NewDocumentCommand \inlinepara { O{} m } {
6393   \begingroup
6394     \__stex_statements_inlinepara_args:n{ #1 }
6395     \clist_set:Nn \l_tmpa_clist \sparagraphtype
6396     \str_if_empty:NTF \sparagraphid {
6397       \str_if_empty:NTF \sparagraphname {
6398         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
6399           \stex_ref_new_doc_target:n {}
6400         }
6401       } {
6402         \stex_ref_new_doc_target:n {}
6403       }
6404     } {
6405       \stex_ref_new_doc_target:n \sparagraphid
6406     }
6407     \stex_if_smsmode:TF{
6408       \str_if_empty:NF \sparagraphname {
6409         \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
6410         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6411       }
6412     }{
6413       \seq_clear:N \l_tmpb_seq
6414       \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
6415         \tl_if_empty:nF{ ##1 }{
6416           \stex_get_symbol:n { ##1 }
6417           \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
6418             \l_stex_get_symbol_uri_str
6419           }
6420         }
6421       }
6422       \exp_args:Nnx
6423       \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
6424         \str_if_empty:NF \sparagraphtype {
6425           \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

6426     }
6427     \str_if_empty:NF \sparagraphfrom {
6428         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
6429     }
6430     \str_if_empty:NF \sparagraphto {
6431         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
6432     }
6433     \str_if_empty:NF \sparagraphname {
6434         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
6435         \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
6436         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
6437     }
6438     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
6439         \clist_map_inline:Nn \l_tmpb_seq {
6440             \stex_ref_new_sym_target:n {##1}
6441         }
6442     }
6443     #2
6444 }
6445 }
6446 \endgroup
6447 \stex_smsmode_do:
6448 }
6449

```

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

```

6450 </package>

```

Chapter 33

The Implementation

```
6451 <*package>
6452 <@@=stex_sproof>
6453
6454 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
6455
```

33.1 Proofs

We first define some keys for the proof environment.

```
6456 \keys_define:nn { stex / spf } {
6457   id          .str_set_x:N = \spfid,
6458   for          .clist_set:N = \l__stex_sproof_spf_for_clist ,
6459   from         .tl_set:N    = \l__stex_sproof_spf_from_tl ,
6460   proofend     .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
6461   type         .str_set_x:N = \spftype,
6462   title        .tl_set:N    = \spftitle,
6463   continues    .tl_set:N    = \l__stex_sproof_spf_continues_tl,
6464   functions    .tl_set:N    = \l__stex_sproof_spf_functions_tl,
6465   term         .tl_set:N    = \l__stex_sproof_spf_term_tl,
6466   method      .tl_set:N    = \l__stex_sproof_spf_method_tl,
6467   hide         .bool_set:N  = \l__stex_sproof_spf_hide_bool
6468 }
6469 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
6470   \str_clear:N \spfid
6471   \tl_clear:N \l__stex_sproof_spf_for_tl
6472   \tl_clear:N \l__stex_sproof_spf_from_tl
6473   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
6474   \str_clear:N \spftype
6475   \tl_clear:N \spftitle
6476   \tl_clear:N \l__stex_sproof_spf_continues_tl
6477   \tl_clear:N \l__stex_sproof_spf_term_tl
6478   \tl_clear:N \l__stex_sproof_spf_functions_tl
6479   \tl_clear:N \l__stex_sproof_spf_method_tl
6480   \bool_set_false:N \l__stex_sproof_spf_hide_bool
6481   \keys_set:nn { stex / spf }{ #1 }
6482 }
6483 \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`

```
6484 \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.

```
6485 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
6486 \cs_new_protected:Npn\sproofnumber {
6487   \int_set:Nn \l_tmpa_int {1}
6488   \bool_while_do:nn {
6489     \int_compare_p:nNn {
6490       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6491     } > 0
6492   }{
6493     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
6494     \int_incr:N \l_tmpa_int
6495   }
6496 }
6497 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
6498   \int_set:Nn \l_tmpa_int {1}
6499   \bool_while_do:nn {
6500     \int_compare_p:nNn {
6501       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6502     } > 0
6503   }{
6504     \int_incr:N \l_tmpa_int
6505   }
6506   \int_compare:nNnF \l_tmpa_int = 1 {
6507     \int_decr:N \l_tmpa_int
6508   }
6509   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
6510     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
6511   }
6512 }
6513
6514 \cs_new_protected:Npn \__stex_sproof_add_counter: {
6515   \int_set:Nn \l_tmpa_int {1}
6516   \bool_while_do:nn {
6517     \int_compare_p:nNn {
6518       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6519     } > 0
6520   }{
6521     \int_incr:N \l_tmpa_int
6522   }
6523   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6524 }
6525
```

```

6526 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6527   \int_set:Nn \l_tmpa_int {1}
6528   \bool_while_do:nn {
6529     \int_compare_p:nNn {
6530       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6531     } > 0
6532   }{
6533     \int_incr:N \l_tmpa_int
6534   }
6535   \int_decr:N \l_tmpa_int
6536   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6537 }

```

\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

```

6538 \def\sproof@box{
6539   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6540 }
6541 \def\sproofend{
6542   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6543     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6544   }
6545 }

```

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

spf@*kw

```

6546 \def\spf@proofsketch@kw{Proof~Sketch}
6547 \def\spf@proof@kw{Proof}
6548 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6549 \AddToHook{begindocument}{
6550   \ltx@ifpackageloaded{babel}{
6551     \makeatletter
6552     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6553     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6554       \input{sproof-ngerman.ldf}
6555     }
6556     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6557       \input{sproof-finnish.ldf}
6558     }
6559     \clist_if_in:NnT \l_tmpa_clist {french}{
6560       \input{sproof-french.ldf}
6561     }
6562     \clist_if_in:NnT \l_tmpa_clist {russian}{
6563       \input{sproof-russian.ldf}
6564     }
6565     \makeatother
6566   }{}
6567 }

```


spfsketch

```

6568 \newcommand\spfsketch[2] [] {
6569   \begin{group}
6570   \let \premise \stex_proof_premise:
6571   \stex_sproof_spf_args:n{#1}
6572   \stex_if_smsmode:TF {
6573     \str_if_empty:NF \spfid {
6574       \stex_ref_new_doc_target:n \spfid
6575     }
6576   }{
6577     \seq_clear:N \l_tmpa_seq
6578     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6579       \tl_if_empty:nF{ ##1 }{
6580         \stex_get_symbol:n { ##1 }
6581         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6582           \l_stex_get_symbol_uri_str
6583         }
6584       }
6585     }
6586     \exp_args:Nnx
6587     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6588       \str_if_empty:NF \spftype {
6589         \stex_annotate_invisible:nnn{type}{\spftype}{ }
6590       }
6591       \clist_set:Nn \l_tmpa_clist \spftype
6592       \tl_set:Nn \l_tmpa_tl {
6593         \titleemph{
6594           \tl_if_empty:NTF \spftitle {
6595             \spf@proofsketch@kw
6596           }{
6597             \spftitle
6598           }
6599         }:-
6600       }
6601       \clist_map_inline:Nn \l_tmpa_clist {
6602         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6603           \tl_clear:N \l_tmpa_tl
6604         }
6605       }
6606       \str_if_empty:NF \spfid {
6607         \stex_ref_new_doc_target:n \spfid
6608       }
6609       \l_tmpa_tl #2 \sproofend
6610     }
6611   }
6612   \endgroup
6613   \stex_smsmode_do:
6614 }
6615

```

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

```

\__stex_sproof_maybe_comment:
\__stex_sproof_maybe_comment_end:
\__stex_sproof_start_comment:
6616 \bool_set_false:N \l__stex_sproof_in_spfblock_bool

```

```

6617
6618 \cs_new_protected:Nn \__stex_sproof_maybe_comment: {
6619   \bool_if:NF \l__stex_sproof_in_spfblock_bool {
6620     \par \setbox \l_tmpa_box \vbox \bgroup \everypar{\__stex_sproof_start_comment:}
6621   }
6622 }
6623 \cs_new_protected:Nn \__stex_sproof_maybe_comment_end: {
6624   \bool_if:NF \l__stex_sproof_in_spfblock_bool { \egroup }
6625 }
6626 \cs_new_protected:Nn \__stex_sproof_start_comment: {
6627   \csname @ @ par\endcsname\egroup\item[]\bgroup\stexcommentfont
6628 }
6629
(End definition for \__stex_sproof_maybe_comment:, \__stex_sproof_maybe_comment_end:, and \__-
stex_sproof_start_comment:.)

```

\stexcommentfont

```

6630 \cs_new_protected:Npn \stexcommentfont {
6631   \small\itshape
6632 }

```

(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.

```

6633 \cs_new_protected:Nn \__stex_sproof_start_env:nnn {
6634   \seq_clear:N \l_tmpa_seq
6635   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6636     \tl_if_empty:NF{ ##1 }{
6637       \stex_get_symbol:n { ##1 }
6638       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6639         \l_stex_get_symbol_uri_str
6640       }
6641     }
6642   }
6643   \exp_args:Nnnx
6644   \begin{stex_annotate_env}{#1}{\seq_use:Nn \l_tmpa_seq {,}}
6645   \str_if_empty:NF \spftype {
6646     \stex_annotate_invisible:nnn{type}{\spftype}{}
6647   }
6648   #3 {\~\stex_annotate:nnn{spftitle}{}{#2}}
6649   \str_if_empty:NF \spfid {
6650     \stex_ref_new_doc_target:n \spfid
6651   }
6652   \begin{stex_annotate_env}{spfbody}{\bool_if:NTF \l__stex_sproof_spf_hide_bool {false}{true}
6653   \bool_if:NT \l__stex_sproof_spf_hide_bool{
6654     \stex_html_backend:F{\setbox\l_tmpa_box\vbox\bgroup}
6655   }
6656   \begin{list}{}{
6657     \setlength\topsep{0pt}
6658     \setlength\parsep{0pt}
6659     \setlength\rightmargin{0pt}

```

```

6660 } \_stex_sproof_maybe_comment:
6661 }
6662 \cs_new_protected:Nn \_stex_sproof_end_env:n {
6663   \stex_if_smsmode:F{
6664     \_stex_sproof_maybe_comment_end:
6665     \end{list}
6666     \bool_if:NT \l__stex_sproof_spf_hide_bool{
6667       \stex_html_backend:F{\egroup}
6668     }
6669     \clist_set:No \l_tmpa_clist \spftype
6670     #1
6671     \end{stex_annotate_env}
6672     \end{stex_annotate_env}
6673   }
6674 }
6675 }
6676 \NewDocumentEnvironment{sproof}{s O{} m}{
6677   \intarray_gzero:N \l__stex_sproof_counter_intarray
6678   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6679   \stex_reactivate_macro:N \yield
6680   \stex_reactivate_macro:N \eqstep
6681   \stex_reactivate_macro:N \assumption
6682   \stex_reactivate_macro:N \conclude
6683   \stex_reactivate_macro:N \spfstep
6684   \_stex_sproof_spf_args:n{#2}
6685   \stex_if_smsmode:TF {
6686     \str_if_empty:NF \spfid {
6687       \stex_ref_new_doc_target:n \spfid
6688     }
6689   }{
6690     \_stex_sproof_start_env:nnn{sproof}{#3}{
6691       \clist_set:No \l_tmpa_clist \spftype
6692       \tl_clear:N \l_tmpa_tl
6693       \clist_map_inline:Nn \l_tmpa_clist {
6694         \tl_if_exist:cT {\_stex_sproof_sproof_##1_start:}{
6695           \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_start:}}
6696         }
6697         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6698           \tl_set:Nn \l_tmpa_tl {\use:n{}}
6699         }
6700       }
6701       \tl_if_empty:NTF \l_tmpa_tl {
6702         \_stex_sproof_sproof_start:
6703       }{
6704         \l_tmpa_tl
6705       }
6706     }
6707   }
6708   \stex_smsmode_do:
6709 }{\_stex_sproof_end_env:n{
6710   \tl_clear:N \l_tmpa_tl
6711   \clist_map_inline:Nn \l_tmpa_clist {
6712     \tl_if_exist:cT {\_stex_sproof_sproof_##1_end:}{
6713       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_sproof_##1_end:}}

```

```

6714     }
6715   }
6716   \tl_if_empty:NTF \l_tmpa_tl {
6717     \__stex_sproof_sproof_end:
6718   }{
6719     \l_tmpa_tl
6720   }
6721 }
6722 \NewDocumentEnvironment{subproof}{s O{} m}{
6723   \__stex_sproof_spf_args:n{#2}
6724   \stex_if_smsmode:TF {
6725     \str_if_empty:NF \spfid {
6726       \stex_ref_new_doc_target:n \spfid
6727     }
6728   }{
6729     \__stex_sproof_start_env:nnn{subproof}{\item[\sproofnumber]\ignorespacesandpars #3}{}
6730   }
6731   \__stex_sproof_add_counter:
6732   \stex_smsmode_do:
6733 }{\__stex_sproof_remove_counter:\__stex_sproof_end_env:n{
6734   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6735     \__stex_sproof_inc_counter:
6736   }
6737   \aftergroup\__stex_sproof_maybe_comment:
6738 }
6739 \AddToHook{env/subproof/before}{\__stex_sproof_maybe_comment_end:}
6740
6741 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6742   \par\noindent\titleemph{
6743     \tl_if_empty:NTF \spftype {
6744       \spf@proof@kw
6745     }{
6746       \spftype
6747     }
6748   }:
6749 }
6750 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6751
6752 \newcommand\stexpatchproof[3] [] {
6753   \str_set:Nx \l_tmpa_str{ #1 }
6754   \str_if_empty:NTF \l_tmpa_str {
6755     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6756     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6757   }{
6758     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6759     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6760   }
6761 }

```

\pstep
 \conclude
 \assumption
 \have
 \eqstep

```

6762
6763 \keys_define:nn { stex / spfsteps } {
6764   id          .str_set_x:N = \spfstepid,
6765   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,

```

```

6766 type .str_set_x:N = \spftype,
6767 title .tl_set:N = \spftitle,
6768 method .tl_set:N = \l__stex_sproof_spf_method_tl,
6769 term .tl_set:N = \l__stex_sproof_spf_term_tl
6770 }
6771 \cs_new_protected:Nn \__stex_sproof_spfstep_args:n {
6772 \str_clear:N \spfstepid
6773 \clist_clear:N \l__stex_sproof_spf_for_clist
6774 \str_clear:N \spftype
6775 \tl_clear:N \l__stex_sproof_spf_method_tl
6776 \tl_clear:N \l__stex_sproof_spf_term_tl
6777 %\bool_set_false:N \l__stex_sproof_inc_counter_bool
6778 \keys_set:nn { stex / spfsteps }{ #1 }
6779 }
6780
6781 \cs_new_protected:Nn \__stex_sproof_make_step_macro:Nnnnn {
6782 \NewDocumentCommand #1 {s O{} +m} {
6783 \__stex_sproof_maybe_comment_end:
6784
6785 \__stex_sproof_spfstep_args:n{##2}
6786 \stex_annotate:nnn{spfstep}{#2}{
6787 \tl_if_empty:NF \l__stex_sproof_spf_term_tl {
6788 \stex_annotate_invisible:nnn{spfyield}{}{\l__stex_sproof_spf_term_tl$}
6789 }
6790 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6791 #4
6792 }{
6793 \item[\IfBooleanTF ##1 {}{#3}]
6794 }
6795 \ignorespacesandpars ##3
6796 }
6797 \bool_if:NF \l__stex_sproof_in_spfblock_bool { \IfBooleanTF ##1 {}{ #5 } }
6798 \__stex_sproof_maybe_comment:
6799 }
6800 \stex_deactivate_macro:Nn #1 {sproof~environments}
6801 }
6802
6803 \__stex_sproof_make_step_macro:Nnnnn \assumption {assumption} \sproofnumber {} \__stex_sproof
6804 \__stex_sproof_make_step_macro:Nnnnn \conclude {conclusion} {\$ \Rightarrow$} {} {}
6805 \__stex_sproof_make_step_macro:Nnnnn \spfstep {} \sproofnumber {} \__stex_sproof_inc_counter
6806
6807 \NewDocumentCommand \eqstep {s m}{
6808 \__stex_sproof_maybe_comment_end:
6809 \bool_if:NTF \l__stex_sproof_in_spfblock_bool {
6810 $=$
6811 }{
6812 \item[$=$]
6813 }
6814 $\stex_annotate:nnn{spfstep}{eq}{ #2 }$
6815 \__stex_sproof_maybe_comment:
6816 }
6817 \stex_deactivate_macro:Nn \eqstep {sproof~environments}
6818
6819 \NewDocumentCommand \yield {+m}{

```

```

6820 \stex_annotate:nnn{spfyield}{\}{ #1 }
6821 }
6822 \stex_deactivate_macro:Nn \yield {sproof~environments}
6823
6824 \NewDocumentEnvironment{spfblock}{\}{\{
6825 \item[]
6826 \bool_set_true:N \l__stex_sproof_in_spfblock_bool
6827 }{
6828 \aftergroup\__stex_sproof_maybe_comment:
6829 }
6830 \AddToHook{env/spfblock/before}{\__stex_sproof_maybe_comment_end:}
6831

```

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

\spfidea

```

6832 \NewDocumentCommand\spfidea{0{\} +m}{
6833 \__stex_sproof_spf_args:n{#1}
6834 \titleemph{
6835 \tl_if_empty:NTF \spftype {Proof~Idea}{
6836 \spftype
6837 }:
6838 }~#2
6839 \sproofend
6840 }

```

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

```

6841 \newcommand\spfjust[1]{
6842 #1
6843 }
6844 \</package>

```

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

Chapter 34

STEX -Others Implementation

```
6845 <*package>
6846
6847 %%%%%%%%%% others.dtx %%%%%%%%%%
6848
6849 <@@=stex_others>
        Warnings and error messages
6850 % None

\MSC Math subject classifier

6851 \NewDocumentCommand \MSC {m} {
6852 % TODO
6853 }

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

6854 \@ifpackageloaded{tikzinput}{
6855 \RequirePackage{stex-tikzinput}
6856 }{}
6857
6858 \bool_if:NT \c_stex_persist_mode_bool {
6859 \let__stex_notation_restore_notation_old:nnnnn
6860 \__stex_notation_restore_notation:nnnnn
6861 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6862 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6863 \ExplSyntaxOn
6864 }
6865 \def__stex_notation_restore_notation:nnnnn{
6866 \ExplSyntaxOff
6867 \catcode'\sim10
6868 \__stex_notation_restore_notation_new:nnnnn
6869 }
6870 \input{\jobname.sms}
6871 \let__stex_notation_restore_notation:nnnnn
6872 \__stex_notation_restore_notation_old:nnnnn
6873 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```

```

6874 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6875 \l_tmpa_str
6876 \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6877 \c_stex_mathhub_main_manifest_prop
6878 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6879 }
6880 }
6881
6882 \stex_get_document_uri:
6883 </package>

```


Chapter 35

STEX -Metatheory Implementation

```
6884 <*package>
6885 <@@=stex_modules>
6886
6887 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6888
6889 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6890 \begingroup
6891 \stex_module_setup:nn{
6892   ns=\c_stex_metatheory_ns_str,
6893   meta=NONE
6894 }{Metatheory}
6895 \stex_reactivate_macro:N \symdecl
6896 \stex_reactivate_macro:N \notation
6897 \stex_reactivate_macro:N \symdef
6898 \ExplSyntaxOff
6899 \csname stex_suppress_html:n\endcsname{
6900   % is-a (a:A, a \in A, a is an A, etc.)
6901   \symdecl{isa}[args=ai]
6902   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6903   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6904   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6905
6906   % bind (\forall, \Pi, \lambda etc.)
6907   \symdecl{bind}[args=Bi,assoc=pre]
6908   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6909   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6910   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6911
6912   % implicit bind
6913   \symdecl{implicitbind}[args=Bi,assoc=pre]
6914   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6915     \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6916     \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6917
6918   % dummy variable
```

```

6919 \symdecl{dummyvar}
6920 \notation{dummyvar}[underscore]{\comp\_}
6921 \notation{dummyvar}[dot]{\comp\cdot}
6922 \notation{dummyvar}[dash]{\comp{\rm --}}
6923
6924 %fromto (function space, Hom-set, implication etc.)
6925 \symdecl{fromto}[args=ai]
6926 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6927 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6928
6929 % mapto (lambda etc.)
6930 \symdecl{mapto}[args=Bi]
6931 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6932 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
6933 %\notation{mapto}[lambdau]{\comp\lambda_#1 \comp.\; #2}{#1 \comp, #2}
6934
6935 % function/operator application
6936 \symdecl{apply}[args=ia]
6937 \notation{apply}[prec=0;0x\infpres,parens,op=\cdot(\cdot)]{#1 \comp( #2 \comp)}{##1 \comp,
6938 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6939
6940 % collection of propositions/booleans/truth values
6941 \symdecl{prop}[name=proposition]
6942 \notation{prop}[prop]{\comp{\rm prop}}
6943 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6944
6945 \symdecl{judgmentholds}[args=1]
6946 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; #1}
6947
6948 % sequences
6949 \symdecl{seqtype}[args=1]
6950 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6951
6952 \symdecl{seqexpr}[args=a]
6953 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6954
6955 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6956 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6957 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6958 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6959 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6960 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6961 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6962 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6963 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6964
6965 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6966 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
6967
6968 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\ellipses}}{##1\comp,##2}
6969 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\ellipses},#2}{##1\comp,##2}
6970 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\ellipses},#2\comp{\ellipses},#3}{##1\comp,##2,##3}
6971
6972 % nat literals

```

```

6973 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6974
6975 % letin (''let'', local definitions, variable substitution)
6976 \symdecl{letin}[args=bii]
6977 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\#2\;\comp{\rm in}}\;#3}
6978 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
6979 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
6980
6981 % structures
6982 \symdecl*{module-type}[args=1]
6983 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6984 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6985 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6986
6987 % objects
6988 \symdecl{object}
6989 \notation{object}{\comp{\mathtt{OBJECT}}}
6990
6991 }
6992
6993 % The following are abbreviations in the sTeX corpus that are left over from earlier
6994 % developments. They will eventually be phased out.
6995
6996 \ExplSyntaxOn
6997 \stex_add_to_current_module:n{
6998   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6999   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
7000   \def\livar{\csname sequence-index\endcsname[li]}
7001   \def\uivar{\csname sequence-index\endcsname[ui]}
7002   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
7003   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
7004 }
7005 \__stex_modules_end_module:
7006 \endgroup
7007 \</package>

```

Chapter 36

Tikzinput Implementation

```
7008 <@@=tikzinput>
7009 <*package>
7010
7011 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
7012
7013 \ProvidesExplPackage{tikzinput}{2022/08/08}{3.2.0}{tikzinput package}
7014 \RequirePackage{l3keys2e}
7015
7016 \keys_define:nn { tikzinput } {
7017   image .bool_set:N = \c_tikzinput_image_bool,
7018   image .default:n = false ,
7019   unknown .code:n = {}
7020 }
7021
7022 \ProcessKeysOptions { tikzinput }
7023
7024 \bool_if:NTF \c_tikzinput_image_bool {
7025   \RequirePackage{graphicx}
7026
7027   \providecommand\usetikzlibrary[]{}
7028   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
7029 }{
7030   \RequirePackage{tikz}
7031   \RequirePackage{standalone}
7032
7033   \newcommand \tikzinput [2] [] {
7034     \setkeys{Gin}{#1}
7035     \ifx \Gin@ewidth \Gin@exclamation
7036       \ifx \Gin@eheight \Gin@exclamation
7037         \input { #2 }
7038       \else
7039         \resizebox{!}{ \Gin@eheight }{
7040           \input { #2 }
7041         }
7042       \fi
7043     \else
7044       \ifx \Gin@eheight \Gin@exclamation
7045         \resizebox{ \Gin@ewidth }{!}{
```

```

7046         \input { #2 }
7047     }
7048     \else
7049         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
7050             \input { #2 }
7051         }
7052     \fi
7053 \fi
7054 }
7055 }
7056
7057 \newcommand \ctikzinput [2] [] {
7058     \begin{center}
7059         \tikzinput [#1] {#2}
7060     \end{center}
7061 }
7062
7063 \@ifpackageloaded{stex}{
7064     \RequirePackage{stex-tikzinput}
7065 }{}
7066
7067 </package>
7068 <*stex>
7069
7069 \ProvidesExplPackage{stex-tikzinput}{2022/08/08}{3.2.0}{stex-tikzinput}
7070 \RequirePackage{stex}
7071 \RequirePackage{tikzinput}
7072
7073 \newcommand\mhtikzinput[2] []{%
7074     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
7075     \stex_in_repository:nn\Gin@mhrepos{
7076         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
7077     }
7078 }
7079 \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
7080
7081 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
7082     \pgfkeys@spdef\pgf@temp{#1}
7083     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
7084     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
7085     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
7086     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
7087     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
7088     \catcode'\@=11
7089     \catcode'\|=12
7090     \catcode'\$=3
7091     \pgfutil@InputIfFileExists{#2}{-}{-}
7092     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
7093     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
7094     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
7095 }
7096
7097
7098 \newcommand\libusetikzlibrary[1]{

```

```

7099 \prop_if_exist:NF \l_stex_current_repository_prop {
7100   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7101 }
7102 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
7103   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
7104 }
7105 \seq_clear:N \l__tikzinput_libinput_files_seq
7106 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
7107 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
7108
7109 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
7110   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
7111   \IfFileExists{ \l_tmpa_str }{
7112     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7113   }{}
7114   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
7115   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
7116 }
7117
7118 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
7119 \IfFileExists{ \l_tmpa_str }{
7120   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
7121 }{}
7122
7123 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
7124   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
7125 }{
7126   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
7127     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
7128       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
7129     }
7130   }{
7131     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
7132   }
7133 }
7134 }
7135 </stex>

```

Chapter 37

document-structure.sty Implementation

```
7136 <*package>
7137 <@@=document_structure>
7138 \ProvidesExplPackage{document-structure}{2022/08/08}{3.2.0}{Modular Document Structure}
7139 \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).

```
7140
7141 \keys_define:nn{ document-structure }{
7142   class      .str_set_x:N = \c_document_structure_class_str,
7143   topsect    .str_set_x:N = \c_document_structure_topsect_str,
7144   unknown    .code:n      = {
7145     \PassOptionsToClass{\CurrentOption}{stex}
7146     \PassOptionsToClass{\CurrentOption}{tikzinput}
7147   }
7148   % showignores .bool_set:N = \c_document_structure_showignores_bool,
7149 }
7150 \ProcessKeysOptions{ document-structure }
7151 \str_if_empty:NT \c_document_structure_class_str {
7152   \str_set:Nn \c_document_structure_class_str {article}
7153 }
7154 \str_if_empty:NT \c_document_structure_topsect_str {
7155   \str_set:Nn \c_document_structure_topsect_str {section}
7156 }
```

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

```
7157 \RequirePackage{xspace}
7158 \RequirePackage{comment}
7159 \RequirePackage{stex}
7160 \AddToHook{begindocument}{
```

```

7161 \ltx@ifpackageloaded{babel}{
7162   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7163   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
7164     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
7165   }
7166 }{}
7167 }

```

`\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.

```

7168 \int_new:N \l_document_structure_section_level_int
7169 \str_case:NnF \c_document_structure_topsect_str {
7170   {part}}{
7171     \int_set:Nn \l_document_structure_section_level_int {0}
7172   }
7173   {chapter}{
7174     \int_set:Nn \l_document_structure_section_level_int {1}
7175   }
7176 }{
7177   \str_case:NnF \c_document_structure_class_str {
7178     {book}{
7179       \int_set:Nn \l_document_structure_section_level_int {0}
7180     }
7181     {report}{
7182       \int_set:Nn \l_document_structure_section_level_int {0}
7183     }
7184   }{
7185     \int_set:Nn \l_document_structure_section_level_int {2}
7186   }
7187 }

```

37.2 Document Structure

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

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

EdN:9

```

7188 \def\current@section@level{document}%
7189 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
7190 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

7191 \cs_new_protected:Npn \skipfragment {

```

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


```

7192 \ifcase\l_document_structure_section_level_int
7193 \or\stepcounter{part}
7194 \or\stepcounter{chapter}
7195 \or\stepcounter{section}
7196 \or\stepcounter{subsection}
7197 \or\stepcounter{subsubsection}
7198 \or\stepcounter{paragraph}
7199 \or\stepcounter{subparagraph}
7200 \fi
7201 }

```

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

`blindfragment (env.)`

```

7202 \newcommand\at@begin@blindsfragment[1]{
7203 \newenvironment{blindfragment}
7204 {
7205 \int_incr:N\l_document_structure_section_level_int
7206 \at@begin@blindsfragment\l_document_structure_section_level_int
7207 }{}

```

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

```

7208 \newcommand\sfragment@nonum[2]{
7209 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
7210 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
7211 }

```

(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.

```

7212 \newcommand\sfragment@num[2]{
7213 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
7214 \@nameuse{#1}{#2}
7215 }{
7216 \cs_if_exist:NTF\rdfmata@sectioning{
7217 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
7218 }{
7219 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
7220 }
7221 }
7222 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
7223 }

```

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

`sfragment (env.)`

```

7224 \keys_define:nn { document-structure / sfragment }{
7225 id .str_set_x:N = \l__document_structure_sfragment_id_str,
7226 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

7227 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
7228 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
7229 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
7230 type          .tl_set:N     = \l__document_structure_sfragment_type_tl,
7231 short         .tl_set:N     = \l__document_structure_sfragment_short_tl,
7232 intro        .tl_set:N     = \l__document_structure_sfragment_intro_tl,
7233 imports       .tl_set:N     = \l__document_structure_sfragment_imports_tl,
7234 loadmodules   .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
7235 }
7236 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
7237   \str_clear:N \l__document_structure_sfragment_id_str
7238   \str_clear:N \l__document_structure_sfragment_date_str
7239   \clist_clear:N \l__document_structure_sfragment_creators_clist
7240   \clist_clear:N \l__document_structure_sfragment_contributors_clist
7241   \tl_clear:N \l__document_structure_sfragment_srccite_tl
7242   \tl_clear:N \l__document_structure_sfragment_type_tl
7243   \tl_clear:N \l__document_structure_sfragment_short_tl
7244   \tl_clear:N \l__document_structure_sfragment_imports_tl
7245   \tl_clear:N \l__document_structure_sfragment_intro_tl
7246   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
7247   \keys_set:nn { document-structure / sfragment } { #1 }
7248 }

```

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.

```

7249 \newif\if@mainmatter\@mainmattertrue
7250 \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.

```

7251 \keys_define:nn { document-structure / sectioning }{
7252   name      .str_set_x:N = \l__document_structure_sect_name_str ,
7253   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
7254   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
7255   clear     .default:n   = {true} ,
7256   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
7257   num       .default:n   = {true}
7258 }
7259 \cs_new_protected:Nn \l__document_structure_sect_args:n {
7260   \str_clear:N \l__document_structure_sect_name_str
7261   \str_clear:N \l__document_structure_sect_ref_str
7262   \bool_set_false:N \l__document_structure_sect_clear_bool
7263   \bool_set_false:N \l__document_structure_sect_num_bool
7264   \keys_set:nn { document-structure / sectioning } { #1 }
7265 }
7266 \newcommand\omdoc@sectioning[3][]{
7267   \l__document_structure_sect_args:n {#1 }
7268   \let\omdoc@sect@name\l__document_structure_sect_name_str
7269   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
7270   \if@mainmatter% numbering not overridden by frontmatter, etc.
7271     \bool_if:NTF \l__document_structure_sect_num_bool {
7272       \sfragment@num{#2}{#3}
7273     }{

```

```

7274     \sfragment@nonum{#2}{#3}
7275   }
7276   \def\current@section@level{\omdoc@sect@name}
7277   \else
7278     \sfragment@nonum{#2}{#3}
7279   \fi
7280 }% 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.

```

7281 \newcommand\sfragment@redefine@addtocontents[1]{%
7282 %\edef\__document_structureimport{#1}%
7283 %\@for\@I:=\__document_structureimport\do{%
7284 %\edef\@path{\csname module@\@I @path\endcsname}%
7285 %\@ifundefined{tf@toc}\relax%
7286 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
7287 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
7288 %\def\addcontentsline##1##2##3{%
7289 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
7290 %\else% hyperref.sty not loaded
7291 %\def\addcontentsline##1##2##3{%
7292 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
7293 %\fi
7294 }% 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.

```

7295 \newenvironment{sfragment}[2][ ]% keys, title
7296 {
7297   \__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.

```

7298   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
7299
7300   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
7301     \sfragment@redefine@addtocontents{
7302       %\@ifundefined{module@id}\used@modules%
7303       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
7304     }
7305   }

```

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

```

7306
7307   \stex_document_title:n { #2 }
7308
7309   \int_incr:N\l__document_structure_section_level_int
7310   \ifcase\l__document_structure_section_level_int
7311     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
7312     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
7313     \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
7314     \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}

```

```

7315 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
7316 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
7317 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
7318 \fi
7319 \at@begin@sfragment[#1]\l__document_structure_section_level_int{#2}
7320 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7321   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7322 }
7323 }% for customization
7324 {}

```

and finally, we localize the sections

```

7325 \newcommand\omdoc@part@kw{Part}
7326 \newcommand\omdoc@chapter@kw{Chapter}
7327 \newcommand\omdoc@section@kw{Section}
7328 \newcommand\omdoc@subsection@kw{Subsection}
7329 \newcommand\omdoc@subsubsection@kw{Subsubsection}
7330 \newcommand\omdoc@paragraph@kw{paragraph}
7331 \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`

```

7332 \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).

```

7333 \cs_if_exist:NTF\frontmatter{
7334   \let\__document_structure_orig_frontmatter\frontmatter
7335   \let\frontmatter\relax
7336 }{
7337   \tl_set:Nn\__document_structure_orig_frontmatter{
7338     \clearpage
7339     \@mainmatterfalse
7340     \pagenumbering{roman}
7341   }
7342 }
7343 \cs_if_exist:NTF\backmatter{
7344   \let\__document_structure_orig_backmatter\backmatter
7345   \let\backmatter\relax
7346 }{
7347   \tl_set:Nn\__document_structure_orig_backmatter{
7348     \clearpage
7349     \@mainmatterfalse
7350     \pagenumbering{roman}
7351   }

```

7352 }

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.

```
7353 \newenvironment{frontmatter}{
7354   \__document_structure_orig_frontmatter
7355 }{
7356   \cs_if_exist:NTF\mainmatter{
7357     \mainmatter
7358   }{
7359     \clearpage
7360     \@mainmattertrue
7361     \pagenumbering{arabic}
7362   }
7363 }
```

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

```
7364 \newenvironment{backmatter}{
7365   \__document_structure_orig_backmatter
7366 }{
7367   \cs_if_exist:NTF\mainmatter{
7368     \mainmatter
7369   }{
7370     \clearpage
7371     \@mainmattertrue
7372     \pagenumbering{arabic}
7373   }
7374 }
```

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

```
7375 \@mainmattertrue\pagenumbering{arabic}
```

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

```
7376 \def \c__document_structure_document_str{document}
7377 \newcommand\afterprematurestop{}
7378 \def\prematurestop@endsfragment{
7379   \unless\ifx\@currenvir\c__document_structure_document_str
7380     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7381       \expandafter\prematurestop@endsfragment
7382     }
7383   }
7384 \providecommand\prematurestop{
7385   \message{Stopping~sTeX~processing~prematurely}
7386   \prematurestop@endsfragment
7387   \afterprematurestop
7388   \end{document}
7389 }
```

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

37.4 Global Variables

\setSGvar set a global variable

```
7390 \RequirePackage{etoolbox}
7391 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
7392 \newrobustcmd\useSGvar[1]{%
7393   \@ifundefined{sTeX@Gvar@#1}
7394   {\PackageError{document-structure}
7395     {The sTeX Global variable #1 is undefined}
7396     {set it with \protect\setSGvar}}
7397   \@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.

```
7398 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7399   \@ifundefined{sTeX@Gvar@#1}
7400   {\PackageError{document-structure}
7401     {The sTeX Global variable #1 is undefined}
7402     {set it with \protect\setSGvar}}
7403   {\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.

```
7404 \*cls)
7405 \@@=notesslides)
7406 \ProvidesExplClass{notesslides}{2022/08/08}{3.2.0}{notesslides Class}
7407 \RequirePackage{13keys2e}
7408
7409 \keys_define:nn{notesslides / cls}{
7410   class .str_set_x:N = \c__notesslides_class_str,
7411   notes .bool_set:N = \c__notesslides_notes_bool ,
7412   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7413   docopt .str_set_x:N = \c__notesslides_docopt_str,
7414   unknown .code:n = {
7415     \PassOptionsToPackage{\CurrentOption}{document-structure}
7416     \PassOptionsToClass{\CurrentOption}{beamer}
7417     \PassOptionsToPackage{\CurrentOption}{notesslides}
7418     \PassOptionsToPackage{\CurrentOption}{stex}
7419   }
7420 }
7421 \ProcessKeysOptions{ notesslides / cls }
7422
7423 \str_if_empty:NF \c__notesslides_class_str {
7424   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7425 }
7426
7427 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7428   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7429 }
7430 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7431   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7432 }
7433
7434 \RequirePackage{stex}
```

```

7435 \stex_html_backend:T {
7436   \bool_set_true:N\c__notesslides_notes_bool
7437 }
7438
7439 \bool_if:NTF \c__notesslides_notes_bool {
7440   \PassOptionsToPackage{notes=true}{notesslides}
7441   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7442 }{
7443   \PassOptionsToPackage{notes=false}{notesslides}
7444   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7445 }
7446 \</cls>

```

now we do the same for the notesslides package.

```

7447 <*package>
7448 \ProvidesExplPackage{notesslides}{2022/08/08}{3.2.0}{notesslides Package}
7449 \RequirePackage{l3keys2e}
7450
7451 \keys_define:nn{notesslides / pkg}{
7452   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7453   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7454   notes            .bool_set:N = \c__notesslides_notes_bool ,
7455   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7456   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7457   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7458   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7459   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
7460   unknown          .code:n      = {
7461     \PassOptionsToClass{\CurrentOption}{stex}
7462     \PassOptionsToClass{\CurrentOption}{tikzinput}
7463   }
7464 }
7465 \ProcessKeysOptions{ notesslides / pkg }
7466
7467 \RequirePackage{stex}
7468 \stex_html_backend:T {
7469   \bool_set_true:N\c__notesslides_notes_bool
7470 }
7471
7472 \newif\ifnotes
7473 \bool_if:NTF \c__notesslides_notes_bool {
7474   \notesttrue
7475 }{
7476   \notestfalse
7477 }
7478

```

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

```

7479 \str_if_empty:NTF \c__notesslides_topsect_str {
7480   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7481 }{
7482   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7483 }
7484 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```



```
7485 </package>
```

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

```
7486 <*cls>
7487 \bool_if:NTF \c__notesslides_notes_bool {
7488   \str_if_empty:NT \c__notesslides_class_str {
7489     \str_set:Nn \c__notesslides_class_str {article}
7490   }
7491   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
7492     {\c__notesslides_class_str}
7493 }{
7494   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7495   \newcounter{Item}
7496   \newcounter{paragraph}
7497   \newcounter{subparagraph}
7498   \newcounter{Hfootnote}
7499 }
7500 \RequirePackage{document-structure}
```

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

```
7501 \RequirePackage{notesslides}
7502 </cls>
```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the \TeX packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the \TeX -specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```
7503 <*package>
7504 \bool_if:NT \c__notesslides_notes_bool {
7505   \RequirePackage{a4wide}
7506   \RequirePackage{marginnote}
7507   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7508   \RequirePackage{mdframed}
7509   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7510   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7511 }
7512 \RequirePackage{stex-tikzinput}
7513 \RequirePackage{comment}
7514 \RequirePackage{url}
7515 \RequirePackage{graphicx}
7516 \RequirePackage{pgf}
7517 \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.

```
7518 \bool_if:NT \c__notesslides_notes_bool {
7519   \renewcommand\usetheme[2][\usepackage{#1}{beamertheme#2}]
7520 }
```

```

7521 \NewDocumentCommand \libusetheme {0{} m} {
7522   \libusepackage[#1]{beamertheme#2}
7523 }
7524

```

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

```

7525 \newcounter{slide}
7526 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7527 \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.

```

7528 \bool_if:NTF \c__notesslides_notes_bool {
7529   \renewenvironment{note}{\ignorespaces}{}
7530 }{
7531   \excludecomment{note}
7532 }

```

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.

```

7533 \bool_if:NT \c__notesslides_notes_bool {
7534   \newlength{\slideframewidth}
7535   \setlength{\slideframewidth}{1.5pt}

```

frame (*env.*) We first define the keys.

```

7536 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7537   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7538     \bool_set_true:N #1
7539   }{
7540     \bool_set_false:N #1
7541   }
7542 }
7543 \keys_define:nn{notesslides / frame}{
7544   label .str_set_x:N = \l__notesslides_frame_label_str,
7545   allowframebreaks .code:n = {
7546     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7547   },
7548   allowdisplaybreaks .code:n = {
7549     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7550   },
7551   fragile .code:n = {
7552     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7553   },
7554   shrink .code:n = {
7555     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7556   },
7557   squeeze .code:n = {
7558     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7559   },
7560   t .code:n = {

```

```

7561     \_notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7562   },
7563   unknown    .code:n      = {}
7564 }
7565 \cs_new_protected:Nn \_notesslides_frame_args:n {
7566   \str_clear:N \l__notesslides_frame_label_str
7567   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7568   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7569   \bool_set_true:N \l__notesslides_frame_fragile_bool
7570   \bool_set_true:N \l__notesslides_frame_shrink_bool
7571   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7572   \bool_set_true:N \l__notesslides_frame_t_bool
7573   \keys_set:nn { notesslides / frame }{ #1 }
7574 }

```

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

```

7575 \renewenvironment{frame}[1][]{
7576   \_notesslides_frame_args:n{#1}
7577   \sffamily
7578   \stepcounter{slide}
7579   \def\@currentlabel{\theslide}
7580   \str_if_empty:NF \l__notesslides_frame_label_str {
7581     \label{\l__notesslides_frame_label_str}
7582   }

```

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

```

7583   \def\itemize@level{outer}
7584   \def\itemize@outer{outer}
7585   \def\itemize@inner{inner}
7586   \renewcommand\newpage{\addtocounter{framenum}{1}}
7587   %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7588   \renewenvironment{itemize}{
7589     \ifx\itemize@level\itemize@outer
7590       \def\itemize@label{$\rhd$}
7591     \fi
7592     \ifx\itemize@level\itemize@inner
7593       \def\itemize@label{$\scriptstyle\rhd$}
7594     \fi
7595     \begin{list}
7596       {\itemize@label}
7597       {\setlength{\labelsep}{.3em}
7598        \setlength{\labelwidth}{.5em}
7599        \setlength{\leftmargin}{1.5em}
7600       }
7601     \edef\itemize@level{\itemize@inner}
7602   }{
7603     \end{list}
7604   }

```

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

```

7605   \stex_html_backend:TF {
7606     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7607     \mdf@patchamsthm
7608   }{
7609     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid

```

```

7610     }
7611   }{
7612     \stex_html_backend:TF {
7613       \miko@slidelabel\egroup\end{stex_annotate_env}
7614     }\medskip\miko@slidelabel\end{mdframed}}
7615   }

```

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

`\frametitle`

```

7616   \renewcommand{\frametitle}[1]{
7617     \stex_document_title:n { #1 }
7618     {\Large\bf\sf\color{blue}{#1}}\medskip
7619   }
7620 }

```

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

EdN:10

`\pause` 10

```

7621 \bool_if:NT \c__notesslides_notes_bool {
7622   \newcommand\pause{}
7623 }

```

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

`nparagraph (env.)`

```

7624 \bool_if:NTF \c__notesslides_notes_bool {
7625   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
7626 }{
7627   \excludecomment{nparagraph}
7628 }

```

`nfragment (env.)`

```

7629 \bool_if:NTF \c__notesslides_notes_bool {
7630   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1][#2]}\end{sfragment}}
7631 }{
7632   \excludecomment{nfragment}
7633 }

```

`ndefinition (env.)`

```

7634 \bool_if:NTF \c__notesslides_notes_bool {
7635   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7636 }{
7637   \excludecomment{ndefinition}
7638 }

```

`nassertion (env.)`

```

7639 \bool_if:NTF \c__notesslides_notes_bool {
7640   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7641 }{
7642   \excludecomment{nassertion}
7643 }

```

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

`nsproof (env.)`

```
7644 \bool_if:NTF \c__notesslides_notes_bool {
7645   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
7646 }{
7647   \excludecomment{nproof}
7648 }
```

`nexample (env.)`

```
7649 \bool_if:NTF \c__notesslides_notes_bool {
7650   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
7651 }{
7652   \excludecomment{nexample}
7653 }
```

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

```
7654 \def\inputref@preskip{\smallskip}
7655 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
7656 \let\orig@inputref\inputref
7657 \def\inputref{\@ifstar\ninputref\orig@inputref}
7658 \newcommand\ninputref[2] []{
7659   \bool_if:NT \c__notesslides_notes_bool {
7660     \orig@inputref[#1]{#2}
7661   }
7662 }
```

(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>}`.

```
7663 \newlength{\slidelogoheight}
7664
7665 \RequirePackage{graphicx}
7666
7667 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7668 \providecommand\mhgraphics[2] []{
7669   \def\Gin@mhrepos{}\setkeys{Gin}{#1}
7670   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7671 }
7672
7673 \bool_if:NTF \c__notesslides_notes_bool {
7674   \setlength{\slidelogoheight}{.4cm}
7675 }{
7676   \setlength{\slidelogoheight}{.25cm}
7677 }
```

```

7678 \ifcsname slidelogo\endcsname\else
7679 \newsavebox{\slidelogo}
7680 \sbox{\slidelogo}{\sTeX}
7681 \fi
7682 \newrobustcmd{\setslidelogo}[2][\{
7683 \tl_if_empty:nTF{#1}{
7684 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#2}}
7685 }{
7686 \sbox{\slidelogo}{\mhgraphics[height=\slidelogoheight,mhrepos=#1]{#2}}
7687 }
7688 }

```

(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.

```

7689 \bool_if:NT \c__notesslides_notes_bool {
7690 \def\author{\@dblarg\@ns@author}
7691 \long\def\@ns@author[#1]#2{%
7692 \def\c__notesslides_shortauthor{#1}%
7693 \def\@author{#2}
7694 }
7695 }

```

(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.

```

7696 \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.

```

7697 \def\copyrightnotice{%
7698 \footnotesize\copyright : \hspace{.3ex}%
7699 \ifcsname source\endcsname\source\else%
7700 \ifcsname c__notesslides_shortauthor\endcsname\c__notesslides_shortauthor\else%
7701 \PackageWarning{notesslides}{Author/Source-undefined-in-copyright-notice}%
7702 ?source/author?\fi%
7703 \fi}
7704 \newsavebox{\cclogo}
7705 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7706 \newif\ifcchref\cchreffalse
7707 \AtBeginDocument{
7708 \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7709 }
7710 \def\licensing{
7711 \ifcchref
7712 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7713 \else
7714 {\usebox{\cclogo}}

```

```

7715 \fi
7716 }
7717 \newrobustcmd{\setlicensing}[2][]{
7718   \def\@url{#1}
7719   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7720   \ifx\@url\@empty
7721     \def\licensing{\usebox{\cclogo}}
7722   \else
7723     \def\licensing{
7724       \ifcchref
7725         \href{#1}{\usebox{\cclogo}}
7726       \else
7727         {\usebox{\cclogo}}
7728     \fi
7729   }
7730 \fi
7731 }

```

(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
7732 \newrobustcmd\miko@slidelabel{
7733   \vbox to \slidelogoheight{
7734     \vss\hbox to \slidewidth
7735       {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7736   }
7737 }

```

(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.

```

7738 \def\Gin@mhrepos{}
7739 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7740 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7741 \newrobustcmd\frameimage[2][]{
7742   \stepcounter{slide}
7743   \bool_if:NT \c__notesslides_frameimages_bool {
7744     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7745     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7746     \begin{center}
7747       \bool_if:NTF \c__notesslides_fiboxed_bool {
7748         \fbox{
7749           \ifx\Gin@ewidth\@empty
7750             \ifx\Gin@mhrepos\@empty
7751               \mhgraphics[width=\slidewidth,#1]{#2}
7752             \else
7753               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7754             \fi
7755           \else% Gin@ewidth empty

```

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

```

7756         \ifx\Gin@mhrepos\@empty
7757         \mhgraphics[#1]{#2}
7758     \else
7759         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7760     \fi
7761 \fi% Gin@ewidth empty
7762 }
7763 }{
7764     \ifx\Gin@ewidth\@empty
7765     \ifx\Gin@mhrepos\@empty
7766         \mhgraphics[width=\slidewidth,#1]{#2}
7767     \else
7768         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7769     \fi
7770     \ifx\Gin@mhrepos\@empty
7771         \mhgraphics[#1]{#2}
7772     \else
7773         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7774     \fi
7775     \fi% Gin@ewidth empty
7776 }
7777 \end{center}
7778 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7779 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7780 }
7781 } % 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`.

```

7782 \stex_html_backend:F {
7783     \bool_if:NT \c__notesslides_sectocframes_bool {
7784         \str_if_eq:VnTF \__notesslidestopsect{part}{
7785             \newcounter{chapter}\counterwithin*{section}{chapter}
7786         }{
7787             \str_if_eq:VnT\__notesslidestopsect{chapter}{
7788                 \newcounter{chapter}\counterwithin*{section}{chapter}
7789             }
7790         }
7791     }
7792 }

```

`\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`

```

7793 \def\part@prefix{}
7794 \@ifpackageloaded{document-structure}{
7795     \str_case:VnF \__notesslidestopsect {

```



```

7796 {part}{
7797   \int_set:Nn \l_document_structure_section_level_int {0}
7798   \def\thesection{\arabic{chapter}.\arabic{section}}
7799   \def\part@prefix{\arabic{chapter}.}
7800 }
7801 {chapter}{
7802   \int_set:Nn \l_document_structure_section_level_int {1}
7803   \def\thesection{\arabic{chapter}.\arabic{section}}
7804   \def\part@prefix{\arabic{chapter}.}
7805 }
7806 }{
7807   \int_set:Nn \l_document_structure_section_level_int {2}
7808   \def\part@prefix{}
7809 }
7810 }
7811
7812 \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.)`

```

7813 \renewenvironment{sfragment}[2][]{
7814   \__document_structure_sfragment_args:n { #1 }
7815   \int_incr:N \l_document_structure_section_level_int
7816   \bool_if:NT \c__notesslides_sectocframes_bool {
7817     \stepcounter{slide}
7818     \begin{frame}[noframenumbering]
7819     \vfill\Large\centering
7820     \red{
7821       \ifcase\l_document_structure_section_level_int\or
7822         \stepcounter{part}
7823         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7824         \addcontentsline{toc}{part}{\protect\numberline{\thepart}#2}
7825         \pdfbookmark[0]{\thepart\ #2}{part.\thepart}
7826         \def\currentsectionlevel{\omdoc@part@kw}
7827       \or
7828         \stepcounter{chapter}
7829         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7830         \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#2}
7831         \pdfbookmark[1]{\thechapter\ #2}{chapter.\cs_if_exist:cT{\thepart}\thepart.\thechapter}
7832         \def\currentsectionlevel{\omdoc@chapter@kw}
7833       \or
7834         \stepcounter{section}
7835         \def\__notesslideslabel{\part@prefix\arabic{section}}
7836         \addcontentsline{toc}{section}{\protect\numberline{\thesection}#2}
7837         \pdfbookmark[2]{\cs_if_exist:cT{\thechapter}{\thechapter}.\thesection\ #2}
7838         {section.\cs_if_exist:cT{\thepart}{\thepart}.\cs_if_exist:cT{\thechapter}{\thechapter}.\thepart}
7839         \def\currentsectionlevel{\omdoc@section@kw}
7840       \or
7841         \stepcounter{subsection}
7842         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7843         \addcontentsline{toc}{subsection}{\protect\numberline{\thesubsection}#2}

```

```

7844         \pdfbookmark[3]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection
7845         {subsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection}
7846         \def\currentsectionlevel{\omdoc@subsection@kw}
7847     \or
7848         \stepcounter{subsubsection}
7849         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7850         \addcontentsline{toc}{subsubsection}{\protect\numberline{\thesubsubsection}#2}
7851         \pdfbookmark[4]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\thesubsubsection}
7852         {subsubsection.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsubsection}
7853         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7854     \or
7855         \stepcounter{paragraph}
7856         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
7857         \addcontentsline{toc}{paragraph}{\protect\numberline{\theparagraph}#2}
7858         \pdfbookmark[5]{\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\thesubsection.\theparagraph}
7859         {paragraph.\cs_if_exist:cT{thepart}{\thepart}.\cs_if_exist:cT{thechapter}{\thechapter.}\thesection.\theparagraph}
7860         \def\currentsectionlevel{\omdoc@paragraph@kw}
7861     \else
7862         \def\__notesslideslabel{}
7863         \def\currentsectionlevel{\omdoc@paragraph@kw}
7864     \fi% end ifcase
7865     \__notesslideslabel\quad #2%
7866 }%
7867 \vfill%
7868 \end{frame}%
7869 }
7870 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7871     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7872 }
7873 }{}
7874 }

```

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

```

7875 \def\inserttheorembodyfont{\normalfont}
7876 %\bool_if:NF \c__notesslides_notes_bool {
7877 %   \defbeamertemplate{theorem begin}{miko}
7878 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7879 %    \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
7880 %    \inserttheorempunctuation\inserttheorembodyfont\xspace}
7881 %   \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

7882 % \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.

```

7883 % \expandafter\def\csname Parent2\endcsname{}
7884 %}
7885
7886 \AddToHook{begindocument}{% this does not work for some reason
7887     \setbeamertemplate{theorems}[ams style]
7888 }
7889 \bool_if:NT \c__notesslides_notes_bool {
7890     \renewenvironment{columns}[1][{}]{%

```

```

7891     \par\noindent%
7892     \begin{minipage}%
7893     \slidewidth\centering\leavevmode%
7894   }{%
7895     \end{minipage}\par\noindent%
7896   }%
7897   \newsavebox\columnbox%
7898   \renewenvironment<>{column}[2][]{%
7899     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7900   }{%
7901     \end{minipage}\end{lrbox}\usebox\columnbox%
7902   }%
7903 }

7904 \bool_if:NTF \c__notesslides_noproblems_bool {
7905   \newenvironment{problems}{}{}
7906 }{
7907   \excludacomment{problems}
7908 }

```

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.

```

7909 \gdef\printexcursions{}
7910 \newcommand\excursionref[2]{% label, text
7911   \bool_if:NT \c__notesslides_notes_bool {
7912     \begin{sparagraph}[title=Excursion]
7913       #2 \sref[fallback=the appendix]{#1}.
7914     \end{sparagraph}
7915   }
7916 }
7917 \newcommand\activate@excursion[2][{}{
7918   \gappto\printexcursions{\inputref{#1}{#2}}
7919 }
7920 \newcommand\excursion[4][{}{ repos, label, path, text
7921   \bool_if:NT \c__notesslides_notes_bool {
7922     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7923   }
7924 }

```

(End definition for `\excursion`. This function is documented on page 62.)

\excursiongroup

```

7925 \keys_define:nn{notesslides / excursiongroup }{
7926   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7927   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7928   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7929 }
7930 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7931   \tl_clear:N \l__notesslides_excursion_intro_tl
7932   \str_clear:N \l__notesslides_excursion_id_str

```

```

7933 \str_clear:N \l__notesslides_excursion_mhrepos_str
7934 \keys_set:nn {notesslides / excursionsgroup }{ #1 }
7935 }
7936 \newcommand\excursionsgroup[1][]{
7937   \__notesslides_excursion_args:n{ #1 }
7938   \ifdefempty\printexcursions{}% only if there are excursions
7939   {\begin{note}
7940     \begin{sfragment}[#1]{Excursions}%
7941     \ifdefempty\l__notesslides_excursion_intro_tl}{
7942       \inputref[\l__notesslides_excursion_mhrepos_str]{
7943         \l__notesslides_excursion_intro_tl
7944       }
7945     }
7946     \printexcursions%
7947     \end{sfragment}
7948   \end{note}}
7949 }
7950 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7951 \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.

```
7952 <*package>
7953 <@@=problems>
7954 \ProvidesExplPackage{problem}{2022/08/08}{3.2.0}{Semantic Markup for Problems}
7955 \RequirePackage{13keys2e}
7956 \RequirePackage{amssymb}% for \Box
7957
7958 \keys_define:nn { problem / pkg }{
7959   notes      .default:n    = { true },
7960   notes      .bool_set:N   = \c__problems_notes_bool,
7961   gnotes     .default:n    = { true },
7962   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7963   hints      .default:n    = { true },
7964   hints      .bool_set:N   = \c__problems_hints_bool,
7965   solutions  .default:n    = { true },
7966   solutions  .bool_set:N   = \c__problems_solutions_bool,
7967   pts        .default:n    = { true },
7968   pts        .bool_set:N   = \c__problems_pts_bool,
7969   min        .default:n    = { true },
7970   min        .bool_set:N   = \c__problems_min_bool,
7971   boxed      .default:n    = { true },
7972   boxed      .bool_set:N   = \c__problems_boxed_bool,
7973   unknown    .code:n       = {
7974     \PassOptionsToPackage{\CurrentOption}{stex}
7975   }
7976 }
7977 \newif\ifsolutions
7978
7979 \ProcessKeysOptions{ problem / pkg }
7980 \bool_if:NTF \c__problems_solutions_bool {
7981   \solutionstrue
7982 }{
7983   \solutionsfalse
```

```

7984 }
7985 \RequirePackage{stex}

```

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

```

7986 \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.

```

7987 \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.

```

7988 \def\prob@problem@kw{Problem}
7989 \def\prob@solution@kw{Solution}
7990 \def\prob@hint@kw{Hint}
7991 \def\prob@note@kw{Note}
7992 \def\prob@gnote@kw{Grading}
7993 \def\prob@pt@kw{pt}
7994 \def\prob@min@kw{min}
7995 \def\prob@correct@kw{Correct}
7996 \def\prob@wrong@kw{Wrong}

```

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

For the other languages, we set up triggers

```

7997 \AddToHook{begindocument}{
7998   \ltx@ifpackageloaded{babel}{
7999     \makeatletter
8000     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8001     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8002       \input{problem-ngerman.ldf}
8003     }
8004     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8005       \input{problem-finnish.ldf}
8006     }
8007     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8008       \input{problem-french.ldf}
8009     }
8010     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8011       \input{problem-russian.ldf}
8012     }
8013     \makeatother
8014   }{}
8015 }

```

39.2 Problems and Solutions

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

```

8016 \keys_define:nn{ problem / problem }{
8017   id      .str_set_x:N = \l__problems_prob_id_str,
8018   pts     .tl_set:N    = \l__problems_prob_pts_tl,
8019   min     .tl_set:N    = \l__problems_prob_min_tl,

```

```

8020 title .tl_set:N = \l__problems_prob_title_tl,
8021 type .tl_set:N = \l__problems_prob_type_tl,
8022 imports .tl_set:N = \l__problems_prob_imports_tl,
8023 name .str_set_x:N = \l__problems_prob_name_str,
8024 refnum .int_set:N = \l__problems_prob_refnum_int
8025 }
8026 \cs_new_protected:Nn \l__problems_prob_args:n {
8027 \str_clear:N \l__problems_prob_id_str
8028 \str_clear:N \l__problems_prob_name_str
8029 \tl_clear:N \l__problems_prob_pts_tl
8030 \tl_clear:N \l__problems_prob_min_tl
8031 \tl_clear:N \l__problems_prob_title_tl
8032 \tl_clear:N \l__problems_prob_type_tl
8033 \tl_clear:N \l__problems_prob_imports_tl
8034 \int_zero_new:N \l__problems_prob_refnum_int
8035 \keys_set:nn { problem / problem }{ #1 }
8036 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
8037 \let\l__problems_prob_refnum_int\undefined
8038 }
8039 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

8040 \newcounter{problem}[section]
8041 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}
8042 \def\theplainsproblem{\arabic{problem}}
8043 \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.

```

8044 \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

```

8045 \newcommand\prob@number{
8046 \int_if_exist:NTF \l__problems_inclprob_refnum_int {
8047 \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
8048 }{
8049 \int_if_exist:NTF \l__problems_prob_refnum_int {
8050 \prob@label{\int_use:N \l__problems_prob_refnum_int }
8051 }{
8052 \prob@label\theplainsproblem
8053 }
8054 }
8055 }
8056 \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.

```

8057 \newcommand\prob@title[3]{%
8058   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
8059     #2 \l__problems_inclprob_title_tl #3
8060   }{
8061     \tl_if_empty:NTF \l__problems_prob_title_tl {
8062       #1
8063     }{
8064       #2 \l__problems_prob_title_tl #3
8065     }
8066   }
8067 }

```

(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.

```

8068 \def\prob@heading{
8069   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
8070   %\sref@label{id{\prob@problem@kw-\prob@number}}{~}
8071 }

```

(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.*)

```

8072 \newenvironment{sproblem}[1][]{
8073   \__problems_prob_args:n{#1}%\sref@target%
8074   \@in@omtexttrue% we are in a statement (for inline definitions)
8075   \refstepcounter{sproblem}\record@problem
8076   \def\current@section@level{\prob@problem@kw}
8077
8078   \str_if_empty:NT \l__problems_prob_name_str {
8079     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
8080     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
8081     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
8082   }
8083
8084   \stex_if_do_html:T{
8085     \tl_if_empty:NF \l__problems_prob_title_tl {
8086       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
8087     }
8088   }
8089
8090   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
8091
8092   \stex_reactivate_macro:N \STEXexport
8093   \stex_reactivate_macro:N \importmodule
8094   \stex_reactivate_macro:N \symdecl
8095   \stex_reactivate_macro:N \notation
8096   \stex_reactivate_macro:N \symdef

```



```

8097 \stex_if_do_html:T{
8098   \begin{stex_annotate_env} {problem} {
8099     \l_stex_module_ns_str ? \l_stex_module_name_str
8100   }
8101 }
8102
8103 \stex_annotate_invisible:nnn{header}{} {
8104   \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
8105   \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
8106   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
8107     \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
8108   }
8109 }
8110 }
8111
8112 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
8113
8114
8115 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
8116   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
8117 }{
8118   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
8119 }
8120 \str_if_exist:NTF \l__problems_inclprob_id_str {
8121   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
8122 }{
8123   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
8124 }
8125
8126
8127 \stex_if_smsmode:F {
8128   \clist_set:No \l_tmpa_clist \sproblemtype
8129   \tl_clear:N \l_tmpa_tl
8130   \clist_map_inline:Nn \l_tmpa_clist {
8131     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
8132       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
8133     }
8134   }
8135   \tl_if_empty:NTF \l_tmpa_tl {
8136     \__problems_sproblem_start:
8137   }{
8138     \l_tmpa_tl
8139   }
8140 }
8141 \stex_ref_new_doc_target:n \sproblemid
8142 \stex_if_smsmode:TF \stex_smsmode_do: \ignorespacesandpars
8143 }{
8144   \__stex_modules_end_module:
8145   \stex_if_smsmode:F{
8146     \clist_set:No \l_tmpa_clist \sproblemtype
8147     \tl_clear:N \l_tmpa_tl
8148     \clist_map_inline:Nn \l_tmpa_clist {
8149       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
8150         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}

```

```

8151     }
8152   }
8153   \tl_if_empty:NTF \l_tmpa_tl {
8154     \__problems_sproblem_end:
8155   }{
8156     \l_tmpa_tl
8157   }
8158 }
8159 \stex_if_do_html:T{
8160   \end{stex_annotate_env}
8161 }
8162
8163 \smallskip
8164 }
8165
8166 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
8167
8168
8169
8170 \cs_new_protected:Nn \__problems_sproblem_start: {
8171   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
8172 }
8173 \cs_new_protected:Nn \__problems_sproblem_end: { \par\smallskip}
8174
8175 \newcommand\stexpatchproblem[3][] {
8176   \str_set:Nx \l_tmpa_str{ #1 }
8177   \str_if_empty:NTF \l_tmpa_str {
8178     \tl_set:Nn \__problems_sproblem_start: { #2 }
8179     \tl_set:Nn \__problems_sproblem_end: { #3 }
8180   }{
8181     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
8182     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
8183   }
8184 }
8185
8186
8187 \bool_if:NT \c__problems_boxed_bool {
8188   \surroundwithhmdframed{problem}
8189 }

```

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

```

8190 \def\record@problem{
8191   \protected@write\auxout{}
8192   {
8193     \string\@problem{\prob@number}
8194   {
8195     \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8196       \l__problems_inclprob_pts_tl
8197     }{
8198       \l__problems_prob_pts_tl
8199     }
8200   }%
8201   {
8202     \tl_if_exist:NTF \l__problems_inclprob_min_tl {

```

```

8203         \l__problems_inclprob_min_tl
8204     }{
8205         \l__problems_prob_min_tl
8206     }
8207 }
8208 }
8209 }

```

(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).

```

8210 \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.

```

8211 \keys_define:nn { problem / solution }{
8212   id          .str_set_x:N = \l__problems_solution_id_str ,
8213   for         .str_set_x:N = \l__problems_solution_for_str ,
8214   type        .str_set_x:N = \l__problems_solution_type_str ,
8215   title       .tl_set:N    = \l__problems_solution_title_tl
8216 }
8217 \cs_new_protected:Nn \__problems_solution_args:n {
8218   \str_clear:N \l__problems_solution_id_str
8219   \str_clear:N \l__problems_solution_type_str
8220   \str_clear:N \l__problems_solution_for_str
8221   \tl_clear:N \l__problems_solution_title_tl
8222   \keys_set:nn { problem / solution }{ #1 }
8223 }

```

\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.

```

8224 \box_new:N \l__problems_solution_box
8225 \newenvironment{solution}[1][{}{
8226   \__problems_solution_args:n{#1}
8227   \stex_html_backend:TF{
8228     \stex_if_do_html:T{
8229       \begin{stex_annotate_env}{solution}{}
8230       \str_if_empty:NF \l__problems_solution_type_str {
8231         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
8232       }
8233       \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problem
8234     }
8235   }{
8236     \setbox\l__problems_solution_box\vbox\bgroup
8237     \par\smallskip\hrule\smallskip
8238     \noindent\textbf{Solution}\tl_if_empty:NF\l__problems_solution_title_tl{~(\l__problems_
8239   }
8240 }{
8241   \stex_html_backend:TF{
8242     \stex_if_do_html:T{
8243       \end{stex_annotate_env}

```

```

8244     }
8245   }{
8246     \smallskip\hrule
8247     \egroup
8248     \bool_if:NT \c__problems_solutions_bool {
8249       \box\l__problems_solution_box
8250     }
8251   }
8252 }
8253
8254 \newcommand\startsolutions{
8255   \bool_set_true:N \c__problems_solutions_bool
8256   \solutionstrue
8257   % \specialcomment{solution}{\@startsolution}{
8258   %   \bool_if:NF \c__problems_boxed_bool {
8259   %     \hrule\medskip
8260   %   }
8261   %   \end{small}%
8262   % }
8263   % \bool_if:NT \c__problems_boxed_bool {
8264   %   \surroundwithmdframed{solution}
8265   % }
8266 }

```

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

\stopsolutions

```

8267 \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.)

```

8268 \bool_if:NTF \c__problems_notes_bool {
8269   \newenvironment{exnote}[1][ ]{
8270     \par\smallskip\hrule\smallskip
8271     \noindent\textbf{\prob@note@kw :~ }\small
8272   }{
8273     \smallskip\hrule
8274   }
8275 }{
8276   \excludacomment{exnote}
8277 }

```

hint (env.)

```

8278 \bool_if:NTF \c__problems_notes_bool {
8279   \newenvironment{hint}[1][ ]{
8280     \par\smallskip\hrule\smallskip
8281     \noindent\textbf{\prob@hint@kw :~ }\small
8282   }{
8283     \smallskip\hrule
8284   }
8285   \newenvironment{exhint}[1][ ]{
8286     \par\smallskip\hrule\smallskip
8287     \noindent\textbf{\prob@hint@kw :~ }\small

```

```

8288 }{
8289     \smallskip\hrule
8290 }
8291 }{
8292     \excludecomment{hint}
8293     \excludecomment{exhint}
8294 }

```

gnote (env.)

```

8295 \bool_if:NTF \c__problems_notes_bool {
8296     \newenvironment{gnote}[1][]{
8297         \par\smallskip\hrule\smallskip
8298         \noindent\textbf{\prob@gnote@kw :~ }\small
8299     }{
8300         \smallskip\hrule
8301     }
8302 }{
8303     \excludecomment{gnote}
8304 }

```

39.3 Marup for Added Value Services

39.4 Multiple Choice Blocks

EdN:12

mcb (env.) ¹²

```

8305 \newenvironment{mcb}{
8306     \begin{enumerate}
8307 }{
8308     \end{enumerate}
8309 }

```

we define the keys for the mcb macro

```

8310 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
8311     \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
8312         \bool_set_true:N #1
8313     }{
8314         \bool_set_false:N #1
8315     }
8316 }
8317 \keys_define:nn { problem / mcb }{
8318     id .str_set_x:N = \l__problems_mcc_id_str ,
8319     feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
8320     T .default:n = { false } ,
8321     T .bool_set:N = \l__problems_mcc_t_bool ,
8322     F .default:n = { false } ,
8323     F .bool_set:N = \l__problems_mcc_f_bool ,
8324     Ttext .tl_set:N = \l__problems_mcc_Ttext_tl ,
8325     Ftext .tl_set:N = \l__problems_mcc_Ftext_tl
8326 }
8327 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

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

```

8328 \str_clear:N \l__problems_mcc_id_str
8329 \tl_clear:N \l__problems_mcc_feedback_tl
8330 \bool_set_false:N \l__problems_mcc_t_bool
8331 \bool_set_false:N \l__problems_mcc_f_bool
8332 \tl_clear:N \l__problems_mcc_Ttext_tl
8333 \tl_clear:N \l__problems_mcc_Ftext_tl
8334 \str_clear:N \l__problems_mcc_id_str
8335 \keys_set:nn { problem / mcc }{ #1 }
8336 }

```

\mcc

```

8337 \def\mccTrueText{\textbf{\prob@correct@kw!~}}
8338 \def\mccFalseText{\textbf{\prob@wrong@kw!~}}
8339 \newcommand\mcc[2][{}]{
8340   \l__problems_mcc_args:n{ #1 }
8341   \item[$\Box$] #2
8342   \bool_if:NT \c__problems_solutions_bool{
8343     \\\
8344     \bool_if:NT \l__problems_mcc_t_bool {
8345       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
8346     }
8347     \bool_if:NT \l__problems_mcc_f_bool {
8348       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
8349     }
8350     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8351       \emph{\l__problems_mcc_feedback_tl}
8352     }
8353   }
8354 } %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.

```

8355 \newcommand\fillinsol[1]{\quad%
8356   \ifsolutions\textcolor{red}{\#!}\else%
8357   \fbox{\phantom{\huge{\#!}}}%
8358   \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.

```

8359
8360 \keys_define:nn{ problem / inclproblem }{
8361   id      .str_set_x:N = \l__problems_inclprob_id_str,
8362   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8363   min     .tl_set:N    = \l__problems_inclprob_min_tl,

```

```

8364 title .tl_set:N = \l__problems_inclprob_title_tl,
8365 refnum .int_set:N = \l__problems_inclprob_refnum_int,
8366 type .tl_set:N = \l__problems_inclprob_type_tl,
8367 mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8368 }
8369 \cs_new_protected:Nn \l__problems_inclprob_args:n {
8370 \str_clear:N \l__problems_prob_id_str
8371 \tl_clear:N \l__problems_inclprob_pts_tl
8372 \tl_clear:N \l__problems_inclprob_min_tl
8373 \tl_clear:N \l__problems_inclprob_title_tl
8374 \tl_clear:N \l__problems_inclprob_type_tl
8375 \int_zero_new:N \l__problems_inclprob_refnum_int
8376 \str_clear:N \l__problems_inclprob_mhrepos_str
8377 \keys_set:nn { problem / inclproblem }{ #1 }
8378 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8379 \let\l__problems_inclprob_pts_tl\undefined
8380 }
8381 \tl_if_empty:NT \l__problems_inclprob_min_tl {
8382 \let\l__problems_inclprob_min_tl\undefined
8383 }
8384 \tl_if_empty:NT \l__problems_inclprob_title_tl {
8385 \let\l__problems_inclprob_title_tl\undefined
8386 }
8387 \tl_if_empty:NT \l__problems_inclprob_type_tl {
8388 \let\l__problems_inclprob_type_tl\undefined
8389 }
8390 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8391 \let\l__problems_inclprob_refnum_int\undefined
8392 }
8393 }
8394
8395 \cs_new_protected:Nn \l__problems_inclprob_clear: {
8396 \let\l__problems_inclprob_id_str\undefined
8397 \let\l__problems_inclprob_pts_tl\undefined
8398 \let\l__problems_inclprob_min_tl\undefined
8399 \let\l__problems_inclprob_title_tl\undefined
8400 \let\l__problems_inclprob_type_tl\undefined
8401 \let\l__problems_inclprob_refnum_int\undefined
8402 \let\l__problems_inclprob_mhrepos_str\undefined
8403 }
8404 \l__problems_inclprob_clear:
8405
8406 \newcommand\includeproblem[2][]{
8407 \l__problems_inclprob_args:n{ #1 }
8408 \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8409 \stex_html_backend:TF {
8410 \str_clear:N \l_tmpa_str
8411 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8412 \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8413 }
8414 \stex_annotate_invisible:nnn{includeproblem}{
8415 \l_tmpa_str / #2
8416 }{}}
8417 }{

```

```

8418     \begingroup
8419     \inputreftrue
8420     \tl_if_empty:nTF{ ##1 }{
8421       \input{#2}
8422     }{
8423       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8424     }
8425   \endgroup
8426 }
8427 }
8428 \__problems_inclprob_clear:
8429 }

```

(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.

```

8430 \AddToHook{enddocument}{
8431   \bool_if:NT \c__problems_pts_bool {
8432     \message{Total:~\arabic{pts}~points}
8433   }
8434   \bool_if:NT \c__problems_min_bool {
8435     \message{Total:~\arabic{min}~minutes}
8436   }
8437 }

```

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

```

8438 \def\pts#1{
8439   \bool_if:NT \c__problems_pts_bool {
8440     \marginpar{#1~\prob@pt@kw}
8441   }
8442 }
8443 \def\min#1{
8444   \bool_if:NT \c__problems_min_bool {
8445     \marginpar{#1~\prob@min@kw}
8446   }
8447 }

```

`\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.

```

8448 \newcounter{pts}
8449 \def\show@pts{
8450   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8451     \bool_if:NT \c__problems_pts_bool {
8452       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8453       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8454     }
8455   }{
8456     \tl_if_exist:NT \l__problems_prob_pts_tl {
8457       \bool_if:NT \c__problems_pts_bool {

```



```

8458         \tl_if_empty:NT\l__problems_prob_pts_tl{
8459             \tl_set:Nn \l__problems_prob_pts_tl {0}
8460         }
8461         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8462         \addtocounter{pts}{\l__problems_prob_pts_tl}
8463     }
8464 }
8465 }
8466 }

```

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

\show@min

```

8467 \newcounter{min}
8468 \def\show@min{
8469     \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8470         \bool_if:NT \c__problems_min_bool {
8471             \marginpar{\l__problems_inclprob_pts_tl\ min}
8472             \addtocounter{min}{\l__problems_inclprob_min_tl}
8473         }
8474     }{
8475         \tl_if_exist:NT \l__problems_prob_min_tl {
8476             \bool_if:NT \c__problems_min_bool {
8477                 \tl_if_empty:NT\l__problems_prob_min_tl{
8478                     \tl_set:Nn \l__problems_prob_min_tl {0}
8479                 }
8480                 \marginpar{\l__problems_prob_min_tl\ min}
8481                 \addtocounter{min}{\l__problems_prob_min_tl}
8482             }
8483         }
8484     }
8485 }
8486 \end{package}

```

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

Chapter 40

Implementation: The hwexam Package

40.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
8487 \*package>
8488 \ProvidesExplPackage{hwexam}{2022/08/08}{3.2.0}{homework assignments and exams}
8489 \RequirePackage{13keys2e}
8490
8491 \newif\iftest\testfalse
8492 \DeclareOption{test}{\testtrue}
8493 \newif\ifmultiple\multiplefalse
8494 \DeclareOption{multiple}{\multipletrue}
8495 \DeclareOption{lang}{\PassOptionsToPackage{\CurrentOption}{problem}}
8496 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8497 \ProcessOptions
```

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

```
8498 \RequirePackage{keyval}[1997/11/10]
8499 \RequirePackage{problem}
```

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

```
8500 \newcommand\hwexam@assignment@kw{Assignment}
8501 \newcommand\hwexam@given@kw{Given}
8502 \newcommand\hwexam@due@kw{Due}
8503 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~blank~for~extra~space}
8504 \newcommand\hwexam@minutes@kw{minutes}
8505 \newcommand\correction@probs@kw{prob.}
8506 \newcommand\correction@pts@kw{total}
8507 \newcommand\correction@reached@kw{reached}
8508 \newcommand\correction@sum@kw{Sum}
8509 \newcommand\correction@grade@kw{grade}
8510 \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

```

8511 \AddToHook{begindocument}{
8512 \ltx@ifpackageloaded{babel}{
8513 \makeatletter
8514 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8515 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{ngerman}}{
8516 \input{hwexam-ngerman.ldf}
8517 }
8518 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{finnish}}{
8519 \input{hwexam-finnish.ldf}
8520 }
8521 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{french}}{
8522 \input{hwexam-french.ldf}
8523 }
8524 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\detokenize{russian}}{
8525 \input{hwexam-russian.ldf}
8526 }
8527 \makeatother
8528 }{}
8529 }
8530

```

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.

```

8531 \newcounter{assignment}
8532 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

8533 \keys_define:nn { hwexam / assignment } {
8534 id .str_set:N = \l_@@_assign_id_str,
8535 number .int_set:N = \l_@@_assign_number_int,
8536 title .tl_set:N = \l_@@_assign_title_tl,
8537 type .tl_set:N = \l_@@_assign_type_tl,
8538 given .tl_set:N = \l_@@_assign_given_tl,
8539 due .tl_set:N = \l_@@_assign_due_tl,
8540 loadmodules .code:n = {
8541 \bool_set_true:N \l_@@_assign_loadmodules_bool
8542 }
8543 }
8544 \cs_new_protected:Nn \_@@_assignment_args:n {
8545 \str_clear:N \l_@@_assign_id_str
8546 \int_set:Nn \l_@@_assign_number_int {-1}
8547 \tl_clear:N \l_@@_assign_title_tl
8548 \tl_clear:N \l_@@_assign_type_tl
8549 \tl_clear:N \l_@@_assign_given_tl
8550 \tl_clear:N \l_@@_assign_due_tl
8551 \bool_set_false:N \l_@@_assign_loadmodules_bool
8552 \keys_set:nn { hwexam / assignment }{ #1 }
8553 }

```

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.

```

8554 \newcommand\given@due[2]{
8555 \bool_lazy_all:nF {
8556 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8557 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8558 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8559 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8560 }{ #1 }
8561
8562 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8563 \tl_if_empty:NF \l_@@_assign_given_tl {
8564 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8565 }
8566 }{
8567 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8568 }
8569
8570 \bool_lazy_or:nnF {
8571 \bool_lazy_and_p:nn {
8572 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8573 }{
8574 \tl_if_empty_p:V \l_@@_assign_due_tl
8575 }
8576 }{
8577 \bool_lazy_and_p:nn {
8578 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8579 }{
8580 \tl_if_empty_p:V \l_@@_assign_due_tl
8581 }
8582 }{ ,~ }
8583
8584 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8585 \tl_if_empty:NF \l_@@_assign_due_tl {
8586 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8587 }
8588 }{
8589 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8590 }
8591
8592 \bool_lazy_all:nF {
8593 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8594 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8595 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8596 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8597 }{ #2 }
8598 }

```

`\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.

```

8599 \newcommand\assignment@title[3]{
8600 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8601 \tl_if_empty:NTF \l_@@_assign_title_tl {
8602 #1
8603 }{
8604 #2\l_@@_assign_title_tl#3
8605 }
8606 }{
8607 #2\l_@@_inclasssign_title_tl#3
8608 }
8609 }

```

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

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

```

8610 \newcommand\assignment@number{
8611 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8612 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8613 \arabic{assignment}
8614 } {
8615 \int_use:N \l_@@_assign_number_int
8616 }
8617 }{
8618 \int_use:N \l_@@_inclasssign_number_int
8619 }
8620 }

```

(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.

```

8621 \newenvironment{assignment}[1][]{
8622 \_@@_assignment_args:n { #1 }
8623 %\sref@target
8624 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8625 \global\stepcounter{assignment}
8626 }{
8627 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8628 }
8629 \setcounter{sproblem}{0}
8630 \renewcommand\prob@label[1]{\assignment@number.##1}
8631 \def\current@section@level{\document@hwexamtype}
8632 %\sref@label{id{\document@hwexamtype \thesection}
8633 \begin{@assignment}
8634 }{
8635 \end{@assignment}
8636 }

```

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

```

8637 \def\ass@title{
8638 {\protect\document@hwexamtype}\arabic{assignment}
8639 \assignment@title{}\;{}{}\;} -- \given@due{}\}
8640 }
8641 \ifmultiple
8642 \newenvironment{@assignment}{
8643 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8644 \begin{sfragment}[loadmodules]{\ass@title}
8645 }{
8646 \begin{sfragment}{\ass@title}
8647 }
8648 }{
8649 \end{sfragment}
8650 }

```

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

```

8651 \else
8652 \newenvironment{@assignment}{
8653 \begin{center}\bf
8654 \Large@title\strut\
8655 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;{}{}\;{}\\}
8656 \large\given@due{--\;}\;{}{--}
8657 \end{center}
8658 }{}
8659 \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.

```

8660 \keys_define:nn { hwexam / inclassignment } {
8661 %id .str_set_x:N = \l_@@_assign_id_str,
8662 number .int_set:N = \l_@@_inclassign_number_int,
8663 title .tl_set:N = \l_@@_inclassign_title_tl,
8664 type .tl_set:N = \l_@@_inclassign_type_tl,
8665 given .tl_set:N = \l_@@_inclassign_given_tl,
8666 due .tl_set:N = \l_@@_inclassign_due_tl,
8667 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8668 }
8669 \cs_new_protected:Nn \_@@_inclassignment_args:n {
8670 \int_set:Nn \l_@@_inclassign_number_int {-1}
8671 \tl_clear:N \l_@@_inclassign_title_tl
8672 \tl_clear:N \l_@@_inclassign_type_tl
8673 \tl_clear:N \l_@@_inclassign_given_tl
8674 \tl_clear:N \l_@@_inclassign_due_tl
8675 \str_clear:N \l_@@_inclassign_mhrepos_str
8676 \keys_set:nn { hwexam / inclassignment }{ #1 }
8677 }
8678 \_@@_inclassignment_args:n {}
8679
8680 \newcommand\inputassignment[2][{}]{

```

```

8681 \_@@_inclassassignment_args:n { #1 }
8682 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8683   \input{#2}
8684 }{
8685   \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8686     \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8687   }
8688 }
8689 \_@@_inclassassignment_args:n {}
8690 }
8691 \newcommand\includeassignment[2][]{
8692   \newpage
8693   \inputassignment[#1]{#2}
8694 }

```

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

40.4 Typesetting Exams

\quizheading

```

8695 \ExplSyntaxOff
8696 \newcommand\quizheading[1]{%
8697   \def\@tas{#1}%
8698   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8699   \ifx\@tas\@empty\else%
8700     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8701   \fi%
8702 }
8703 \ExplSyntaxOn

```

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

\testheading

```

8704
8705 \def\hwexamheader{\input{hwexam-default.header}}
8706
8707 \def\hwexamminutes{
8708   \tl_if_empty:NTF \testheading@duration {
8709     {\testheading@min}~\hwexam@minutes@kw
8710   }{
8711     \testheading@duration
8712   }
8713 }
8714
8715 \keys_define:nn { hwexam / testheading } {
8716   min .tl_set:N = \testheading@min,
8717   duration .tl_set:N = \testheading@duration,
8718   reqpts .tl_set:N = \testheading@reqpts,
8719   tools .tl_set:N = \testheading@tools
8720 }
8721 \cs_new_protected:Nn \_@@_testheading_args:n {
8722   \tl_clear:N \testheading@min
8723   \tl_clear:N \testheading@duration

```

```

8724 \tl_clear:N \testheading@reqpts
8725 \tl_clear:N \testheading@tools
8726 \keys_set:nn { hwexam / testheading }{ #1 }
8727 }
8728 \newenvironment{testheading}[1][ ]{
8729 \_@@_testheading_args:n{ #1 }
8730 \newcount\check@time\check@time=\testheading@min
8731 \advance\check@time by -\theassignment@totalmin
8732 \newif\if@bonuspoints
8733 \tl_if_empty:NTF \testheading@reqpts {
8734 \@bonuspointsfalse
8735 }{
8736 \newcount\bonus@pts
8737 \bonus@pts=\theassignment@totalpts
8738 \advance\bonus@pts by -\testheading@reqpts
8739 \edef\bonus@pts{\the\bonus@pts}
8740 \@bonuspointstrue
8741 }
8742 \edef\check@time{\the\check@time}
8743
8744 \makeatletter\hwexamheader\makeatother
8745 }{
8746 \newpage
8747 }

```

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

\testspace

```

8748 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8749 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8750 \newcommand\testemptypage[1][ ]{\iftest\begin{center}\hwexam@testemptypage@kw\end{center}\vfi}

```

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

\@problem This macro acts on a problem's record in the *.aux file. Here we redefine it (it was defined to do nothing in problem.sty) to generate the correction table.

```

8751 <@@=problems>
8752 \renewcommand\@problem[3]{
8753 \stepcounter{assignment@probs}
8754 \def\__problemspts{#2}
8755 \ifx\__problemspts\@empty\else
8756 \addtocounter{assignment@totalpts}{#2}
8757 \fi
8758 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8759 \xdef\correction@probs{\correction@probs & #1}%
8760 \xdef\correction@pts{\correction@pts & #2}
8761 \xdef\correction@reached{\correction@reached &}

```



```

8762 }
8763 <@@=hwexam>

```

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

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

```

8764 \newcounter{assignment@probs}
8765 \newcounter{assignment@totalpts}
8766 \newcounter{assignment@totalmin}
8767 \def\correction@probs{\correction@probs@kw}
8768 \def\correction@pts{\correction@pts@kw}
8769 \def\correction@reached{\correction@reached@kw}
8770 \stepcounter{assignment@probs}
8771 \newcommand\correction@table{
8772 \resizebox{\textwidth}{!}{%
8773 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8774 &\multicolumn{\theassignment@probs}{c|}|%|
8775 {\footnotesize\correction@forgrading@kw} &\\\hline
8776 \correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline
8777 \correction@pts & \theassignment@totalpts & \\\hline
8778 \correction@reached & & \[.7cm]\hline
8779 \end{tabular}}
8780 </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/graphics.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

¹³EdNOTE: we need an un-numbered version sfragment*

- [SIa] *sLaTeX/sTeX-IDE*. URL: <https://github.com/slatex/sTeX-IDE> (visited on 04/22/2022).
- [SIb] *sLaTeX/stexls-vscode-plugin*. URL: <https://github.com/slatex/stexls-vscode-plugin> (visited on 04/22/2022).
- [SLS] *sLaTeX/stexls*. URL: <https://github.com/slatex/stexls> (visited on 04/22/2022).
- [ST] *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: <https://ctan.org/pkg/stex> (visited on 04/22/2022).
- [sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: <https://github.com/sLaTeX/sTeX> (visited on 05/11/2020).
- [Tana] Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: <http://ctan.org/pkg/beamer> (visited on 01/07/2014).
- [Tanb] Till Tantau. *User Guide to the Beamer Class*. URL: <http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.
- [TL] *TeX Live*. URL: <http://www.tug.org/texlive/> (visited on 12/11/2012).