

The sTeX3 Package *

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

2022-02-15

Abstract

sTeX is a collection of L^AT_EX package that allow to markup documents semantically without leaving the document format, 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,
- 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 3.0 (last revised 2022-02-15)

Contents

| | | |
|-----------|---|-----------|
| I | Manual | 1 |
| 1 | What is sTeX? | 2 |
| 2 | Quickstart | 3 |
| 2.1 | Setup | 3 |
| 2.1.1 | The sTeX IDE | 3 |
| 2.1.2 | Manual Setup | 3 |
| 2.2 | A First sTeX Document | 4 |
| 3 | Using Semantic Macros | 6 |
| 4 | sTeX Archives | 7 |
| 4.1 | The Local MathHub-Directory | 7 |
| 4.2 | The Structure of sTeX Archives | 7 |
| 4.3 | MANIFEST.MF-Files | 8 |
| 5 | Creating New Modules and Symbols | 9 |
| 5.1 | Advanced Structuring Mechanisms | 9 |
| 5.2 | Primitive Symbols (The sTeX Metatheory) | 10 |
| 6 | sTeX Statements (Definitions, Theorems, Examples, ...) | 11 |
| 7 | Additional Packages | 12 |
| 7.1 | Modular Document Structuring | 12 |
| 7.2 | Slides and Course Notes | 12 |
| 7.3 | Homework, Problems and Exams | 12 |
| 8 | Stuff | 13 |
| 8.1 | Modules | 13 |
| 8.1.1 | Semantic Macros and Notations | 13 |
| | Other Argument Types | 15 |
| | Precedences | 17 |
| 8.1.2 | Archives and Imports | 17 |
| | Namespaces | 17 |
| | Paths in Import-Statements | 18 |
| II | Documentation | 19 |
| 9 | sTeX-Basics | 20 |
| 9.1 | Macros and Environments | 20 |
| 10 | sTeX-MathHub | 22 |
| 10.1 | Macros and Environments | 22 |
| 10.1.1 | Files, Paths, URIs | 22 |
| 10.1.2 | MathHub Archives | 23 |

| | | |
|------------|--|-----------|
| 11 | sTeX-References | 25 |
| 11.1 | Macros and Environments | 25 |
| 12 | sTeX-Modules | 26 |
| 12.1 | Macros and Environments | 26 |
| 12.1.1 | The <code>module</code> -environment | 28 |
| 13 | sTeX-Module Inheritance | 31 |
| 13.1 | Macros and Environments | 31 |
| 13.1.1 | SMS Mode | 31 |
| 13.1.2 | Imports and Inheritance | 32 |
| 14 | sTeX-Symbols | 35 |
| 14.1 | Macros and Environments | 35 |
| 15 | sTeX-Terms | 38 |
| 15.1 | Macros and Environments | 38 |
| 16 | sTeX-Structural Features | 41 |
| 16.1 | Macros and Environments | 41 |
| 16.1.1 | Structures | 41 |
| 17 | sTeX-Statements | 42 |
| 17.1 | Macros and Environments | 42 |
| 18 | sTeX-Proofs: Structural Markup for Proofs | 43 |
| 18.1 | Introduction | 45 |
| 18.2 | The User Interface | 46 |
| 18.2.1 | Package Options | 46 |
| 18.2.2 | Proofs and Proof steps | 46 |
| 18.2.3 | Justifications | 46 |
| 18.2.4 | Proof Structure | 47 |
| 18.2.5 | Proof End Markers | 48 |
| 18.2.6 | Configuration of the Presentation | 48 |
| 18.3 | Limitations | 48 |
| 19 | sTeX-Metatheory | 50 |
| 19.1 | Symbols | 50 |
| III | Extensions | 51 |
| 20 | Tikzinput | 52 |
| 20.1 | Macros and Environments | 52 |

| | |
|---|-----------|
| 21 document-structure: Semantic Markup for Open Mathematical Documents in \LaTeX | 53 |
| 21.1 Introduction | 53 |
| 21.2 The User Interface | 54 |
| 21.2.1 Package and Class Options | 54 |
| 21.2.2 Document Structure | 54 |
| 21.2.3 Ignoring Inputs | 56 |
| 21.2.4 Structure Sharing | 56 |
| 21.2.5 Global Variables | 56 |
| 21.2.6 Colors | 57 |
| 21.3 Limitations | 57 |
| 22 NotesSlides – Slides and Course Notes | 58 |
| 22.1 Introduction | 58 |
| 22.2 The User Interface | 58 |
| 22.2.1 Package Options | 58 |
| 22.2.2 Notes and Slides | 59 |
| 22.2.3 Header and Footer Lines of the Slides | 60 |
| 22.2.4 Frame Images | 60 |
| 22.2.5 Colors and Highlighting | 61 |
| 22.2.6 Front Matter, Titles, etc. | 61 |
| 22.2.7 Excursions | 61 |
| 22.2.8 Miscellaneous | 62 |
| 22.3 Limitations | 62 |
| 23 problem.sty: An Infrastructure for formatting Problems | 63 |
| 23.1 Introduction | 63 |
| 23.2 The User Interface | 63 |
| 23.2.1 Package Options | 63 |
| 23.2.2 Problems and Solutions | 64 |
| 23.2.3 Multiple Choice Blocks | 65 |
| 23.2.4 Including Problems | 65 |
| 23.2.5 Reporting Metadata | 65 |
| 23.3 Limitations | 65 |
| 24 hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams | 67 |
| 24.1 Introduction | 68 |
| 24.2 The User Interface | 68 |
| 24.2.1 Package and Class Options | 68 |
| 24.2.2 Assignments | 68 |
| 24.2.3 Typesetting Exams | 68 |
| 24.2.4 Including Assignments | 69 |
| 24.3 Limitations | 69 |
| IV Implementation | 71 |

| | | |
|-----------|--|------------|
| 25 | STeX-Basics Implementation | 72 |
| 25.1 | The STeXDocument Class | 72 |
| 25.2 | Preliminaries | 72 |
| 25.3 | Messages and logging | 73 |
| 25.4 | Persistence | 74 |
| 25.5 | HTML Annotations | 74 |
| 25.6 | Languages | 77 |
| 25.7 | Activating/Deactivating Macros | 78 |
| 26 | STeX-MathHub Implementation | 80 |
| 26.1 | Generic Path Handling | 80 |
| 26.2 | PWD and kpsewhich | 82 |
| 26.3 | File Hooks and Tracking | 83 |
| 26.4 | MathHub Repositories | 84 |
| 27 | STeX-References Implementation | 92 |
| 27.1 | Document URIs and URLs | 92 |
| 27.2 | Setting Reference Targets | 94 |
| 27.3 | Using References | 95 |
| 28 | STeX-Modules Implementation | 98 |
| 28.1 | The module environment | 101 |
| 28.2 | Invoking modules | 107 |
| 29 | STeX-Module Inheritance Implementation | 109 |
| 29.1 | SMS Mode | 109 |
| 29.2 | Inheritance | 112 |
| 30 | STeX-Symbols Implementation | 117 |
| 30.1 | Symbol Declarations | 117 |
| 30.2 | Notations | 124 |
| 31 | STeX-Terms Implementation | 134 |
| 31.1 | Symbol Invocations | 134 |
| 31.2 | Terms | 137 |
| 31.3 | Notation Components | 144 |
| 32 | STeX-Structural Features Implementation | 147 |
| 32.1 | Imports with modification | 147 |
| 32.2 | The feature environment | 154 |
| 32.3 | Features | 156 |
| 33 | STeX-Statements Implementation | 161 |
| 33.1 | Definitions | 161 |
| 33.2 | Assertions | 166 |
| 33.3 | Examples | 169 |
| 33.4 | Logical Paragraphs | 171 |

| | |
|---|------------|
| 34 The Implementation | 176 |
| 34.1 Package Options | 176 |
| 34.2 Proofs | 176 |
| 34.3 Justifications | 182 |
| 35 \LaTeX-Others Implementation | 184 |
| 36 \LaTeX-Metatheory Implementation | 185 |
| 37 Tikzinput Implementation | 188 |
| 38 document-structure.sty Implementation | 190 |
| 38.1 The document-structure Class | 190 |
| 38.2 Class Options | 190 |
| 38.3 Beefing up the <code>document</code> environment | 191 |
| 38.4 Implementation: document-structure Package | 191 |
| 38.5 Package Options | 191 |
| 38.6 Document Structure | 193 |
| 38.7 Front and Backmatter | 196 |
| 38.8 Global Variables | 198 |
| 39 NotesSlides – Implementation | 199 |
| 39.1 Class and Package Options | 199 |
| 39.2 Notes and Slides | 201 |
| 39.3 Header and Footer Lines | 205 |
| 39.4 Frame Images | 206 |
| 39.5 Colors and Highlighting | 207 |
| 39.6 Sectioning | 208 |
| 39.7 Excursions | 210 |
| 40 The Implementation | 212 |
| 40.1 Package Options | 212 |
| 40.2 Problems and Solutions | 213 |
| 40.3 Multiple Choice Blocks | 219 |
| 40.4 Including Problems | 220 |
| 40.5 Reporting Metadata | 221 |
| 41 Implementation: The hwexam Class | 223 |
| 41.1 Class Options | 223 |
| 42 Implementation: The hwexam Package | 225 |
| 42.1 Package Options | 225 |
| 42.2 Assignments | 226 |
| 42.3 Including Assignments | 229 |
| 42.4 Typesetting Exams | 230 |
| 42.5 Leftovers | 232 |

Part I
Manual

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 to use semantic annotations in L^AT_EX documents,
- RuSTeX to convert `tex` sources to (semantically enriched) `xhtml`,
- The MMT software, that extracts semantic information from the thus generated `xhtml` and provides semantically informed added value services.

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

Foregoing on the sTeX IDE, we will need several pieces of software; namely:

- **The sTeX-Package** available [here](#)¹. Note, that the CTAN repository for L^AT_EX packages may contain outdated versions of the sTeX package, so make sure, that your TEXMF system variable is configured such that the packages available in the linked repository are prioritized over potential default packages that come with your T_EX distribution.

- **The Mmt System** available [here](#)². We recommend following the setup routine documented [here](#).

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

- To make sure that sTeX too knows where to find its archives, we need to set a global system variable **MATHHUB**, that points to your local **MathHub**-directory (see [chapter 4](#)).

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

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

¹EdNOTE: For now, we require the latex3-branch

²EdNOTE: For now, we require the sTeX-branch, requiring manually compiling the MMT sources

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

2.2 A First s_TE_X Document

Having set everything up, we can write a first s_TE_X document. As an example, we will use the `smglom/calculus` and `smglom/arithmetics` archives, which should be present in the designated MathHub-folder.

The document we will consider is the following:

```
\documentclass{article}
\usepackage{stex}
\usepackage{xcolor}
\def\compemph#1{\textcolor{blue}{#1}}

\begin{document}
\usemodule[smglom/calculus]{series}
\usemodule[smglom/arithmetics]{realarith}

The \symref{series}{series}  $\sum_{n=1}^{\infty} \frac{1}{2^n}$ 
\realdivide[\frac]{1}{2}
\realpower{2}{n}
\symref{converges}{converges} towards 1$.
\end{document}
```

Compiling this document with `pdflatex` should yield the output

The **series** $\sum_{n=1}^{\infty} \frac{1}{2^n}$ **converges** towards 1.

Note that the \sum and ∞ -symbols are highlighted in blue, and the words “series” and “converges” in bold. This signifies that these words and symbols reference s_TE_X *symbols* formally declared somewhere; associating their *presentation* in the document with their (formal) definition - i.e. their semantics. The precise way in which they are highlighted (if at all) can of course be customized (see ³).

\usemodule

The command `\usemodule[some/archive]{modulename}` finds some module in the appropriate archive – in the first case (`\usemodule[smglom/calculus]{series}`), s_TE_X looks for the archive `smglom/calculus` in our local MathHub-directory (see [chapter 4](#)), and in its source-folder for a file `series.tex`. Since no such file exists, and by default the document is assumed to be in *english*, it picks the file `series.en.tex`, and indeed, in here we find a statement `\begin{smodule}{series}`.

s_TE_X now reads this file and makes all semantic macros therein available to use, along with all its dependencies. This enables the usage of `\infinitiesum` later on.

Analogously, `\usemodule[smglom/arithmetics]{realarith}` opens the file `realarith.en.tex` in the `.../smglom/arithmetics/source-folder` and makes its contents available, e.g. `\realdivide` and `\realpower`.

³EdNOTE: somewhere later

`\symref`
`\symname`

The command `\symref{symbolname}{text}` marks the `text` in the second argument as representing the `symbolname` in the first argument – which is why the word “series” is set in boldface. In the pdf, this is all that happens. In the `xhtml` (which we will investigate shortly) however, we will note that the word “series” is now annotated with the full URI of the symbol denoting the *mathematical concept of a series*. In other words, the word is associated with an unambiguous semantics.

Notably, in both cases above (*series* and *converges*) the text that *references* the symbol and the name of the symbol are identical. Since this occurs quite often, the shorthand `\symname{converges}` would have worked as well, where `\symname{foo-bar}` behaves exactly like `\symref{foo-bar}{foo bar}` - i.e. the text is simply the name of the symbol with “-” replaced by a space.

`\importmodule`

If you investigated the contents of the imported modules (`realarith` and `series`) more closely, you’ll note that none of them contain a symbol “converges”. Yet, we can use `\symref` to refer to “converges”. That is because the symbol `converges` is found in `smglom/calculus/source/sequenceConvergence.en.tex`, and `series.en.tex` contains the line `\importmodule{sequenceConvergence}`. The `\importmodule`-statement makes the module referenced available to all documents that include the current module. As such, a “current module” has to exist for `\importmodule` to work, which is why the command is only allowed within a `module-environment`.

TODO explain `xhtml` conversion, MMT compilation (requires an archive...?).

Chapter 3

Using Semantic Macros

TODO

Chapter 4

TeX Archives

4.1 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, TeX uses *archives* that determine the global namespaces for symbols and statements and make it possible for TeX to find content referenced via such URIs.

All TeX archives need to exist in the local MathHub-directory. TeX knows where this folder is via one of three means:

1. If the TeX package is loaded with the option `mathhub=/path/to/mathhub`, then 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 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, 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.

4.2 The Structure of TeX Archives

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

Each such archive needs two subdirectories:

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

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

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

4.3 MANIFEST.MF-Files

The `MANIFEST.MF` in the `META-INF`-directory consists of key-value-pairs, instructing \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/smglob/calculus
narration-base: http://mathhub.info/smglob/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`: 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`.

Chapter 5

Creating New Modules and Symbols

TODO

Example 1

```
\begin{smodule}{assoctest}
\symdef[ args=1 a ]{foo}{\comp{a:}#1\comp{; b:}#2\comp{; c:}#3}{\comp[#1\comp{;}# #1\comp{##2\comp{;#2\comp{}}]
$ \foo {w_1}{w_2}{x,y,z}$
\end{smodule}
```

Module 1: $a : w_1; b : w_2; c : [w_1; x + [w_1; y + z; w_2]; w_2]$

5.1 Advanced Structuring Mechanisms

Given modules:

Example 2

```
\begin{smodule}{magma}
\symdef{universe}{\comp{\mathcal U}}
\symdef[ args=2, op=\circ ]{operation}{#1 \comp{\circ} #2}
\end{smodule}
\begin{smodule}{monoid}
\importmodule{magma}
\symdef{unit}{\comp{e}}
\end{smodule}
\begin{smodule}{group}
\importmodule{monoid}
\symdef[ args=1 ]{inverse}{\comp{-1}}
\end{smodule}
```

Module 2:
Module 3:
Module 4:

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 3

```
\begin{smodule}{ring}
\begin{copymodule}{group}{addition}
\renamedecl[name=universe]{universe}{runiverse}
\renamedecl[name=plus]{operation}{rplus}
\renamedecl[name=zero]{unit}{rzero}
\renamedecl[name=uminus]{inverse}{rminus}
\end{copymodule}
\notation[plus,op=+,prec=60]{rplus}{#1 \comp+ #2}
\notation[zero]{rzero}{\comp0}
\notation[uminus,op=-]{rminus}{\comp- #1}
\begin{copymodule}{monoid}{multiplication}
\assign{universe}{\runiverse}
\renamedecl[name=times]{operation}{rtimes}
\renamedecl[name=one]{unit}{rone}
\end{copymodule}
\notation[cdot,op=\cdot,prec=50]{rtimes}{#1 \comp\cdot #2}
\notation[one]{rone}{\compl}
Test: $\rtimes a{\rplus c{\rtimes de}}$
\end{smodule}
```

Module 5:

Test: $a \circ a$

TODO: explain donotclone

Example 4

```
\begin{smodule}{int}
\symdef{Integers}{\comp{\mathbb Z}}
\symdef[args=2,op=+]{plus}{#1 \comp+ #2}
\symdef{zero}{\comp0}
\symdef[args=1,op=-]{uminus}{\comp-#1}

\begin{interpretmodule}{group}{intisgroup}
\assign{universe}{\Integers}
\assign{operation}{\plus!}
\assign{unit}{\zero}
\assign{inverse}{\uminus!}
\end{interpretmodule}
\end{smodule}
```

Module 6:

5.2 Primitive Symbols (The $\text{\texttt{STEX}}$ Metatheory)

Chapter 6

TeX Statements (Definitions, Theorems, Examples, ...)

Chapter 7

Additional Packages

7.1 Modular Document Structuring

7.2 Slides and Course Notes

7.3 Homework, Problems and Exams

Chapter 8

Stuff

8.1 Modules

`\sTeX`
`\stex`

Both print this \TeX logo.

8.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

To declare a symbol (in a module), we use `\symdecl`, which takes as argument the name of the corresponding semantic macro, e.g. `\symdecl{foo}` introduces the macro `\foo`. Additionally, `\symdecl` takes several options, the most important one being its arity. `foo` as declared above yields a *constant* symbol. To introduce an *operator* which takes arguments, we have to specify which arguments it takes.

Module 7: For example, to introduce binary multiplication, we can do `\symdecl[args=2]{mult}`. We can then supply the semantic macro with arbitrarily many notations, such as `\notation{mult}{#1 #2}`.

Example 5

```
\symdecl[args=2]{mult}
\notation{mult}{#1 #2}
 $\mult{a}{b}$ 
```

ab

Since usually, a freshly introduced symbol also comes with a notation from the start, the `\symdef` command combines `\symdecl` and `\notation`. So instead of the above, we could have also written

```
\symdef[args=2]{mult}{#1 #2}
```

Adding more notations like `\notation[cdot]{mult}{#1 \comp{\cdot} #2}` or `\notation[times]{mult}{#1 \comp{\times} #2}` allows us to write $\mult[cdot]{a}{b}$ and $\mult[times]{a}{b}$:

Example 6

```
\notation[cdot]{mult}{#1 \comp{\cdot} #2}
\notation[times]{mult}{#1 \comp{\times} #2}
 $\mult[cdot]{a}{b}$  and  $\mult[times]{a}{b}$ 
```

$a \cdot b$ and $a \times b$

.

Not using an explicit option with a semantic macro yields the first declared notation, unless changed⁴.

Outside of math mode, or by using the starred variant `\foo*`, allows to provide a custom notation, where notational (or textual) components can be given explicitly in square brackets.

Example 7

```
 $\mult*{a}[\comp{\ast}]{b}$  is the
\mult[\comp{product of}][ $a$ ][ $b$ ]
```

$a * b$ is the product of a and b

.

In custom mode, prefixing an argument with a star will not print that argument, but still export it to OMDoc:

Example 8

```
\mult[\comp{Multiplying}]* $\mult{a}{b}$ [ again by  $b$  yields ...]
```

Multiplying again by b yields...

The syntax `*[int]` allows switching the order of arguments. For example, given a 2-ary semantic macro `\forevery` with exemplary notation `\forall #1. #2`, we can write

Example 9

```
\symdecl[ args=2]{forevery}
\forevery* [2]{The proposition  $P$ }[ $\comp{holds for every}$ ][1]{ $x \in A$ }
```

The proposition P holds for every $x \in A$

⁴EdNOTE: TODO

When using `*[n]`, after reading the provided (n th) argument, the “argument counter” automatically continues where we left off, so the `*[1]` in the above example can be omitted.

For a macro with `arity > 0`, we can refer to the operator *itself* semantically by suffixing the semantic macro with an exclamation point `!` in either text or math mode. For that reason `\notation` (and thus `\symdef`) take an additional optional argument `op=`, which allows to assign a notation for the operator itself. e.g.

Example 10

```
\symdef[ args=2,op={+}]{add}{#1 \comp+ #2}
The operator  $\mathbin{\textcolor{teal}{+}}$  adds two elements, as in  $\mathbin{\textcolor{teal}{+}} ab$ .
```

The operator $+$ adds two elements, as in $a + b$.

`*` is composable with `!` for custom notations, as in:

Example 11

```
\mult![\comp{Multiplication}] (denoted by  $\mathbin{\textcolor{teal}{*}}!$   $\mathbin{\textcolor{teal}{*}} \mathbin{\textcolor{teal}{\cdot}}$ ) is defined by...
```

Multiplication (denoted by \cdot) is defined by...

The macro `\comp` as used everywhere above is responsible for highlighting, linking, and tooltips, and should be wrapped around the notation (or text) components that should be treated accordingly. While it is attractive to just wrap a whole notation, this would also wrap around e.g. the arguments themselves, so instead, the user is tasked with marking the notation components themselves.

The precise behaviour of `\comp` is governed by the macro `\@comp`, which takes two arguments: The tex code of the text (unexpanded) to highlight, and the URI of the current symbol. `\@comp` can be safely redefined to customize the behaviour.

The starred variant `\symdecl*{foo}` does not introduce a semantic macro, but still declares a corresponding symbol. `foo` (like any other symbol, for that matter) can then be accessed via `\STEXsymbol{foo}` or (if `foo` was declared in a module `Foo`) via `\STEXModule{Foo}?{foo}`.

both `\STEXsymbol` and `\STEXModule` take any arbitrary ending segment of a full URI to determine which symbol or module is meant. e.g. `\STEXsymbol{Foo?foo}` is also valid, as are e.g. `\STEXModule{path?Foo}?{foo}` or `\STEXsymbol{path?Foo?foo}`

There’s also a convient shortcut `\symref{?foo}{some text}` for `\STEXsymbol{?foo}!` `[some text]`

Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have “normal” (or more precisely: *i*-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters

representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two i-type arguments.

Besides i-type arguments, \TeX has two other types, which we will discuss now.

The first are *binding* (b-type) arguments, representing variables that are *bound* by the operator. This is the case for example in the above `\forevery`-macro: The first argument is not actually an argument that the `forevery` “function” is “applied” to; rather, the first argument is a new variable (e.g. x) that is *bound* in the subsequent argument. More accurately, the macro should therefore have been implemented thusly:

```
\symdef[args=bi]{forevery}{\forall #1.\; #2}
```

Module 8: b-type arguments are indistinguishable from i-type arguments within \TeX , but are treated very differently in OMDOC and by MMT. More interesting *within* \TeX are a-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

Example 12

```
\symdef[ args=a]{mult}{#1}{##1 \comp\cdot ##2}
$\mult{a,b,c,{d^e},f}$
```

$$a \cdot b \cdot c \cdot d^e \cdot f$$

As the example above shows, notations get a little more complicated for associative arguments. For every a-type argument, the `\notation`-macro takes an additional argument that declares how individual entries in an a-type argument list are aggregated. The first notation argument then describes how the aggregated expression is combined into the full representation.

For a more interesting example, consider a flexary operator for ordered sequences in ordered set, that taking arguments $\{a, b, c\}$ and `\mathbb{R}` prints $a \leq b \leq c \in \mathbb{R}$. This operator takes two arguments (an a-type argument and an i-type argument), aggregates the individuals of the associative argument using `\leq`, and combines the result with `\in` and the second argument thusly:

Example 13

```
\symdef[ args=ai]{numseq}{#1 \comp\in #2}{##1 \comp\leq ##2}
$\numseq{a,b,c}{\mathbb{R}}$
```

$$a \leq b \leq c \in \mathbb{R}$$

Finally, B-type arguments combine the functionalities of a and b, i.e. they represent flexary binding operator arguments.

5 6

⁵EDNOTE: what about e.g. `\int \int \int f dx dy dz`?

⁶EDNOTE: “decompose” a-type arguments into fixed-arity operators?

Precedences

Every notation has an (upwards) *operator precedence* and for each argument a (downwards) *argument precedence* used for automated bracketing. For example, a notation for a binary operator `\foo` could be declared like this:

```
\notation[prec=200;500x600]{foo}{#1 \comp{+} #2}
```

assigning an operator precedence of 200, an argument precedence of 500 for the first argument, and an argument precedence of 600 for the second argument.

\S T E X insert brackets thusly: Upon encountering a semantic macro (such as `\foo`), its operator precedence (e.g. 200) is compared to the current downwards precedence (initially `\neginfprec`). If the operator precedence is *larger* than the current downwards precedence, parentheses are inserted around the semantic macro.

Notations for symbols of arity 0 have a default precedence of `\infprec`, i.e. by default, parentheses are never inserted around constants. Notations for symbols with arity > 0 have a default operator precedence of 0. If no argument precedences are explicitly provided, then by default they are equal to the operator precedence.

Consequently, if some operator A should bind stronger than some operator B , then A ’s operator precedence should be smaller than B ’s argument precedences.

For example:

Module 9:

Example 14

```
\notation[prec=100]{plus}{#1 \comp{+} #2}
\notation[prec=50]{times}{#1 \comp{\cdot} #2}
 $\text{\S } \text{\texttt{plus}}\{a\}\{\text{\texttt{times}}\{b\}\{c\}\}\text{\texttt{\$}}$  and  $\text{\texttt{\$}}\text{\texttt{times}}\{a\}\{\text{\texttt{plus}}\{b\}\{c\}\}\text{\texttt{\$}}$ 
```

```
 $a + b \cdot c$  and  $a \cdot (b + c)$ 
```

8.1.2 Archives and Imports

Namespaces

Ideally, \S T E X 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, \T E X 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 \S T E X 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{module}{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 suffix¹.

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.

Paths in Import-Statements

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 discouraged.

¹which is internally attached to the module name instead, but a user need not worry about that.

Part II

Documentation

Chapter 9

sTeX-Basics

Both the sTeX package and class offer the following package options:

debug ($\langle log\text{-}prefix \rangle *$) Logs debugging information with the given prefixes to the terminal, or all if **all** is given.

lang ($\langle language \rangle *$) Languages to load with the **babel** package.

mathhub ($\langle directory \rangle$) MathHub folder to search for repositories.

sms ($\langle boolean \rangle$) use *persisted* mode (see ???).

image ($\langle boolean \rangle$) passed on to tikzinput.

9.1 Macros and Environments

| | |
|-------------------------|----------------------------|
| $\backslash\text{sTeX}$ | Both print this sTeX logo. |
| $\backslash\text{stex}$ | |

| | |
|-----------------------------------|---|
| $\backslash\text{stex_debug:nn}$ | $\backslash\text{stex_debug:nn} \{ \langle log\text{-}prefix \rangle \} \{ \langle message \rangle \}$ |
|-----------------------------------|---|

Logs $\langle message \rangle$, if the package option **debug** contains $\langle log\text{-}prefix \rangle$.

| | |
|---|--|
| $\backslash\text{stex_add_to_sms:n}$ | Adds the provided code to the <code>.sms</code> -file of the document. |
|---|--|

| | |
|------------------------------------|--|
| $\backslash\text{if@latexml}$ | L ^A T _E X ₂ e and L ^A T _E X ₃ conditionals for L ^A T _E X _M L. |
| $\backslash\text{latexml_if_p:}$ | |
| $\backslash\text{latexml_if:T}$ | |
| $\backslash\text{latexml_if:F}$ | |
| $\backslash\text{latexml_if:TF}$ | |

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

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

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

| | |
|--------------------------------|---|
| <code>stex_annotate_env</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> . |
|--------------------------------|---|

| |
|--|
| <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`.

| | |
|--|--|
| <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>\MSC</code> | <code>\MSC{⟨msc⟩}</code> |
|-------------------|--------------------------|

Designates the *math subject classifier* of the current module / file.

Chapter 10

sTEX-MathHub

Code related to managing and using MathHub repositories, files, paths and related hooks and methods.

10.1 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. |
|--------------------------------|---|

10.1.1 Files, Paths, URIs

| | |
|--|---|
| <code>\stex_path_from_string:Nn</code> | <code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{ \langle string \rangle \}$ |
| <code>\stex_path_from_string:(NV cn cV)</code> | |

turns the $\langle string \rangle$ into a path by splitting it at `/`-characters and stores the result in $\langle path-variable \rangle$. Also applies `\stex_path_canonicalize:N`.

| | |
|--------------------------------------|---|
| <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:NTF</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 (heuristically guessed) full path to the main file, based on the PWD and <code>\jobname</code> . |
| <code>\c_stex_pwd_str</code> | |
| <code>\c_stex_mainfile_seq</code> | |
| <code>\c_stex_mainfile_str</code> | |

`\g_stex_currentfile_seq`

The file being currently processed (respecting `\input` etc.)

Test 1

```
\ExplSyntaxOn
\def\cpath@print#1{
\stex_path_from_string:Nn \l_tmpb_seq { #1 }
\stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
\str_use:N \l_tmpa_str
}
\ExplSyntaxOff
\begin{center}
\begin{tabular}{|l|l|l|}\hline
path & canonicalized path & expected\\\hline
aaa & \cpath@print{aaa} & aaa \\
.././aaa & \cpath@print{.././aaa} & & .././aaa \\
aaa/bbb & \cpath@print{aaa/bbb} & & aaa/bbb \\
aaa/. & \cpath@print{aaa/.} & & \\
.././aaa/bbb & \cpath@print{.././aaa/bbb} & & .././aaa/bbb \\
../aaa/./bbb & \cpath@print{../aaa/./bbb} & & ../bbb \\
../aaa/bbb & \cpath@print{../aaa/bbb} & & ../aaa/bbb \\
aaa/bbb/./ddd & \cpath@print{aaa/bbb/./ddd} & & aaa/ddd \\
aaa/bbb/./ddd & \cpath@print{aaa/bbb/./ddd} & & aaa/bbb/ddd \\
./ & \cpath@print{./} & & \\
aaa/bbb/./.. & \cpath@print{aaa/bbb/./..} & & \\
\end{tabular}
\end{center}
```

| path | canonicalized path | expected |
|---------------|--------------------|--------------|
| aaa | aaa | aaa |
| .././aaa | .././aaa | .././aaa |
| aaa/bbb | aaa/bbb | aaa/bbb |
| aaa/. | | |
| .././aaa/bbb | .././aaa/bbb | .././aaa/bbb |
| ../aaa/./bbb | ../bbb | ../bbb |
| ../aaa/bbb | ../aaa/bbb | ../aaa/bbb |
| aaa/bbb/./ddd | aaa/ddd | aaa/ddd |
| aaa/bbb/./ddd | aaa/bbb/ddd | aaa/bbb/ddd |
| ./ | | |
| aaa/bbb/./.. | | |

10.1.2 MathHub Archives

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

We determine the path to the local MathHub folder via one of three means, in order of precedence:

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

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

`\l_stex_current_repository_prop`

Always points to the *current* MathHub repository (if we currently are in one). Has the fields `id`, `ns` (namespace), `narr` (narrative namespace; currently not in use) and `deps` (dependencies; currently not in use).

| | |
|---|---|
| <hr/> <hr/> <code>\stex_set_current_repository:n</code> | Sets the current repository to the one with the provided ID. calls <code>__stex_mathhub_do_manifest:n</code> , so works whether this repository's MANIFEST.MF-file has already been read or not. |
| <hr/> <hr/> <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. |
| <hr/> <hr/> <code>\stex_in_repository:nn</code> | <code>\stex_in_repository:nn{<repository-name>}{<code>}</code> Change the current repository to <code>{<repository-name>}</code> (or not, if <code>{<repository-name>}</code> is empty), and passes its ID on to <code>{<code>}</code> as #1. Switches back to the previous repository after executing <code>{<code>}</code> . |
| <hr/> <hr/> <code>\mhpath *</code> | <code>\mhpath{<archive-ID>}{<filename>}</code> Expands to the full path of file <code><filename></code> in repository <code><archive-ID></code> . Does not check whether the file or the repository exist. |
| <hr/> <hr/> <code>\inputref</code> <hr/> <code>\inputref:nn</code> | <code>\inputref[<archive-ID>]{<filename>}</code> <code>\inputs</code> the file <code><filename></code> in repository <code><archive-ID></code> . |
| <hr/> <hr/> <code>\libinput</code> | <code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf</code> -archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist. |

Test 2

```

\ExplSyntaxOn
\stex_require_repository:n { Foo/Bar }
id:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {id}\ \
narr:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {narr}\ \
ns:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {ns}\ \
deps:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {deps}\ \
\stex_require_repository:n { Bar/Foo }
\ExplSyntaxOff

```

```

id: Foo/Bar
narr:
ns: http://mathhub.info/tests/Foo/Bar
deps:

```

Chapter 11

sTeX-References

Code related to links and cross-references

11.1 Macros and Environments

Chapter 12

sTeX-Modules

Code related to Modules

12.1 Macros and Environments

`\l_stex_current_module_str`

All information of a module is stored as a property list. `\l_stex_current_module_str` always points to the current module (if existent).

Most importantly, the `content`-field stores all the code to execute on activation; i.e. when this module is being included.

Additionally, it stores:

- The *name* in field `name`,
- the *namespace* in field `ns`,
- this module's *language* in field `lang`,
- if a language module that translates some other modules, the *original* module in field `sig` (for signature),
- the *metatheory* in field `meta`,
- the URIs of all *imported modules* in field `imports`,
- the names of all *declarations* in field `constants`,
- the *file* this module was declared in in field `file`,

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

A property list mapping file paths to the lists of all modules declared therein. `\g_stex_modules_in_file_seq` always points to the current file(-stream - `\inputs` are considered the same file).

```
\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 `content` field of the current module.

```
\stex_add_constant_to_current_module:n
```

Adds the declaration with the provided name to the `constants` field of the current module.

```
\stex_add_import_to_current_module:n
```

Adds the module with the provided full URI to the `imports` field of the current module.

```
\stex_modules_compute_namespace:nN \stex_modules_compute_namespace:nN
{\<namespace>} {\<path>}
```

Computes the namespace for file `<path>` in repository with namespace `<namespace>` as follows:

If the file is `.../source/sub/file.tex` and the namespace `http://some.namespace/foo`, then the namespace of is `http://some.namespace/foo/sub/file`.

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

```
\ExplSyntaxOn
\stex_modules_current_namespace:
Namespace~1:\\ \l_stex_modules_ns_str \\
Faking~a~repository:\\
\stex_set_current_repository:n{Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \testtemp
\edef\testtempb{\detokenize{source}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\edef\testtempb{\detokenize{test}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtemp }
\stex_modules_current_namespace:
Namespace~2:\\ \l_stex_modules_ns_str
\ExplSyntaxOff
```

```

Namespace 1:
file://stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest

```

12.1.1 The module-environment

module `\begin{module}[\langle options \rangle]{\langle name \rangle}`
 Opens a new module with name $\langle name \rangle$.
 TODO document options.

`\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.

`\stex_modules_heading:` Takes care of the module header, if the `showmods` package option is true. This macro can be overridden for customization.

@module `\begin{@module}[\langle options \rangle]{\langle name \rangle}`
 Core functionality of the `module-environment` without a header.

Test 4

```

\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\begin{smodule}{Foo}
Module-path:-
\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { ns }?
\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { name }\\
Language:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { lang }\\
Signature:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { sig }\\
Metatheory:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { meta }\\
\end{smodule}
\ExplSyntaxOff

```

```

Module 10:  Module path: http://mathhub.info/tests/Foo/Bar?Foo
Language:
Signature:
Metatheory:

```

Test 5

```
\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\stex_debug:nn{modules}{Test:-\stex_path_to_string:N \g_stex_currentfile_seq }
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\stex_debug:nn{modules}{Test:-\stex_path_to_string:N \g_stex_currentfile_seq }
\begin{smodule}[title=Foo Bar]{Bar}
Module-path:-
\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { ns }?
\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { name }\\
Language:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { lang }\\
Signature:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { sig }\\
Metatheory:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { meta }\\
\end{smodule}
\ExplSyntaxOff
```

Module 11: FooBar Module path: <http://mathhub.info/tests/Foo/Bar/Foo?Bar>
Language:
Signature:
Metatheory:

`\STEXModule` `\STEXModule {<fragment>}`

Attempts to find a module whose URI ends with `<fragment>` 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 `?<symbolname>`. In the first case, it stores the full URI in `<macro>`; in the second case, it invokes the symbol `<symbolname>` in the selected module.

Test 6

```
\begin{smodule}{STEXModuleTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{STEXModuleTest2}
\importmodule{STEXModuleTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{STEXModuleTest3}
\importmodule{STEXModuleTest2}
\symdecl{foo}
\STEXModule{STEXModuleTest1}!\teststring
\teststring\\
\STEXModule{STEXModuleTest2}!\teststring
\teststring\\
\STEXModule{STEXModuleTest3}!\teststring
\teststring\\
\STEXModule{STEXModuleTest1}?{foo}[\comp{foo1}]\\
\STEXModule{STEXModuleTest2}?{foo}[\comp{foo2}]\\
\STEXModule{STEXModuleTest3}?{foo}[\comp{foo3}]\\
\end{smodule}
```

Module 12:
Module 13:
Module 14: file://stextest?STEXModuleTest1
file://stextest?STEXModuleTest2
file://stextest?STEXModuleTest3
foo1
foo2
foo3

`\stex_activate_module:n`

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

Chapter 13

STEX-Module Inheritance

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

13.1 Macros and Environments

13.1.1 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:

`\g_stex_smsmode_allowedmacros_tl`

Macros that are executed as is; i.e. with the category code scheme used in SMS mode.

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

Importantly, these macros need to call `\stex_smsmode_set_codes:` after reading all arguments. Note, that `\stex_smsmode_set_codes:` takes care of checking whether we are in SMS mode in the first place, so calling this function eagerly is unproblematic.

`\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_set_codes:` should be called at the end of the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called with the SMS mode category code scheme active.

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

Sets the current category code scheme to that of the SMS mode, if SMS mode is currently active and if necessary.

This method should be called at the end of every macro or `\begin` environment code that are allowed in SMS mode.

`\stex_in_smsmode:nn`

`\stex_in_smsmode:nn {<name>} {<code>}`

Executes `<code>` in SMS mode. `<name>` can be arbitrary, but should be distinct, since it allows for nesting `\stex_in_smsmode:nn` without spuriously terminating SMS mode.

Test 7

```
\immediate\openout\testfile=./tests/sometest.tex
\immediate\write\testfile{\detokenize{\this is \a test}^J}
\immediate\write\testfile{\detokenize{this \is a \test}}
\immediate\closeout\testfile
\ExplSyntaxOn
\stex_file_in_smsmode:nn{tests/sometest.tex}{}
\ExplSyntaxOff
```

13.1.2 Imports and Inheritance

`\importmodule`

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

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

Test 8

```
\begin{smodule}{Foo}
\symdecl[name=foo, args=3]{bar}
\symdecl[ args=bai]{foobar}
Meaning:-\present\bar\
\end{smodule}
Meaning:-\present\bar\
\begin{smodule}{Importtest}
\importmodule{Foo}
Meaning:-\present\bar\
\end{smodule}
\begin{smodule}{Importtest2}
\importmodule{Importtest}
Meaning:-\present\bar\
\end{smodule}
```

```
Module 15:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
              Meaning: >macro:->\protect \bar <
Module 16:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
Module 17:    Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
```

`\usemodule`

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

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

Test 9

```

\begin{smodule}{UseTest1}
\symdecl{foo}
\end{smodule}
\begin{smodule}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:- \present\foo\
\end{smodule}
\begin{smodule}{UseTest3}
\importmodule{UseTest2}
Meaning:- \present\foo\
Meaning:- \present\bar\

All modules: \ExplSyntaxOn
\seq_use:Nn \l_stex_all_modules_seq {,-} \
All-symbols:-
\seq_use:Nn \l_stex_all_symbols_seq {,-}
\ExplSyntaxOff
\end{smodule}

```

Module 18:
Module 19: Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?UseTest1?foo}<
Module 20: Meaning: >undefined<
Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?UseTest2?bar}<

All modules: <http://mathhub.info/sTeX?Metatheory,file://stextest?UseTest3,file://stextest?UseTest2>
All symbols: <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?collection>, <http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?sequence-index>, <http://mathhub.info/sTeX?Metatheory?seqtype>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?mathematical-structure>, <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?dummyvar>, <http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?collection>, <http://mathhub.info/sTeX?Metatheory?seqtype>, <http://mathhub.info/sTeX?Metatheory?sequence-index>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?mathematical-structure>, <file://stextest?UseTest2?bar>

Test 10

```

Circular dependencies:
\begin{smodule}{CircDep1}
\importmodule[Foo/Bar]{circular1?Circular1}
\importmodule[Bar/Foo]{circular2?Circular2}
\present\fooA\
\present\fooB\
\end{smodule}

```

Circular dependencies:
Module 21: >macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<
>macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB}<

| | |
|---|--|
| <hr/> <hr/> | <hr/> |
| <code>\stex_import_module_uri:nn</code> | <code>\stex_import_module_uri:nn {\langle archive-ID \rangle} {\langle module-path \rangle}</code> |
| | <p>Determines the URI of a module by splitting $\langle module-path \rangle$ into $\langle path \rangle ? \langle name \rangle$. If $\langle module-path \rangle$ does <i>not</i> contain a ?-character, we consider it to be the $\langle name \rangle$, and $\langle path \rangle$ to be empty.</p> <p>If $\langle archive-ID \rangle$ is empty, it is automatically set to the ID of the current archive (if one exists).</p> <ol style="list-style-type: none"> 1. If $\langle archive-ID \rangle$ is empty: <ol style="list-style-type: none"> (a) If $\langle path \rangle$ is empty, then $\langle name \rangle$ must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code>, or a file with name $\langle name \rangle . \langle lang \rangle . \text{tex}$ must exist in the same folder, containing a module $\langle name \rangle$. That module should have the same namespace as the current one. (b) If $\langle path \rangle$ is not empty, it must point to the relative path of the containing file as well as the namespace. 2. Otherwise: <ol style="list-style-type: none"> (a) If $\langle path \rangle$ is empty, then $\langle name \rangle$ must have been declared earlier in the same file and retrievable from <code>\g_stex_modules_in_file_seq</code>, or a file with name $\langle name \rangle . \langle lang \rangle . \text{tex}$ must exist in the top <code>source</code> folder of the archive, containing a module $\langle name \rangle$. That module should lie directly in the namespace of the archive. (b) If $\langle path \rangle$ is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive. If a module by that namespace exists, it is returned. Otherwise, we call <code>\stex_require_module:nn</code> on the <code>source</code> directory of the archive to find the file. |

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

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

Finally, activates that module by executing its `content`-field.

Chapter 14

TeX-Symbols

Code related to symbol declarations and notations

14.1 Macros and Environments

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

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

- **name**: An (OMDOC) name. By default equal to $\langle\mathit{macroname}\rangle$.
- **type**: An (ideally semantic) term. Not used by TeX, 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. $\backslash\mathrm{symdecl}[\mathit{args=ii}]\{\mathit{plus}\}$ allows for $\backslash\mathit{plus}\{2\}\{2\}$.
 - a an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. $\backslash\mathrm{symdecl}[\mathit{args=a}]\{\mathit{plus}\}$ allows for $\backslash\mathit{plus}\{2,2,2\}$.
 - b a *variable* argument. Is treated by TeX 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. $\backslash\mathrm{symdecl}[\mathit{args=bi}]\{\mathit{forall}\}$ allows for $\backslash\mathit{forall}\{x\in\mathit{Nat}\}\{x\geq 0\}$.

`\stex_symdecl_do:n`

Implements the core functionality of `\symdecl`, and is called by `\symdecl` and `\symdef`.

Ultimately stores the symbol $\langle URI \rangle$ in the property list `\l_stex_symdecl_⟨URI⟩_prop` with fields:

- `name` (string),
- `module` (string),
- `notations` (sequence of strings; initially empty),
- `local` (boolean),
- `type` (token list),
- `args` (string of `is`, `as` and `bs`),
- `arity` (integer string),
- `assocs` (integer string; number of associative arguments),

Test 11

```
\begin{smodule}{SymdeclTest}
\symdecl[name=foo, args=3]{bar}
\symdecl[name=foobar, args=iab]{bari}
\symdecl[def=\bar* abc]{bardef}
\ExplSyntaxOn
Meaning:-\present\bar\
\stex_get_symbol:n { bar }
Result:-\l_stex_get_symbol_uri_str\
Meaning:-\present\bardef\
\ExplSyntaxOff
\end{smodule}
```

```
Module 22:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?foo}<
Result: file://stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?SymdeclTest?bardef}<
```

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...

`\notation`

`\notation[⟨args⟩]{⟨symbol⟩}{⟨notations+⟩}`

Introduces a new notation for $\langle symbol \rangle$, see `\stex_notation_do:nn`

`\stex_notation_do:nn` `\stex_notation_do:nn{<URI>}{<notations+>}`

Implements the core functionality of `\notation`, and is called by `\notation` and `\symdef`.

Ultimately stores the notation in the property list `\g_stex_notation_<URI>#<variant>#<lang>_prop` with fields:

- symbol (URI string),
- language (string),
- variant (string),
- opprec (integer string),
- argprecs (sequence of integer strings)

Test 12

```
\begin{smodule}{NotationTest}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1} ^ {#2} _ {#3} \comp\rangle }
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^ {#3} \comp\rangle }{ {#1}_{\comp
```

Module 23:

`\symdef` `\symdef[<args>]{<symbol>}{<notations+>}`

Combines `\symdecl` and `\notation` by introducing a new symbol and assigning a new notation for it.

Test 13

```
\begin{smodule}{SymdefTest}
\symdef[args=a, prec=50]{plus}{ #1 }{##1 \comp+ ##2}
$\plus{a,b,c}$
\end{smodule}
```

Module 24: $a + b + c$

Chapter 15

STEX-Terms

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

15.1 Macros and Environments

| | |
|--|---|
| <hr/> <hr/> <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> |
| <hr/> <hr/> <code>\symref</code> | <code>\symref{<symbol>}{<text>}</code> shortcut for <code>\STEXsymbol{<symbol>}! [<text>]</code> |
| <hr/> <hr/> <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 <code>!</code> , it will invoke the symbol <i>itself</i> rather than its application (and continue to <code>\stex_term_custom:nn</code>), i.e. it allows to refer to <code>\plus!</code> [addition] as an operation, rather than <code>\plus[addition of]{some}{terms}</code> . |
| <hr/> <hr/> <code>_stex_term_math_oms:nnnn</code> <code>_stex_term_math_oma:nnnn</code> <code>_stex_term_math_omb:nnnn</code> | <code><URI><fragment><precedence><body></code> Annotates <code><body></code> as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol <code><URI></code> , generated by the specific notation <code><fragment></code> with (upwards) operator precedence <code><precedence></code> . Inserts parentheses according to the current downwards precedence and operator precedence. |
| <hr/> <hr/> <code>_stex_term_math_arg:nnn</code> | <code>\stex_term_arg:nnn<int><prec><body></code> Annotates <code><body></code> as the <code><int></code> th argument of the current OMA or OMBIND, with (downwards) argument precedence <code><prec></code> . |
| <hr/> <hr/> <code>_stex_term_math_assoc_arg:nnnn</code> | <code>\stex_term_arg:nnn<int><prec><notation><body></code> Annotates <code><body></code> as the <code><int></code> th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence <code><prec></code> and associative notation <code><notation></code> . |

`\infprec`
`\neginfprec`

Maximal and minimal notation precedences.

`\dobrackets`

`\dobrackets {⟨body⟩}`

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 `\withbrackets`.

`\withbrackets`

`\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}`

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 `\left` and `\right` in display-mode.

Test 14

```
\begin{smodule}{MathTest1}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1} ^ {#2} _ {#3} \comp\rangle }
$\bar{abc}$ and $\bar{foo} \ abc$.
\end{smodule}
```

Module 25: $\langle a^b_c \rangle$ and $\langle a^b_c \rangle$.

Test 15

```
\begin{smodule}{MathTest2}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^ {#3} \comp\rangle }{ {#1}_{\comp\langle #1 \comp\mid [ #2 ] ^ {#3} \comp\rangle } }
$\foobar a{b,c,d,e,f}g$ and $\foobar[foo] a{b,c}g$ and $\foobar abc$

\symdecl[ args=a]{ plus }
\symdecl[ args=a]{ mult }
\notation[prec=50]{ plus }{#1}{##1 \comp+ ##2}
\notation[prec=100]{ mult }{#1}{##1 \comp\cdot ##2}
$\plus{a,\mult{b,c}}$ and $\mult{a,\plus{\frac{ab}{c}}}$
$[\plus{a,\mult{b,c}}]\text{ and }[\mult{a,\plus{\frac{ab}{c}}}]$
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[{$\displaystyle
\mult{a,\plus{\frac{ab}{c}}}$}
\end{smodule}
```

Module 26: $\langle a \mid [b;c,d,e,f]^g \rangle$ and $\langle a \mid [b;c]^g \rangle$ and $\langle a \mid [b]^c \rangle$

$$a + (b \cdot c) \text{ and } a \cdot \frac{a}{b} + \frac{a}{c}$$

$$a + (b \cdot c) \text{ and } a \cdot \frac{a}{b} + \frac{a}{c}$$

$$a + (b \cdot c) \text{ and } a \cdot \frac{a}{b} + \frac{a}{c}$$

`\stex_term_custom:nn`

`\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}`

Implements custom one-time notation. Invoked by `\stex_invoke_symbol:n` in text mode, or if followed by `*` in math mode, or whenever followed by `!`.

Test 16

```
\begin{smodule}{TextTest}
\importmodule{Foo}

\bar[some ]a[ and some ]b[ and also some ]c[ here].

$\bar*[\text{some }]a[\text{ and some }]b[\text{ and also some }]c[\text{ here}]]$.

$\bar!![\mathtt{bar}]$

\bar*{a}*{b}[or just some ]c

\bar![bar]

\bar[or first ]*[2]{b}[ , then ]*[3]{c}[ , and finally ]a

\end{smodule}
```

Module 27:

some a and some b and also some c here.
some a and some b and also some c here.
bar
or just some c
bar
or first b, then c, and finally a

`\stex_highlight_term:nn` `\stex_highlight_term:nn{<URI>}{<args>}`

Establishes a context for `\comp`. Stores the URI in a variable so that `\comp` knows which symbol governs the current notation.

`\comp` `\comp{<args>}`

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

Marks `<args>` as a notation component of the current symbol for highlighting, linking, etc.

The precise behavior is governed by `\@comp`, which takes as additional argument the URI of the current symbol. By default, `\@comp` adds the URI as a PDF tooltip and colors the highlighted part in blue.

`\@defemph` behaves like `\@comp`, and can be similarly redefined, but marks an expression as *definiendum* (used by `\definiendum`)

`\STEXinvisible` 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.

`\ellipses` TODO

Chapter 16

TeX-Structural Features

Code related to structural features

16.1 Macros and Environments

16.1.1 Structures

`mathstructure` TODO

Chapter 17

sTeX-Statements

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

17.1 Macros and Environments

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

Chapter 18

sTeX-Proofs: Structural Markup for Proofs

The `sproof` package is part of the sTeX collection, a version of T_EX/L^AT_EX that allows to markup T_EX/L^AT_EX documents semantically without leaving the document format, essentially turning T_EX/L^AT_EX into a document format for mathematical knowledge management (MKM).

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

Contents

18.1 Introduction

The `sproof` (semantic proofs) package supplies macros and environment that allow to annotate the structure of mathematical proofs in \LaTeX files. This structure can be used by MKM systems for added-value services, either directly from the \LaTeX sources, or after translation. Even though it is part of the \LaTeX collection, it can be used independently, like its sister package `statements`.

\LaTeX is a version of $\text{\TeX}/\text{\LaTeX}$ that allows to markup $\text{\TeX}/\text{\LaTeX}$ documents semantically without leaving the document format, essentially turning $\text{\TeX}/\text{\LaTeX}$ into a document format for mathematical knowledge management (MKM).

```
\begin{sproof}[id=simple-proof,for=sum-over-odds]
  {We prove that  $\sum_{i=1}^n (2i-1) = n^2$  by induction over  $n$ }
  \begin{spfcase}{For the induction we have to consider the following cases:}
    \begin{spfcase}{ $n=1$ }
      \begin{spfstep}[display=flow] then we compute  $1=1^2$ \end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n=2$ }
      \begin{sproofcomment}[display=flow]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[display=flow] We compute  $1+3=2^2=4$ .\end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n>1$ }
      \begin{spfstep}[type=assumption,id=ind-hyp]
        Now, we assume that the assertion is true for a certain  $k \geq 1$ ,
        i.e.  $\sum_{i=1}^k (2i-1) = k^2$ $.
      \end{spfstep}
      \begin{sproofcomment}
        We have to show that we can derive the assertion for  $n=k+1$  from
        this assumption, i.e.  $\sum_{i=1}^{k+1} (2i-1) = (k+1)^2$ $.
      \end{sproofcomment}
      \begin{spfstep}
        We obtain  $\sum_{i=1}^{k+1} (2i-1) = \sum_{i=1}^k (2i-1) + 2(k+1) - 1$ 
        \begin{justification}[method=arith:split-sum]
          by splitting the sum.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}
        Thus we have  $\sum_{i=1}^{k+1} (2i-1) = k^2 + 2k + 1$ 
        \begin{justification}[method=fertilize]
          by inductive hypothesis.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}[type=conclusion]
        We can \begin{justification}[method=simplify]simplify\end{justification}
        the right-hand side to  $(k+1)^2$ , which proves the assertion.
      \end{spfstep}
    \end{spfcase}
    \begin{spfstep}[type=conclusion]
      We have considered all the cases, so we have proven the assertion.
    \end{spfstep}
  \end{spfcase}
\end{sproof}
```

Example 1: A very explicit proof, marked up semantically

We will go over the general intuition by way of our running example (see Figure 1 for the source and Figure 2 for the formatted result).⁷

⁷EDNOTE: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.

18.2 The User Interface

18.2.1 Package Options

`showmeta` The `sproof` package takes a single option: `showmeta`. If this is set, then the metadata keys are shown (see [Kohlhase:metakeys] for details and customization options).

18.2.2 Proofs and Proof steps

`sproof` The `proof` environment is the main container for proofs. It takes an optional `KeyVal` argument that allows to specify the `id` (identifier) and `for` (for which assertion is this a proof) keys. The regular argument of the `proof` environment contains an introductory comment, that may be used to announce the proof style. The `proof` environment contains a sequence of `\step`, `proofcomment`, and `pfcases` environments that are used to markup the proof steps. The `proof` environment has a variant `Proof`, which does not use the proof end marker. This is convenient, if a proof ends in a case distinction, which brings it's own proof end marker with it. The `Proof` environment is a variant of `proof` that does not mark the end of a proof with a little box; presumably, since one of the subproofs already has one and then a box supplied by the outer proof would generate an otherwise empty line. 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 `KeyVal` argument as `sproof` and another one: a natural language text that sketches the proof.

`spfstep` Regular proof steps are marked up with the `step` environment, 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.

Note that both `\premise` and `\justarg` can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.

18.2.3 Justifications

`justification` This evidence is marked up with the `justification` environment in the `sproof` package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence. The environment takes an optional `KeyVal` argument, which can have the `method` key, whose value is the name of a proof method (this will only need to mean something to the application that consumes the semantic annotations). Furthermore, the justification can contain “premises” (specifications to assertions that were used justify the step) and “arguments” (other information taken into account by the proof method).

`\premise` The `\premise` macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the example in Figure 1 we have used the `\premise` macro to identify the inductive hypothesis.

`\justarg` The `\justarg` macro is very similar to `\premise` with the difference that it is used to mark up arguments to the proof method. Therefore the content of the first argument is interpreted as a mathematical object rather than as an identifier as in the case of `\premise`. In our example, we specified that the simplification should take place on the right hand side of the equation. Other examples include proof methods that instantiate. Here we would indicate the substituted object in a `\justarg` macro.

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

P.1 For the induction we have to consider the following cases:

P.1.1 $n = 1$: then we compute $1 = 1^2$ □

P.1.1 $n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$ □

P.1.1 $n > 1$:

P.1.1.1 Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^k (2i - 1) = k^2$.

P.1.1.1 We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.

P.1.1.1 We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum

P.1.1.1 Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.

P.1.1.1 We can simplify the right-hand side to $(k + 1)^2$, which proves the assertion. □

P.1.1 We have considered all the cases, so we have proven the assertion. □

Example 2: The formatted result of the proof in Figure 1

18.2.4 Proof Structure

| | |
|-----------------------|---|
| subproof | The pfcases 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. |
| method | |
| spfcases | The pfcases environment is used to mark up a proof by cases. Technically it is a variant of the subproof where the method is by-cases . Its contents are spfcases environments that mark up the cases one by one. |
| spfcases | The content of a pfcases environment are a sequence of case proofs marked up in the pfcases environment, which takes an optional KeyVal argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a pfcases environment is the same as that of a proof , i.e. steps , proofcomments , and pfcases environments. \spfcasesketch is a variant of the spfcases environment that takes the same arguments, but instead of the spfsteps in the body uses a third argument for a proof sketch. |
| \spfcasesketch | |
| sproofcomment | The proofcomment environment is much like a step , only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to do, or what we have achieved so far. As such, it cannot be the target of a \premise . |

18.2.5 Proof End Markers

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:

`\sproofend` The `sproof` package provides the `\sproofend` macro for this. 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{}`.

18.2.6 Configuration of the Presentation

Finally, we provide configuration hooks in Figure 1 for the keywords in proofs. These are mainly intended for package authors building on `statements`, e.g. for multi-language support.⁸ The proof step labels can be customized via the `\pstlabelstyle` macro:

| Environment | configuration macro | value |
|-------------|---------------------|-------------|
| sproof | \spf@proof@kw | Proof |
| sketchproof | \spf@sketchproof@kw | ProofSketch |

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

`\pstlabelstyle{style}` sets the style; see Figure 2 for an overview of styles. Package writers can add additional styles by adding a macro `\pstmake@label@style` that takes two arguments: a comma-separated list of ordinals that make up the prefix and the current ordinal. Note that comma-separated lists can be conveniently iterated over by the `\LaTeX \@for...:=...\do{...}` macro; see Figure 2 for examples.

| style | example | configuration macro |
|--------|----------|---|
| long | 0.8.1.5 | <code>\def\pst@make@label@long#1#2{\@for\@I:=#1\do{\@I.}#2}</code> |
| angles | $\ggg 5$ | <code>\def\pst@make@label@angles#1#2{\ensurermath{\@for\@I:=#1\do{\@I\angle}}#2}</code> |
| short | 5 | <code>\def\pst@make@label@short#1#2{#2}</code> |
| empty | | <code>\def\pst@make@label@empty#1#2{}</code> |

Figure 2: Configuration Proof Step Label Styles

18.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `sTeX` issue tracker at [\[sTeX\]](#).

⁸EDNOTE: we might want to develop an extension `sproof-babel` in the future.

1. The numbering scheme of proofs cannot be changed. It is more geared for teaching proof structures (the author's main use case) and not for writing papers. reported by Tobias Pfeiffer (fixed)
2. currently proof steps are formatted by the `LATEX description` environment. We would like to configure this, e.g. to use the `inparaenum` environment for more condensed proofs. I am just not sure what the best user interface would be I can imagine redefining an internal environment `spf@proofstep@list` or adding a key `prooflistenv` to the `proof` environment that allows to specify the environment directly. Maybe we should do both.

Chapter 19

sTeX-Metatheory

The default meta theory for an sTeX module. Contains 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).

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.

19.1 Symbols

Part III
Extensions

Chapter 20

Tikzinput

20.1 Macros and Environments

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhpath

Chapter 21

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

The `document-structure` package is part of the $\S\TeX$ collection, a version of \TeX / \LaTeX that allows to markup \TeX / \LaTeX documents semantically without leaving the document format, essentially turning \TeX / \LaTeX into a document format for mathematical knowledge management (MKM).

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

21.1 Introduction

$\S\TeX$ is a version of \TeX / \LaTeX that allows to markup \TeX / \LaTeX documents semantically without leaving the document format, essentially turning \TeX / \LaTeX into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDOC format [Koh06]

The `document-structure` package supplies macros and environments that allow to label document fragments and to reference them later in the same document or in other documents. In essence, this enhances the document-as-trees model to documents-as-directed-acyclic-graphs (DAG) model. This structure can be used by MKM systems for added-value services, either directly from the $\S\TeX$ sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the $\S\TeX$ 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.⁹

21.2 The User Interface

The `document-structure` package generates two files: `document-structure.cls`, and `document-structure.sty`. The OMDoc class is a minimally changed variant of the standard `article` class that includes the functionality provided by `document-structure.sty`. The rest of the documentation pertains to the functionality introduced by `document-structure.sty`.

21.2.1 Package and Class Options

The `document-structure` class accept 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> |
| <code>showignores</code> | show the the contents of the <code>ignore</code> environment after all |
| <code>showmeta</code> | show the metadata; see <code>metakeys.sty</code> |
| <code>showmods</code> | show modules; see <code>modules.sty</code> |
| <code>extrefs</code> | allow external references; see <code>sref.sty</code> |
| <code>defindex</code> | index definienda; see <code>statements.sty</code> |
| <code>minimal</code> | for testing; do not load any \TeX packages |

The `document-structure` package accepts the same except the first two.

21.2.2 Document Structure

| | |
|----------------------------|---|
| <code>document</code> | The top-level <code>document</code> environment can be given key/value information by the |
| <code>\documentkeys</code> | <code>\documentkeys</code> macro in the preamble ² . This can be used to give metadata about the |
| <code>id</code> | document. For the moment only the <code>id</code> key is used to give an identifier to the <code>omdoc</code> |
| <code>omgroup</code> | element resulting from the L ^A T _E XML transformation. |
| | The structure of the document is given by the <code>omgroup</code> environment just like in OM- |
| | DOC. In the L ^A T _E X route, the <code>omgroup</code> environment is flexibly mapped to sectioning com- |
| | mands, inducing the proper sectioning level from the nesting of <code>omgroup</code> environments. |
| | Correspondingly, the <code>omgroup</code> environment takes an optional key/value argument for |
| | metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The op- |
| <code>id</code> | tional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code> |
| <code>creators</code> | for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The |
| <code>contributors</code> | <code>short</code> allows to give a short title for the generated section. If the title contains semantic |
| <code>short</code> | macros, they need to be protected by <code>\protect</code> , and we need to give the <code>loadmodules</code> |
| <code>loadmodules</code> | key it needs no value. For instance we would have |

```

\begin{smodule}{foo}
\symdef{bar}{B^a_r}
...
\begin{omgroup}[id=sec.bardderiv,loadmodules]{Introducing $\protect\bar$ Derivations}

```

⁹EdNOTE: integrate with latexml's XMRef in the Math mode.

²We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

`blindomgroup`

\TeX automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a `subsection*` as an introduction for a chapter). Therefore the `document-structure` package provides a variant `blindomgroup` 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 `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 3 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` 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. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

```
\begin{document}
\begin{blindomgroup}
\begin{blindomgroup}
\begin{frontmatter}
\maketitle\newpage
\begin{omgroup}[display=flow]{Preface}
... <<preface>> ...
\end{omgroup}
\clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
\end{frontmatter}
\end{blindomgroup}
... <<introductory remarks>> ...
\end{blindomgroup}
\begin{omgroup}{Introduction}
... <<intro>> ...
\end{omgroup}
... <<more chapters>> ...
\bibliographystyle{alpha}\bibliography{kwarc}
\end{document}
```

Example 3: A typical Document Structure of a Book

`\skipomgroup`

The `\skipomgroup` “skips an `omgroup`”, 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 `\skipomgroup`.

`\currentsectionlevel`
`\CurrentSectionLevel`

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

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

21.2.3 Ignoring Inputs

`ignore` The `ignore` environment can be used for hiding text parts from the document structure.
`showignores` The body of the environment is not PDF or DVI output unless the `showignores` option is given to the `document-structure` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

editing One may want to hide unfinished or obsolete parts of a document

narrative/content markup In \TeX we mark up narrative-structured documents. In the generated OMDoc documents we want to be able to cache content objects that are not directly visible. For instance in the `statements` package [Koh20d] we use the `\inlinedef` macro to mark up phrase-level definitions, which verbalize more formal definitions. The latter can be hidden by an `ignore` and referenced by the `verbalizes` key in `\inlinedef`.

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

`\afterprematurestop` `\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].

21.2.4 Structure Sharing

`\STRlabel` The `\STRlabel` macro takes two arguments: a label and the content and stores the content for later use by `\STRcopy[\langle URL \rangle]{\langle label \rangle}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `\langle URL \rangle` that lets L^AT_EXML generate the correct reference.
`\STRcopy`
`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in L^AT_EX. This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.¹⁰

21.2.5 Global Variables

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. `\setSGvar{\langle vname \rangle}{\langle text \rangle}` to set the global variable `\langle vname \rangle` to `\langle text \rangle` and `\useSGvar{\langle vname \rangle}` to reference it.
`\setSGvar`
`\useSGvar`
`\ifSGvar` With `\ifSGvar` we can test for the contents of a global variable: the macro call

¹⁰EdNOTE: document LMID und LMXRef here if we decide to keep them.

`\ifSGvar{⟨vname⟩}{⟨val⟩}{⟨ctext⟩}` tests the content of the global variable `⟨vname⟩`, only if (after expansion) it is equal to `⟨val⟩`, the conditional text `⟨ctext⟩` is formatted.

21.2.6 Colors

For convenience, the `document-structure` package defines a couple of color macros for the `color` package: For instance `\blue` abbreviates `\textcolor{blue}`, so that `\blue{⟨something⟩}` writes `⟨something⟩` in blue. The macros `\red`, `\green`, `\cyan`, `\magenta`, `\brown`, `\yellow`, `\orange`, `\gray`, and finally `\black` are analogous.

21.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `TeX` GitHub repository [\[sTeX\]](#).

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made.

Chapter 22

NotesSlides – Slides and Course Notes

We present a document class from which we can generate both course slides and course notes in a transparent way.

22.1 Introduction

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

22.2 The User Interface

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.

22.2.1 Package Options

The `notesslides` class takes a variety of class options:¹¹

- | | |
|---------------------|---|
| <code>slides</code> | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see |
| <code>notes</code> | Section 22.2.2). |

| | |
|--|---|
| <code>sectocframes</code> | <ul style="list-style-type: none"> If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated. |
| <code>showmeta</code> | <ul style="list-style-type: none"> <code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options). |
| <code>frameimages</code> <code>fiboxed</code> | <ul style="list-style-type: none"> If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 22.2.4). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box. |
| <code>topsect</code> | <ul style="list-style-type: none"> <code>topsect=<sect></code> can be used to specify the top-level sectioning level; the default for <code><sect></code> is <code>section</code>. |

22.2.2 Notes and Slides

`frame` Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details.
`note` The `notesslides` class adds the `note` environment for encapsulating the course note fragments.⁴

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

```
\ifnotes\maketitle\else
\frame[noframenumbering]\maketitle\fi

\begin{note}
  We start this course with ...
\end{note}

\begin{frame}
  \frametitle{The first slide}
  ...
\end{frame}
\begin{note}
  ... and more explanatory text
\end{note}

\begin{frame}
  \frametitle{The second slide}
  ...
\end{frame}
...
```

Example 4: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 4.

`\ifnotes` Note the use of the `\ifnotes` conditional, which allows different treatment between

¹¹EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

⁴MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive L^AT_EX trickery. Hints to the author are welcome.

`notes` and `slides` mode – manually setting `\notesttrue` or `\notesfalse` is strongly discouraged however.

⚠: We need to give the title frame the `noframenumbers` 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).

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 from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

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

22.2.3 Header and Footer Lines of the Slides

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

The default footer line of the `notesslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder . By default it is *Michael Kohlhase* in the `notesslides` package since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name. 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.

22.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add \TeX notes. 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.¹²

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

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

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

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


¹²EdNOTE: MK: the `hyperref` link does not seem to work yet. I wonder why but do not have the time to fix it.

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

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

22.2.5 Colors and Highlighting

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

22.2.6 Front Matter, Titles, etc.

22.2.7 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:

```
\excursion{founif}{../ex/founif}{We will cover first-order unification in}
...
\begin{appendix}\printexcursions\end{appendix}
```

```
\excursion      The \excursion{<ref>}{<path>}{<text>} is syntactic sugar for
\activateexcursion
\begin{nparagraph}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nparagraph}
```

```
\activateexcursion      where \activateexcursion{<path>} augments the \printexcursions macro by a
\printexcursions        call \inputref{<path>}. In this way, the3 \printexcursions macro (usually in the
                        appendix) will collect up all excursions that are specified in the main text.
```

Sometimes, we want to reference – in an excursion – part of another. We can use

```
\excursionref \excursionref{<label>} for that.
```

Finally, we usually want to put the excursions into an `omgroup` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro:

```
\excursiongroup \excursiongroup[id=<id>,intro=<path>] is equivalent to
```

```
\begin{note}
\begin{omgroup}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{omgroup}
\end{note}
```

22.2.8 Miscellaneous

22.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the \TeX GitHub repository [[sTeX](#)].

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` 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 `omdoc` package.

Chapter 23

problem.sty: An Infrastructure for formatting Problems

The `problem` package supplies an infrastructure that allows specify problems and to reuse them efficiently in multiple environments.

23.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions⁵. 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.

23.2 The User Interface

23.2.1 Package Options

| | |
|------------------------|---|
| <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. |
| <code>mh</code> | The <code>mh</code> option turns on MathHub support; see [<code>Kohlhase:mss</code>]. |
| <code>showmeta</code> | Finally, if the <code>showmeta</code> is set, then the metadata keys are shown (see [<code>Kohlhase:metakeys</code>] |
| | for details and customization options). |

⁵for the moment multiple choice problems are not supported, but may well be in a future version

23.2.2 Problems and Solutions

problem The main environment provided by the **problem** package 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. For an example of a marked up problem see Figure 5 and the resulting markup see Figure 6.

```
\usepackage[solutions,hints,pts,min]{problem}
\begin{document}
  \begin{sproblem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
    How many Elephants can you fit into a Volkswagen beetle?
  \begin{hint}
    Think positively, this is simple!
  \end{hint}
  \begin{exnote}
    Justify your answer
  \end{exnote}
  \begin{solution}[for=elephants,height=3cm]
    Four, two in the front seats, and two in the back.
  \begin{gnote}
    if they do not give the justification deduct 5 pts
  \end{gnote}
  \end{solution}
  \end{sproblem}
\end{document}
```

Example 5: A marked up Problem

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

```
Problem 0.1 (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.


---


```

Example 6: The Formatted Problem from Figure 5

hint The **hint** and **exnote** environments can be used in a **problem** environment to give hints and to make notes that elaborate certain aspects of the problem.

exnote

gnote The **gnote** (grading notes) environment can be used to document situations that

may arise in grading.

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

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.

23.2.3 Multiple Choice Blocks

Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc[⟨keyvals⟩]{⟨text⟩}` macro, which takes an optional key/value argument `⟨keyvals⟩` for choice metadata and a required argument `⟨text⟩` for the proposed answer text. The following keys are supported

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

See Figure ?? for an example

23.2.4 Including Problems

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.

23.2.5 Reporting Metadata

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.

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` package options are set. This allows to give students hints about the estimated time and the points to be awarded.

23.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `STEXGitHub` repository [[sTeX](#)].

1. none reported yet

```

\begin{sproblem}[title=Functions]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def}
    \mcc[F,feedback=that is for C and C++){function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Noooooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}

```

Problem 0.2 (Functions)

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

1. def
2. function
3. fun
4. public static void

Problem 0.3 (Functions)

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

1. def
!
2. function
that is for C and C++
3. fun
that is for Standard ML
4. public static void
that is for Java

Example 7: A Problem with a multiple choice block

Chapter 24

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

The `hwexam` package and class allows individual course assignment sheets and compound assignment documents using problem files marked up with the `problem` package.

Contents

24.1 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 [Kohlhase:problem]. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

24.2 The User Interface

24.2.1 Package and Class Options

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

`showmeta` If the `showmeta` option is set, then the metadata keys are shown (see [Kohlhase:metakeys] for details and customization options).

The `hwexam` class additionally accepts the options `report`, `book`, `chapter`, `part`, and `showignores`, of the `omdoc` package [Kohlhase:smomdl] on which it is based and passes them on to that. For the `extrefs` option see [Kohlhase:sref].

24.2.2 Assignments

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

24.2.3 Typesetting Exams

`multiple` Furthermore, the `hwexam` package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

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

`\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` Finally, the `\testheading` takes an optional keyword argument where the keys
`duration` `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.
`min`
`reqpts`

24.2.4 Including Assignments

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

24.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `STEX`GitHub repository [\[sTeX\]](#).

1. none reported yet.

Part IV

Implementation

Chapter 25

\TeX -Basics Implementation

25.1 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}{2021/08/01}{1.9}{bla}
7 \LoadClass[border=1px,varwidth]{standalone}
8 \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 </cls>
```

25.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21 \RequirePackage{expl-keystr-compatible}
22
23 %\RequirePackage{morewrites}
24 %\RequirePackage{amsmath}
25
```

Package options:

```

26 \keys_define:nn { stex } {
27   debug      .clist_set:N = \c_stex_debug_clist ,
28   lang       .clist_set:N = \c_stex_languages_clist ,
29   mathhub    .tl_set_x:N = \mathhub ,
30   sms        .bool_set:N = \c_stex_persist_mode_bool ,
31   image      .bool_set:N = \c_tikzinput_image_bool ,
32   unknown    .code:n      = {}
33 }
34 \ProcessKeysOptions { stex }

\stex The sTeXlogo:
\TeX
35 \protected\def\stex{%
36   \@ifundefined{texorpdfstring}%
37   {\let\texorpdfstring\@firstoftwo}%
38   }%
39   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
40 }
41 \def\TeX{\stex}

```

(End definition for `\stex` and `\TeX`. These functions are documented on page 20.)

25.3 Messages and logging

```

42 <@@=stex_log>

Warnings and error messages
43 \msg_new:nnn{stex}{error/unknownlanguage}{
44   Unknown~language:~#1
45 }
46 \msg_new:nnn{stex}{warning/nomathhub}{
47   MATHHUB~system~variable~not~found~and~no~
48   \detokenize{\mathhub}~value~set!
49 }
50 \msg_new:nnn{stex}{error/deactivated-macro}{
51   The~\detokenize{#1}~command~is~only~allowed~in~#2!
52 }

\stex_debug:nn A simple macro issuing package messages with subpath.
53 \cs_new_protected:Nn \stex_debug:nn {
54   \clist_if_in:NnTF \c_stex_debug_clist { all } {
55     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
56       \\Debug~#1:~#2\\
57     }
58     \msg_none:nn{stex}{debug / #1}
59   }{
60     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
61       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
62         \\Debug~#1:~#2\\
63       }
64       \msg_none:nn{stex}{debug / #1}
65     }
66   }
67 }

```

(End definition for `\stex_debug:nn`. This function is documented on page 20.)

Redirecting messages:

```

68 \clist_if_in:NnTF \c_stex_debug_clist {all} {
69   \msg_redirect_module:nnn{ stex }{ none }{ term }
70 }{
71   \clist_map_inline:Nn \c_stex_debug_clist {
72     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
73   }
74 }
75
76 \stex_debug:nn{log}{debug~mode~on}

```

25.4 Persistence

77 `<@=stex_persist>`

`\c__stex_persist_sms_iow` File variable used for the sms-File

```

78 \iow_new:N \c__stex_persist_sms_iow
79 \AddToHook{begindocument}{
80   \bool_if:NTF \c_stex_persist_mode_bool {
81     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
82   } {
83     % \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
84   }
85 }
86 \AddToHook{enddocument}{
87   \bool_if:NF \c_stex_persist_mode_bool {
88     % \iow_close:N \c__stex_persist_sms_iow
89   }
90 }

```

(End definition for `\c__stex_persist_sms_iow`.)

`\stex_add_to_sms:n` Adds the provided code to the .sms-file of the document.

```

91 \cs_new_protected:Nn \stex_add_to_sms:n {
92   \bool_if:NF \c_stex_persist_mode_bool {
93     % \iow_now:Nn \c__stex_persist_sms_iow { #1 }
94   }
95 }

```

(End definition for `\stex_add_to_sms:n`. This function is documented on page 20.)

25.5 HTML Annotations

96 `<@=stex_annotate>`
97 `\RequirePackage{rustex}`

We add the namespace abbreviation `ns:stex="http://kwarc.info/ns/sTeX"` to `RuTeX`:

```

98 \rustex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}

```

`\if@latexml` Conditionals for L^AT_EX_ML:

```

\latexml_if_p:
\latexml_if:TF
99 \ifcsname if@latexml\endcsname\else

```



```

100 \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
101 \fi
102
103 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
104   \if@latexml
105     \prg_return_true:
106   \else:
107     \prg_return_false:
108   \fi:
109 }

```

(End definition for \if@latexml and \latexml_if:TF. These functions are documented on page 20.)

`\l__stex_annotate_arg_tl` Used by annotation macros to ensure that the HTML output to annotate is not empty.
`\c__stex_annotate_emptyarg_tl`

```

110 \tl_new:N \l__stex_annotate_arg_tl
111 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
112   \rustex_if:TF {
113     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
114   }{-}
115 }

```

(End definition for \l__stex_annotate_arg_tl and \c__stex_annotate_emptyarg_tl.)

`__stex_annotate_checkempty:n`

```

116 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
117   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
118   \tl_if_empty:NT \l__stex_annotate_arg_tl {
119     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
120   }
121 }

```

(End definition for __stex_annotate_checkempty:n.)

`\l_stex_html_do_output_bool` Whether to (locally) produce HTML output

```

\stex_if_do_html:
122 \bool_new:N \l_stex_html_do_output_bool
123 \bool_set_true:N \l_stex_html_do_output_bool
124 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
125   \bool_if:nTF \l_stex_html_do_output_bool
126     \prg_return_true: \prg_return_false:
127 }

```

(End definition for \l_stex_html_do_output_bool and \stex_if_do_html:. These functions are documented on page ??.)

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

```

128 \cs_new_protected:Nn \stex_suppress_html:n {
129   \exp_args:Nne \use:nn {
130     \bool_set_false:N \l_stex_html_do_output_bool
131     #1
132   }{
133     \stex_if_do_html:T {
134       \bool_set_true:N \l_stex_html_do_output_bool
135     }
136   }
137 }

```

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

`\stex_annotate:env`

`\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.

```

138 \rustex_if:TF{
139   \cs_new_protected:Nn \stex_annotate:nnn {
140     \__stex_annotate_checkempty:n { #3 }
141     \rustex_annotate_HTML:nn {
142       property="stex:#1" ~
143       resource="#2"
144     } {
145       \mode_if_vertical:TF{
146         \tl_use:N \l__stex_annotate_arg_tl\par
147       }{
148         \tl_use:N \l__stex_annotate_arg_tl
149       }
150     }
151   }
152   \cs_new_protected:Nn \stex_annotate_invisible:n {
153     \__stex_annotate_checkempty:n { #1 }
154     \rustex_annotate_HTML:nn {
155       stex:visible="false" ~
156       style:display="none"
157     } {
158       \mode_if_vertical:TF{
159         \tl_use:N \l__stex_annotate_arg_tl\par
160       }{
161         \tl_use:N \l__stex_annotate_arg_tl
162       }
163     }
164   }
165   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
166     \__stex_annotate_checkempty:n { #3 }
167     \rustex_annotate_HTML:nn {
168       property="stex:#1" ~
169       resource="#2" ~
170       stex:visible="false" ~
171       style:display="none"
172     } {
173       \mode_if_vertical:TF{
174         \tl_use:N \l__stex_annotate_arg_tl\par
175       }{
176         \tl_use:N \l__stex_annotate_arg_tl
177       }
178     }
179   }
180   \NewDocumentEnvironment{stex_annotate_env} { m m } {
181     \par
182     \rustex_annotate_HTML_begin:n {
183       property="stex:#1" ~
184       resource="#2"
185     }

```

```

186   }{
187     \par\rustex_annotate_HTML_end:
188   }
189 }{
190   \latexml_if:TF {
191     \cs_new_protected:Nn \stex_annotate:nnn {
192       \__stex_annotate_checkempty:n { #3 }
193       \mode_if_math:TF {
194         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
195           \tl_use:N \l__stex_annotate_arg_tl
196         }
197       }{
198         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
199           \tl_use:N \l__stex_annotate_arg_tl
200         }
201       }
202     }
203     \cs_new_protected:Nn \stex_annotate_invisible:n {
204       \__stex_annotate_checkempty:n { #1 }
205       \mode_if_math:TF {
206         \cs:w latexml@invisible@math\cs_end:{
207           \tl_use:N \l__stex_annotate_arg_tl
208         }
209       } {
210         \cs:w latexml@invisible@text\cs_end:{
211           \tl_use:N \l__stex_annotate_arg_tl
212         }
213       }
214     }
215     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
216       \__stex_annotate_checkempty:n { #3 }
217       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
218         \tl_use:N \l__stex_annotate_arg_tl
219       }
220     }
221     \NewDocumentEnvironment{stex_annotate_env} { m m } {
222       \par\begin{latexml@annotateenv}{#1}{#2}
223     }{
224       \par\end{latexml@annotateenv}
225     }
226   }{
227     \cs_new_protected:Nn \stex_annotate:nnn {#3}
228     \cs_new_protected:Nn \stex_annotate_invisible:n {}
229     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
230     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
231   }
232 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page [21](#).)

25.6 Languages

```

233 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

234 \prop_const_from_keyval:Nn \c_stex_languages_prop {
235   en = english ,
236   de = ngerman ,
237   ar = arabic ,
238   bg = bulgarian ,
239   ru = russian ,
240   fi = finnish ,
241   ro = romanian ,
242   tr = turkish ,
243   fr = french
244 }
245
246 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
247   english   = en ,
248   ngerman   = de ,
249   arabic    = ar ,
250   bulgarian = bg ,
251   russian   = ru ,
252   finnish   = fi ,
253   romanian  = ro ,
254   turkish   = tr ,
255   french    = fr
256 }
257 % todo: chinese simplified (zhs)
258 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 21.)

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

```

259 \clist_if_empty:NF \c_stex_languages_clist {
260   \clist_clear:N \l_tmpa_clist
261   \clist_map_inline:Nn \c_stex_languages_clist {
262     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
263       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
264     } {
265       \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
266     }
267   }
268   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
269   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
270 }

```

25.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

271 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
272   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
273   \def#1{
274     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
275   }
276 }

```

(End definition for `\stex_deactivate_macro:Nn`. This function is documented on page 21.)

`\stex_reactivate_macro:N`

```

277 \cs_new_protected:Nn \stex_reactivate_macro:N {
278   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
279 }

```

(End definition for `\stex_reactivate_macro:N`. This function is documented on page 21.)

`\stex_do_aftergroup:nn`

```

280 <@@=stex_aftergroup>
281 \tl_new:N \l__stex_aftergroup_tl
282 \cs_new_protected:Nn \stex_do_aftergroup:n {
283   \int_compare:nNnTF \l_stex_module_group_depth_int = \currentgrouplevel {
284     #1
285   }{
286     #1
287     \expandafter \tl_gset:Nn \expandafter \l__stex_aftergroup_tl \expandafter { \l__stex_aft
288     \aftergroup\__stex_aftergroup_do:
289   }
290 }
291 \cs_new_protected:Nn \__stex_aftergroup_do: {
292   \int_compare:nNnTF \l_stex_module_group_depth_int = \currentgrouplevel {
293     \l__stex_aftergroup_tl
294     \tl_clear:N \l__stex_aftergroup_tl
295   }{
296     \l__stex_aftergroup_tl
297     \aftergroup\__stex_aftergroup_do:
298   }
299 }

```

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

```

300 </package>

```

Chapter 26

STEX -MathHub Implementation

```
301 <*package>
302
303 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
304
305 <@@=stex_path>
306
307 Warnings and error messages
308 \msg_new:nnn{stex}{error/norepository}{
309   No~archive~#1~found~in~#2
310 }
311 \msg_new:nnn{stex}{error/notinarchive}{
312   Not~currently~in~an~archive,~but~\detokenize{#1}~
313   needs~one!
314 }
315 \msg_new:nnn{stex}{error/nofile}{
316   \detokenize{#1}~could~not~find~file~#2
317 }
318 \msg_new:nnn{stex}{error/twofiles}{
319   \detokenize{#1}~found~two~candidates~for~#2
320 }
```

26.1 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
\stex_path_from_string:NV
\stex_path_from_string:cn
\stex_path_from_string:cV
319 \cs_new_protected:Nn \stex_path_from_string:Nn {
320   \str_set:Nx \l_tmpa_str { #2 }
321   \str_if_empty:NTF \l_tmpa_str {
322     \seq_clear:N #1
323   }{
324     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
325     \sys_if_platform_windows:T{
326       \seq_clear:N \l_tmpa_tl
```

```

327     \seq_map_inline:Nn #1 {
328       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
329       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
330     }
331     \seq_set_eq:NN #1 \l_tmpa_tl
332   }
333   \stex_path_canonicalize:N #1
334 }
335 }
336 \cs_generate_variant:Nn \stex_path_from_string:Nn
337 { NV, cn, cV }

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 22.)

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

```

338 \cs_new_protected:Nn \stex_path_to_string:NN {
339   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
340 }
341
342 \cs_new:Nn \stex_path_to_string:N {
343   \seq_use:Nn #1 /
344 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 22.)

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

```

345 \str_const:Nn \c__stex_path_dot_str {.}
346 \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.

```

347 \cs_new_protected:Nn \stex_path_canonicalize:N {
348   \seq_if_empty:NF #1 {
349     \seq_clear:N \l_tmpa_seq
350     \seq_get_left:NN #1 \l_tmpa_tl
351     \str_if_empty:NT \l_tmpa_tl {
352       \seq_put_right:Nn \l_tmpa_seq {}
353     }
354     \seq_map_inline:Nn #1 {
355       \str_set:Nn \l_tmpa_tl { ##1 }
356       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
357         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
358           \seq_if_empty:NNTF \l_tmpa_seq {
359             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
360               \c__stex_path_up_str
361             }
362           }{
363             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
364             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
365               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
366                 \c__stex_path_up_str
367               }

```

```

368         }{
369         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
370         }
371     }
372     }{
373     \str_if_empty:NF \l_tmpa_tl {
374     \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
375     }
376     }
377 }
378 }
379 \seq_gset_eq:NN #1 \l_tmpa_seq
380 }
381 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 22.)

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

```

382 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
383   \seq_if_empty:NTF #1 {
384     \prg_return_false:
385   }{
386     \seq_get_left:NN #1 \l_tmpa_tl
387     \str_if_empty:NTF \l_tmpa_tl {
388       \prg_return_true:
389     }{
390       \prg_return_false:
391     }
392   }
393 }

```

(End definition for `\stex_path_if_absolute:NTF`. This function is documented on page 22.)

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

394 \str_new:N\l_stex_kpsewhich_return_str
395 \cs_new_protected:Nn \stex_kpsewhich:n {
396   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
397   \exp_args:NNo \str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
398   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
399 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 22.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

400 \sys_if_platform_windows:TF{
401   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
402 }{
403   \stex_kpsewhich:n{-var-value~PWD}
404 }
405

```



```

406 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
407 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
408 \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 22.)

26.3 File Hooks and Tracking

```

409 <@@=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

```

410 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

411 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
412 \stex_path_from_string:Nn \c_stex_mainfile_seq
413 \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 22.)

`\g_stex_currentfile_seq` Hooks for file inputs that push/pop `\g_stex_files_stack` to update `\c_stex_mainfile_seq`.

```

414 \seq_gclear_new:N\g_stex_currentfile_seq
415 \cs_new_protected:Nn \stex_filestack_push:n {
416   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
417   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
418     \stex_path_from_string:Nn\g_stex_currentfile_seq{
419       \c_stex_pwd_str/#1
420     }
421   }
422   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
423   \exp_args:NNo\seq_gpush:Nn\g_stex_files_stack\g_stex_currentfile_seq
424 }
425 \cs_new_protected:Nn \stex_filestack_pop: {
426   \seq_if_empty:NF\g_stex_files_stack{
427     \seq_gpop:NN\g_stex_files_stack\l_tmpa_seq
428   }
429   \seq_if_empty:NTF\g_stex_files_stack{
430     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
431   }{
432     \seq_get:NN\g_stex_files_stack\l_tmpa_seq
433     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
434   }
435 }
436

```

```

437 \AddToHook{file/before}{
438   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
439 }
440 \AddToHook{file/after}{
441   \stex_filestack_pop:
442 }

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 23.)

26.4 MathHub Repositories

```

443 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
444 \str_if_empty:NTF\mathhub{
445   \stex_kpsewhich:n{-var-value~MATHHUB}
446   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
447
448   \str_if_empty:NTF\c_stex_mathhub_str{
449     \msg_warning:nn{stex}{warning/nomathhub}
450   }{
451     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
452     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
453   }
454 }{
455   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
456   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
457     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
458       \c_stex_pwd_str/\mathhub
459     }
460   }
461   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
462   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
463 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 23.)

```

\__stex_mathhub_do_manifest:n
464 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
465   \str_set:Nx \l_tmpa_str { #1 }
466   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
467     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
468     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
469     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
470     \__stex_mathhub_find_manifest:N \l_tmpa_seq
471     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
472       \msg_error:nnxx{stex}{error/norepository}{#1}{
473         \stex_path_to_string:N \c_stex_mathhub_str
474       }
475     } {
476       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
477     }
478   }
479 }

```

(End definition for _stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

480 \str_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:

```

481 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
482   \seq_set_eq:NN\l_tmpa_seq #1
483   \bool_set_true:N\l_tmpa_bool
484   \bool_while_do:Nn \l_tmpa_bool {
485     \seq_if_empty:NTF \l_tmpa_seq {
486       \bool_set_false:N\l_tmpa_bool
487     }{
488       \file_if_exist:nTF{
489         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
490       }{
491         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
492         \bool_set_false:N\l_tmpa_bool
493       }{
494         \file_if_exist:nTF{
495           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
496         }{
497           \seq_put_right:Nn\l_tmpa_seq{META-INF}
498           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
499           \bool_set_false:N\l_tmpa_bool
500         }{
501           \file_if_exist:nTF{
502             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
503           }{
504             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
505             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
506             \bool_set_false:N\l_tmpa_bool
507           }{
508             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
509           }
510         }
511       }
512     }
513   }
514   \seq_set_eq:NN\l_stex_mathhub_manifest_file_seq\l_tmpa_seq
515 }

```

(End definition for _stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior File variable used for MANIFEST-files

516 \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:

```

517 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
518   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
519   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
520   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
521     \str_set:Nn \l_tmpa_str {##1}
522     \exp_args:NNoo \seq_set_split:Nnn
523       \l_tmpb_seq \c_colon_str \l_tmpa_str
524     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
525       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
526         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
527       }
528       \exp_args:No \str_case:nnTF \l_tmpa_tl {
529         {id} {
530           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
531             { id } \l_tmpb_tl
532         }
533         {narration-base} {
534           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
535             { narr } \l_tmpb_tl
536         }
537         {url-base} {
538           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
539             { docurl } \l_tmpb_tl
540         }
541         {source-base} {
542           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
543             { ns } \l_tmpb_tl
544         }
545         {ns} {
546           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
547             { ns } \l_tmpb_tl
548         }
549         {dependencies} {
550           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
551             { deps } \l_tmpb_tl
552         }
553       }{}{}
554     }{}
555   }
556   \ior_close:N \c__stex_mathhub_manifest_ior
557 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

558 \cs_new_protected:Nn \stex_set_current_repository:n {
559   \stex_require_repository:n { #1 }
560   \prop_set_eq:Nc \l_stex_current_repository_prop {
561     c_stex_mathhub_#1_manifest_prop
562   }
563 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 24.)

`\stex_require_repository:n`

```

564 \cs_new_protected:Nn \stex_require_repository:n {
565   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
566     \stex_debug:nn{mathhub}{Opening~archive:~#1}
567     \__stex_mathhub_do_manifest:n { #1 }
568     \exp_args:Nx \stex_add_to_sms:n {
569       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
570         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
571         ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,
572         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
573         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
574       }
575     }
576   }
577 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 24.)

`\l_stex_current_repository_prop` Current MathHub repository

```

578 %\prop_new:N \l_stex_current_repository_prop
579
580 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
581 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
582   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
583 } {
584   \__stex_mathhub_parse_manifest:n { main }
585   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
586   \l_tmpa_str
587   \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
588   \c_stex_mathhub_main_manifest_prop
589   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
590   \stex_debug:nn{mathhub}{Current~repository:~
591     \prop_item:Nn \l_stex_current_repository_prop {id}
592   }
593 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 23.)

`\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.

```

594 \cs_new_protected:Nn \stex_in_repository:nn {
595   \str_set:Nx \l_tmpa_str { #1 }
596   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
597   \str_if_empty:NTF \l_tmpa_str {
598     \prop_if_exist:NTF \l_stex_current_repository_prop {
599       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
600       \exp_args:Ne \l_tmpa_cs{
601         \prop_item:Nn \l_stex_current_repository_prop { id }
602       }
603     }{
604       \l_tmpa_cs{}
605     }
606   }{
607     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}

```

```

608 \stex_require_repository:n \l_tmpa_str
609 \str_set:Nx \l_tmpa_str { #1 }
610 \exp_args:Nne \use:nn {
611   \stex_set_current_repository:n \l_tmpa_str
612   \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
613 }{
614   \stex_debug:nn{mathhub}{switching~back~to:~
615     \prop_if_exist:NTF \l_stex_current_repository_prop {
616       \prop_item:Nn \l_stex_current_repository_prop { id }::~
617       \meaning\l_stex_current_repository_prop
618     }{
619       no~repository
620     }
621   }
622   \prop_if_exist:NTF \l_stex_current_repository_prop {
623     \stex_set_current_repository:n {
624       \prop_item:Nn \l_stex_current_repository_prop { id }
625     }
626   }{
627     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
628   }
629 }
630 }
631 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 24.)

```

\inputref
\stex_inputref:nn 632 \newif \ifinputref \inputreffalse
\mhinput\stex_mhinput:nn 633
634 \cs_new_protected:Nn \stex_mhinput:nn {
635   \stex_in_repository:nn {#1} {
636     \ifinputref
637       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
638     \else
639       \inputreftrue
640       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
641       \inputreffalse
642     \fi
643   }
644 }
645 \NewDocumentCommand \mhinput { 0{} m}{
646   \stex_mhinput:nn{ #1 }{ #2 }
647 }
648
649 \cs_new_protected:Nn \stex_inputref:nn {
650   \stex_in_repository:nn {#1} {
651     \bool_lazy_any:nTF {
652       {\rustex_if_p:} {\latexml_if_p:}
653     } {
654       \str_clear:N \l_tmpa_str
655       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
656         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
657       }

```

```

658     \stex_annotate_invisible:nnn{inputref}{
659       \l_tmpa_str / #2
660     }{}
661   }{
662     \begingroup
663       \inputreftrue
664       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
665     \endgroup
666   }
667 }
668 }
669
670 \NewDocumentCommand \inputref { 0{ } m }{
671   \stex_inputref:nn{ #1 }{ #2 }
672 }
673
674 \cs_new_protected:Nn \stex_mhbibresource:nn {
675   \stex_in_repository:nn {#1} {
676     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
677   }
678 }
679 \newcommand\addmhbibresource[2][]{
680   \stex_mhbibresource:nn{ #1 }{ #2 }
681 }

```

(End definition for `\inputref`, `\stex_inputref:nn`, and `\mhinput\stex_mhinput:nn`. These functions are documented on page 24.)

`\mhpath`

```

682 \def \mhpath #1 #2 {
683   \exp_args:Ne \str_if_eq:nnTF{#1}{#2}{
684     \c_stex_mathhub_str /
685     \prop_item:Nn \l_stex_current_repository_prop { id }
686     / source / #2
687   }{
688     \c_stex_mathhub_str / #1 / source / #2
689   }
690 }

```

(End definition for `\mhpath`. This function is documented on page 24.)

`\libinput`

```

691 \cs_new_protected:Npn \libinput #1 {
692   \prop_if_exist:NF \l_stex_current_repository_prop {
693     \msg_error:nnn{stex}{error/notinarchive}\libinput
694   }
695   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
696     \msg_error:nnn{stex}{error/notinarchive}\libinput
697   }
698   \bool_set_false:N \l_tmpa_bool
699   \tl_clear:N \l_tmpa_tl
700   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
701   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
702   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
703   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {

```

```

704 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
705 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
706 / meta-inf / lib / #1.tex}{
707 \bool_set_true:N \l_tmpa_bool
708 \tl_put_right:Nx \l_tmpa_tl {
709 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
710 / meta-inf / lib / #1.tex}
711 }
712 }{}
713 }
714 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
715 / \l_tmpa_str / lib / #1.tex
716 }{
717 \bool_set_true:N \l_tmpa_bool
718 \tl_put_right:Nx \l_tmpa_tl {
719 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
720 / \l_tmpa_str / lib / #1.tex}
721 }
722 }{}
723 \bool_if:NF \l_tmpa_bool {
724 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
725 }
726 \l_tmpa_tl
727 }

```

(End definition for `\libinput`. This function is documented on page 24.)

`\libusepackage`

```

728 \NewDocumentCommand \libusepackage {0{} m} {
729 \prop_if_exist:NF \l_stex_current_repository_prop {
730 \msg_error:nnn{stex}{error/notinarchive}\libusepackage
731 }
732 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
733 \msg_error:nnn{stex}{error/notinarchive}\libusepackage
734 }
735 \bool_set_false:N \l_libusepackage_bool
736 \tl_clear:N \l_tmpa_tl
737 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
738 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
739 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
740 \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
741 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
742 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
743 / meta-inf / lib / #2.sty}{
744 \bool_set_true:N \l_libusepackage_bool
745 \tl_put_right:Nx \l_tmpa_tl {
746 \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
747 / meta-inf / lib / #2}
748 }
749 }{}
750 }
751 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
752 / \l_tmpa_str / lib / #2.sty
753 }{

```



```

754 \bool_if:NT \l_libusepackage_bool {
755   \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
756 }
757 \bool_set_true:N \l_libusepackage_bool
758 \tl_put_right:Nx \l_tmpa_tl {
759   \exp_not:N \usepackage[#1] { \stex_path_to_string:N \l_tmpa_seq
760     / \l_tmpa_str / lib / #2}
761 }
762 }{}
763 \bool_if:NF \l_libusepackage_bool {
764   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
765 }
766 \l_tmpa_tl
767 }

```

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

```

768
769 \AddToHook{begindocument}{
770 \ltx@ifpackageloaded{graphicx}{
771   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
772   \newcommand\mhgraphics[2][]{%
773     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
774     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
775   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
776 }{}
777 \ltx@ifpackageloaded{listings}{
778   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
779   \newcommand\lstinputmhlstlisting[2][]{%
780     \def\lst@mhrepos{}\setkeys{lst}{#1}%
781     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
782   \newcommand\clstinputmhlstlisting[2][]{\begin{center}\lstinputmhlstlisting[#1]{#2}\end{center}}
783 }{}
784 }
785
786
787 \end{package}

```

Chapter 27

STEX -References Implementation

```
788 <*package>
789
790 %%%%%%%%%% references.dtx %%%%%%%%%%
791
792 %\RequirePackage{hyperref}
793 %\RequirePackage{cleveref}
794 <@@=stex_refs>
795
796 Warnings and error messages
797
798 \iow_new:N \c__stex_refs_refs_iow
799 \AddToHook{begindocument}{
800   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
801 }
802 \AddToHook{enddocument}{
803   \iow_close:N \c__stex_refs_refs_iow
804 }
805
806 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
807
808 \NewDocumentCommand \STEXreftitle { m } {
809   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
810 }
811
```

27.1 Document URIs and URLs

```
809 \seq_new:N \g__stex_refs_all_refs_seq
810
811 \str_new:N \l_stex_current_docns_str
812
813 \cs_new_protected:Nn \stex_get_document_uri: {
814   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
815   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
816   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
817   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
818 }
819
```

```

818 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
819
820 \str_clear:N \l_tmpa_str
821 \prop_if_exist:NT \l_stex_current_repository_prop {
822   \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
823     \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
824   }
825 }
826
827 \str_if_empty:NTF \l_tmpa_str {
828   \str_set:Nx \l_stex_current_docns_str {
829     file:/\stex_path_to_string:N \l_tmpa_seq
830   }
831 }{
832   \bool_set_true:N \l_tmpa_bool
833   \bool_while_do:Nn \l_tmpa_bool {
834     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
835     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
836       {source} { \bool_set_false:N \l_tmpa_bool }
837     }{}{
838       \seq_if_empty:NT \l_tmpa_seq {
839         \bool_set_false:N \l_tmpa_bool
840       }
841     }
842   }
843
844   \seq_if_empty:NTF \l_tmpa_seq {
845     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
846   }{
847     \str_set:Nx \l_stex_current_docns_str {
848       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
849     }
850   }
851 }
852 }
853
854 \str_new:N \l_stex_current_docurl_str
855 \cs_new_protected:Nn \stex_get_document_url: {
856   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
857   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
858   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
859   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
860   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
861
862   \str_clear:N \l_tmpa_str
863   \prop_if_exist:NT \l_stex_current_repository_prop {
864     \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
865       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
866         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
867       }
868     }
869
870   \str_if_empty:NTF \l_tmpa_str {
871     \str_set:Nx \l_stex_current_docurl_str {

```

```

872     file:/\stex_path_to_string:N \l_tmpa_seq
873   }
874 }{
875   \bool_set_true:N \l_tmpa_bool
876   \bool_while_do:Nn \l_tmpa_bool {
877     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
878     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
879       {source} { \bool_set_false:N \l_tmpa_bool }
880     }{}{
881       \seq_if_empty:NT \l_tmpa_seq {
882         \bool_set_false:N \l_tmpa_bool
883       }
884     }
885   }
886
887   \seq_if_empty:NTF \l_tmpa_seq {
888     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
889   }{
890     \str_set:Nx \l_stex_current_docurl_str {
891       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
892     }
893   }
894 }
895 }

```

27.2 Setting Reference Targets

```

896 \str_const:Nn \c__stex_refs_url_str{URL}
897 \str_const:Nn \c__stex_refs_ref_str{REF}
898 % @currentlabel -> number
899 % @currentlabelname -> title
900 % @currentHref -> name.number <- id of some kind
901 % \theH# -> \arabic{section}
902 % \the# -> number
903 % \hyper@makecurrent{#}
904 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
905   \stex_get_document_uri:
906   \str_set:Nx \l_tmpa_str { #1 }
907   \str_if_empty:NT \l_tmpa_str {
908     \int_zero:N \l_tmpa_int
909     \bool_set_true:N \l_tmpa_bool
910     \bool_while_do:Nn \l_tmpa_bool {
911       \cs_if_exist:cTF {
912         sref_\l_stex_current_docns_str?? REF_\int_use:N \l_tmpa_int _type
913       }{
914         \int_incr:N \l_tmpa_int
915       }{
916         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
917         \bool_set_false:N \l_tmpa_bool
918       }
919     }
920   }
921   \str_set:Nx \l_tmpa_str {
922     \l_stex_current_docns_str??\l_tmpa_str

```

```

923 }
924 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
925 \stex_if_smsmode:TF {
926   \stex_get_document_url:
927   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
928   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
929 }{
930   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
931   \exp_args:Nx\label{sref_\l_tmpa_str}
932   \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{\l_tmpa_str}}
933   \str_gset:cx {sref_\l_tmpa_str _type}\c__stex_refs_ref_str
934 }
935 }
936 \cs_new_protected:Npn \stexauxadddocref #1 {
937   \str_set:Nx \l_tmpa_str {#1}
938   \str_gset_eq:cN{sref_\l_tmpa_str _type}\c__stex_refs_ref_str
939   \seq_gput_right:Nx \g__stex_refs_all_refs_seq {\l_tmpa_str}
940 }
941 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
942   \stex_get_document_uri:
943   \stex_if_smsmode:TF {
944     \stex_get_document_url:
945     \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str
946     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
947   }
948   }{
949     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
950     \exp_args:Nx\label{sref_sym_#1}
951
952     \exp_args:NNNx\immediate\write\@auxout{\stexauxadddocref{sym_#1}}
953     \str_gset:cx {sref_sym_#1_type}\c__stex_refs_ref_str
954   }
955 }

```

27.3 Using References

```

956 \str_new:N \l__stex_refs_indocument_str
957 \keys_define:nn { stex / sref } {
958   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
959   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
960   pre           .tl_set:N = \l__stex_refs_pre_tl ,
961   post          .tl_set:N = \l__stex_refs_post_tl ,
962   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
963 }
964
965 \bool_new:N \c__stex_refs_hyperref_bool
966 \bool_set_false:N \c__stex_refs_hyperref_bool
967 \AddToHook{begindocument}{
968   \@ifpackageloaded{hyperref}{
969     \bool_set_true:N \c__stex_refs_hyperref_bool
970   }{}
971 }
972
973

```

```

974 \cs_new_protected:Nn \__stex_refs_args:n {
975   \tl_clear:N \l__stex_refs_linktext_tl
976   \tl_clear:N \l__stex_refs_fallback_tl
977   \tl_clear:N \l__stex_refs_pre_tl
978   \tl_clear:N \l__stex_refs_post_tl
979   \str_clear:N \l__stex_refs_repo_str
980   \keys_set:nn { stex / sref } { #1 }
981 }
982
983 \NewDocumentCommand \sref { 0{} m }{
984   \__stex_refs_args:n { #1 }
985   \str_if_empty:NTF \l__stex_refs_indocument_str {
986     \str_set:Nn \l_tmpa_str { #2 }
987     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
988     \tl_set:Nn \l_tmpa_tl {
989       \l__stex_refs_fallback_tl
990     }
991     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
992       \str_set:Nn \l_tmpb_str { ##1 }
993       \str_if_eq:eeT { \l_tmpa_str } {
994         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
995       } {
996         \seq_map_break:n {
997           \tl_set:Nn \l_tmpa_tl {
998             % doc uri in \l_tmpb_str
999             \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str _type}}
1000             \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1001               % reference
1002               \cs_if_exist:cTF{autoref}{
1003                 \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
1004               }{
1005                 \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
1006               }
1007             }{
1008               % URL
1009               \if_bool:N \c__stex_refs_hyperref_bool {
1010                 \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str _str}}{\l__stex_refs_fallback
1011               }{
1012                 \l__stex_refs_fallback_tl
1013               }
1014             }
1015           }
1016         }
1017       }
1018     }
1019     \l_tmpa_tl
1020   }{
1021     % TODO
1022   }
1023 }
1024
1025 \NewDocumentCommand \srefsym { 0{} m }{
1026   \stex_get_symbol:n { #2 }
1027   \__stex_refs_args:n { #1 }

```

```

1028 \str_if_empty:NTF \l__stex_refs_indocument_str {
1029   \tl_set:Nn \l_tmpa_tl {
1030     \l__stex_refs_fallback_tl
1031   }
1032   \tl_if_exist:cT{sref_sym_\l_stex_get_symbol_uri_str_type}{
1033     \tl_set:Nn \l_tmpa_tl {
1034       % doc uri in \l_tmpb_str
1035       \str_set:Nx \l_tmpa_str {\use:c{sref_sym_\l_stex_get_symbol_uri_str_type}}
1036       \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1037         % reference
1038         \cs_if_exist:cTF{autoref}{
1039           \l__stex_refs_pre_tl\autoref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1040         }{
1041           \l__stex_refs_pre_tl\ref{sref_sym_\l_stex_get_symbol_uri_str}\l__stex_refs_post_tl
1042         }
1043       }{
1044         % URL
1045         \if_bool:N \c__stex_refs_hyperref_bool {
1046           \exp_args:Nx \href{\use:c{sref_sym_url_\l_stex_get_symbol_uri_str_str}}{\l__stex_refs_ref_str}
1047         }{
1048           \l__stex_refs_fallback_tl
1049         }
1050       }
1051     }
1052   }
1053   \l_tmpa_tl
1054 }{
1055   % TODO
1056 }
1057 }
1058
1059 \cs_new_protected:Npn \srefsymuri #1 #2 {
1060   \hyperref[sref_sym_#1]{#2}
1061 }
1062
1063 </package>

```

Chapter 28

STEX -Modules Implementation

```
1064 <*package>
1065
1066 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1067
1068 <@@=stex_modules>
1069
1070 Warnings and error messages
1071 \msg_new:nnn{stex}{error/unknownmodule}{
1072   No~module~#1~found
1073 }
1074 \msg_new:nnn{stex}{error/syntax}{
1075   Syntax~error:~#1
1076 }
1077 \msg_new:nnn{stex}{error/siglanguage}{
1078   Module~#1~declares~signature~#2,~but~does~not~
1079   declare~its~language
1080 }
1081 \msg_new:nnn{stex}{error/conflictingmodules}{
1082   Conflicting~imports~for~module~#1
1083 }
1084
1085 \l_stex_current_module_str The current module:
1086 \str_new:N \l_stex_current_module_str
1087
1088 (End definition for \l_stex_current_module_str. This variable is documented on page 26.)
1089
1090 \l_stex_all_modules_seq Stores all available modules
1091 \seq_new:N \l_stex_all_modules_seq
1092
1093 (End definition for \l_stex_all_modules_seq. This variable is documented on page 26.)
1094
1095 \stex_if_in_module_p:
1096 \stex_if_in_module:TF
1097 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1098   \str_if_empty:NTF \l_stex_current_module_str
1099   \prg_return_false: \prg_return_true:
1100 }
```


(End definition for `\stex_if_in_module:TF`. This function is documented on page 27.)

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

```
1089 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1090   \prop_if_exist:cTF { c_stex_module_#1_prop }
1091   \prg_return_true: \prg_return_false:
1092 }
```

(End definition for `\stex_if_module_exists:nTF`. This function is documented on page 27.)

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

```
1093 \cs_new_protected:Nn \stex_add_to_current_module:n {
1094   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1095 }
1096 \cs_new_protected:Npn \STEXexport {
1097   \begingroup
1098   \newlinechar=-1\relax
1099   \endlinechar=-1\relax
1100   %\catcode'\ = 9\relax
1101   \expandafter\endgroup\STEXexport:n
1102 }
1103 \cs_new_protected:Nn \STEXexport:n {
1104   \ignorespaces #1
1105   \stex_add_to_current_module:n { \ignorespaces #1 }
1106   \stex_smsmode_do:
1107 }
1108 \stex_deactivate_macro:Nn \STEXexport {module~environments}
```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 27.)

`\stex_add_constant_to_current_module:n`

```
1109 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1110   \str_set:Nx \l_tmpa_str { #1 }
1111   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1112 }
1113
1114 %\cs_new_protected:Nn \stex_add_field_to_current_module:n {
1115 % \str_set:Nx \l_tmpa_str { #1 }
1116 % \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
1117 %}
```

(End definition for `\stex_add_constant_to_current_module:n`. This function is documented on page 27.)

`\stex_collect_imports:n`

```
1118 \cs_new_protected:Nn \stex_collect_imports:n {
1119   \seq_clear:N \l_stex_collect_imports_seq
1120   \__stex_modules_collect_imports:n {#1}
1121 }
1122 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1123   \seq_map_inline:cn {c_stex_module_#1_imports} {
1124     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1125       \__stex_modules_collect_imports:n { ##1 }
1126     }
1127 }
```

```

1127 }
1128 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1129   \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1130 }
1131 }

```

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

`\stex_add_import_to_current_module:n`

```

1132 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1133   \str_set:Nx \l_tmpa_str { #1 }
1134   \exp_args:Nno
1135   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
1136     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
1137   }
1138 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 27.)

`\stex_modules_compute_namespace:nN`

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

```

1139 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
1140   \str_set:Nx \l_tmpa_str { #1 }
1141   \seq_set_eq:NN \l_tmpa_seq #2
1142   % split off file extension
1143   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1144   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1145   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1146   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1147
1148   \bool_set_true:N \l_tmpa_bool
1149   \bool_while_do:Nn \l_tmpa_bool {
1150     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1151     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1152       {source} { \bool_set_false:N \l_tmpa_bool }
1153     }{}{
1154       \seq_if_empty:NT \l_tmpa_seq {
1155         \bool_set_false:N \l_tmpa_bool
1156       }
1157     }
1158   }
1159
1160   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1161   \str_if_empty:NTF \l_stex_modules_subpath_str {
1162     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1163   }{
1164     \str_set:Nx \l_stex_modules_ns_str {
1165       \l_tmpa_str/\l_stex_modules_subpath_str
1166     }
1167   }
1168 }

```

(End definition for `\stex_modules_compute_namespace:nN`. This function is documented on page 27.)

Stores its return values in:

```

\l_stex_modules_ns_str
\l_stex_modules_subpath_str
1169 \str_new:N \l_stex_modules_ns_str
1170 \str_new:N \l_stex_modules_subpath_str

(End definition for \l_stex_modules_ns_str and \l_stex_modules_subpath_str. These variables are
documented on page ??.)

```

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

```

1171 \cs_new_protected:Nn \stex_modules_current_namespace: {
1172   \str_clear:N \l_stex_modules_subpath_str
1173   \prop_if_exist:NTF \l_stex_current_repository_prop {
1174     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1175     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1176   }{
1177     % split off file extension
1178     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1179     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1180     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1181     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1182     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1183     \str_set:Nx \l_stex_modules_ns_str {
1184       file:/\stex_path_to_string:N \l_tmpa_seq
1185     }
1186   }
1187 }

```

(End definition for `\stex_modules_current_namespace:`. This function is documented on page 27.)

28.1 The module environment

module arguments:

```

1188 \keys_define:nn { stex / module } {
1189   title      .tl_set:N      = \smodulename ,
1190   type       .str_set_x:N   = \smodulename ,
1191   id         .str_set_x:N   = \smoduleid ,
1192   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1193   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1194   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1195   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1196   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1197   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1198   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1199 }
1200
1201 \cs_new_protected:Nn \__stex_modules_args:n {
1202   \str_clear:N \smodulename
1203   \str_clear:N \smodulename
1204   \str_clear:N \smoduleid
1205   \str_clear:N \l_stex_module_ns_str
1206   \str_clear:N \l_stex_module_lang_str
1207   \str_clear:N \l_stex_module_sig_str
1208   \str_clear:N \l_stex_module_creators_str

```

```

1209 \str_clear:N \l_stex_module_contributors_str
1210 \str_clear:N \l_stex_module_meta_str
1211 \str_clear:N \l_stex_module_srccite_str
1212 \keys_set:nn { stex / module } { #1 }
1213 }
1214
1215 % module parameters here? In the body?
1216

```

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

```

1217 \cs_new_protected:Nn \stex_module_setup:nn {
1218   \str_set:Nx \l_stex_module_name_str { #2 }
1219   \__stex_modules_args:n { #1 }

   First, we set up the name and namespace of the module.
   Are we in a nested module?

1220   \stex_if_in_module:TF {
1221     % Nested module
1222     \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1223       { ns } \l_stex_module_ns_str
1224     \str_set:Nx \l_stex_module_name_str {
1225       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1226         { name } / \l_stex_module_name_str
1227     }
1228   }{
1229     % not nested:
1230     \str_if_empty:NT \l_stex_module_ns_str {
1231       \stex_modules_current_namespace:
1232       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1233       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1234         / { \l_stex_module_ns_str }
1235       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1236       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1237         \str_set:Nx \l_stex_module_ns_str {
1238           \stex_path_to_string:N \l_tmpa_seq
1239         }
1240       }
1241     }
1242   }

```

Next, we determine the language of the module:

```

1243 \str_if_empty:NT \l_stex_module_lang_str {
1244   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1245   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1246   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1247   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1248   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1249     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1250       inferred~from~file~name}
1251     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1252   }
1253 }
1254
1255 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1256 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1257 \l_tmpa_str {
1258   \ltx@ifpackageloaded{babel}{
1259     \exp_args:Nx \selectlanguage { \l_tmpa_str }
1260   }{}
1261 } {
1262   \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1263 }
1264 }}

```

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

```

1265 \str_if_empty:NTF \l_stex_module_sig_str {
1266   \exp_args:Nnx \prop_gset_from_keyval:cn {
1267     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1268   } {
1269     name      = \l_stex_module_name_str ,
1270     ns        = \l_stex_module_ns_str ,
1271     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1272     lang      = \l_stex_module_lang_str ,
1273     sig       = \l_stex_module_sig_str ,
1274     meta      = \l_stex_module_meta_str
1275   }
1276   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1277   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1278   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1279   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1280   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1281 \str_if_empty:NT \l_stex_module_meta_str {
1282   \str_set:Nx \l_stex_module_meta_str {
1283     \c_stex_metatheory_ns_str ? Metatheory
1284   }
1285 }
1286 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1287   \bool_set_true:N \l_stex_in_meta_bool
1288   \exp_args:Nx \stex_add_to_current_module:n {
1289     \bool_set_true:N \l_stex_in_meta_bool
1290     \stex_activate_module:n {\l_stex_module_meta_str}
1291     \bool_set_false:N \l_stex_in_meta_bool
1292   }
1293   \stex_activate_module:n {\l_stex_module_meta_str}
1294   \bool_set_false:N \l_stex_in_meta_bool
1295 }
1296 }{
1297   \str_if_empty:NT \l_stex_module_lang_str {
1298     \msg_error:nnxx{stex}{error/siglanguage}{
1299       \l_stex_module_ns_str?\l_stex_module_name_str
1300     }{\l_stex_module_sig_str}
1301   }
1302
1303   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1304   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1305 \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1306 \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1307 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1308 \str_set:Nx \l_tmpa_str {
1309   \stex_path_to_string:N \l_tmpa_seq /
1310   \l_tmpa_str . \l_stex_module_sig_str .tex
1311 }
1312 \IfFileExists \l_tmpa_str {
1313   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1314     \str_clear:N \l_stex_current_module_str
1315     \seq_clear:N \l_stex_all_modules_seq
1316     \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1317   }
1318 }{
1319   \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1320 }
1321 \stex_if_smsmode:F {
1322   \stex_activate_module:n {
1323     \l_stex_module_ns_str ? \l_stex_module_name_str
1324   }
1325 }
1326 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1327 }
1328 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 28.)

module The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}

1329 \int_new:N \l_stex_module_group_depth_int
1330 \cs_new_protected:Nn \__stex_modules_begin_module: {
1331   \stex_reactivate_macro:N \STEXexport
1332   \stex_reactivate_macro:N \importmodule
1333   \stex_reactivate_macro:N \symdecl
1334   \stex_reactivate_macro:N \notation
1335   \stex_reactivate_macro:N \symdef
1336
1337   \stex_debug:nn{modules}{
1338     New~module:\\
1339     Namespace:~\l_stex_module_ns_str\\
1340     Name:~\l_stex_module_name_str\\
1341     Language:~\l_stex_module_lang_str\\
1342     Signature:~\l_stex_module_sig_str\\
1343     Metatheory:~\l_stex_module_meta_str\\
1344     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1345   }
1346
1347   \seq_put_right:Nx \l_stex_all_modules_seq {
1348     \l_stex_module_ns_str ? \l_stex_module_name_str
1349   }
1350
1351   % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1352   % { \l_stex_module_ns_str ? \l_stex_module_name_str }

```

```

1353
1354
1355 \stex_if_smsmode:F{
1356   \begin{stex_annotate_env} {theory} {
1357     \l_stex_module_ns_str ? \l_stex_module_name_str
1358   }
1359
1360   \stex_annotate_invisible:nnn{header}{} {
1361     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1362     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1363     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1364       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1365     }
1366     \str_if_empty:NF \smoduletype {
1367       \stex_annotate:nnn{type}{\smoduletype}{}
1368     }
1369   }
1370 }
1371 \int_set:Nn \l_stex_module_group_depth_int {\currentgrouplevel}
1372 % TODO: Inherit metatheory for nested modules?
1373 }
1374 \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}

1375 \cs_new_protected:Nn \_stex_modules_end_module: {
1376   % \str_set:Nx \l_tmpa_str {
1377   %   c_stex_module_
1378   %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1379   %   \prop_item:Nn \l_stex_current_module_prop { name }
1380   %   _prop
1381   % }
1382   %^^A \prop_new:c { \l_tmpa_str }
1383   % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1384   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1385   }

```

(End definition for _stex_modules_end_module:.)

smodule The core environment, with no header

```

1386 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1387 \NewDocumentEnvironment { smodule } { 0{} m } {
1388   \stex_module_setup:nn{#1}{#2}
1389   \par
1390   \stex_if_smsmode:F{
1391     \tl_clear:N \l_tmpa_tl
1392     \clist_map_inline:Nn \smoduletype {
1393       \tl_if_exist:cT {\_stex_modules_smodule_##1_start:}{
1394         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_start:}}
1395       }
1396     }
1397     \tl_if_empty:NTF \l_tmpa_tl {
1398       \_stex_modules_smodule_start:

```

```

1399     }{
1400         \l_tmpa_tl
1401     }
1402 }
1403 \__stex_modules_begin_module:
1404 \stex_ref_new_doc_target:n \smoduleid
1405 \stex_smsmode_do:
1406 } {
1407 \__stex_modules_end_module:
1408 \stex_if_smsmode:TF {
1409 %     \exp_args:Nx \stex_add_to_sms:n {
1410 %         \prop_gset_from_keyval:cn {
1411 %             c_stex_module_
1412 %             \prop_item:Nn \l_stex_current_module_prop { ns } ?
1413 %             \prop_item:Nn \l_stex_current_module_prop { name }
1414 %             _prop
1415 %         } {
1416 %             name      = \prop_item:cn { \l_tmpa_str } { name } ,
1417 %             ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1418 %             file      = \prop_item:cn { \l_tmpa_str } { file } ,
1419 %             lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1420 %             sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1421 %             meta      = \prop_item:cn { \l_tmpa_str } { meta }
1422 %         }
1423 %     }
1424 }{
1425     \end{stex_annotate_env}
1426     \clist_set:No \l_tmpa_clist \smoduletype
1427     \tl_clear:N \l_tmpa_tl
1428     \clist_map_inline:Nn \l_tmpa_clist {
1429         \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1430             \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1431         }
1432     }
1433     \tl_if_empty:NTF \l_tmpa_tl {
1434         \__stex_modules_smodule_end:
1435     }{
1436         \l_tmpa_tl
1437     }
1438 }
1439 }
1440
1441 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1442 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1443
1444 \newcommand\stexpatchmodule[3] [] {
1445     \str_set:Nx \l_tmpa_str{ #1 }
1446     \str_if_empty:NTF \l_tmpa_str {
1447         \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1448         \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1449     }{
1450         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1451         \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1452     }

```



```

1453 }
1454

```

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1455 \NewDocumentCommand \STEXModule { m } {
1456   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1457   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1458   \tl_set:Nn \l_tmpa_tl {
1459     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1460   }
1461   \seq_map_inline:Nn \l_stex_all_modules_seq {
1462     \str_set:Nn \l_tmpb_str { ##1 }
1463     \str_if_eq:eeT { \l_tmpa_str } {
1464       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1465     } {
1466       \seq_map_break:n {
1467         \tl_set:Nn \l_tmpa_tl {
1468           \stex_invoke_module:n { ##1 }
1469         }
1470       }
1471     }
1472   }
1473   \l_tmpa_tl
1474 }
1475
1476 \cs_new_protected:Nn \stex_invoke_module:n {
1477   \stex_debug:nn{modules}{Invoking~module~#1}
1478   \peek_charcode_remove:NTF ! {
1479     \__stex_modules_invoke_uri:nN { #1 }
1480   } {
1481     \peek_charcode_remove:NTF ? {
1482       \__stex_modules_invoke_symbol:nn { #1 }
1483     } {
1484       \msg_error:nnx{stex}{error/syntax}{
1485         ?~or~!~expected~after~
1486         \c_backslash_str STEXModule{#1}
1487       }
1488     }
1489   }
1490 }
1491
1492 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1493   \str_set:Nn #2 { #1 }
1494 }
1495
1496 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1497   \stex_invoke_symbol:n{#1?#2}
1498 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 29.)

`\stex_activate_module:n`

```
1499 \bool_new:N \l_stex_in_meta_bool
1500 \bool_set_false:N \l_stex_in_meta_bool
1501 \cs_new_protected:Nn \stex_activate_module:n {
1502   \stex_debug:nn{modules}{Activating~module~#1}
1503   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1504     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1505   }
1506   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1507     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1508     \use:c{ c_stex_module_#1_code }
1509   }
1510 }
```

(End definition for \stex_activate_module:n. This function is documented on page 30.)

```
1511 \</package>
```

Chapter 29

STEX -Module Inheritance Implementation

```
1512 <*package>
1513
1514 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1515
```

29.1 SMS Mode

```
1516 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1517 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1518 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1519 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1520
1521 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1522   \makeatletter
1523   \makeatother
1524   \ExplSyntaxOn
1525   \ExplSyntaxOff
1526   \rustexBREAK
1527 }
1528
1529 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1530   \symdef
1531   \importmodule
1532   \notation
1533   \symdecl
1534   \STEXexport
1535   \inlineass
1536   \inlinedef
1537   \inlineex
1538   \endinput
1539   \setnotation
```

```

1540 \copynotation
1541 }
1542
1543 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1544   \tl_to_str:n {
1545     smodule,
1546     copymodule,
1547     interpretmodule
1548     sdefinition,
1549     sexample,
1550     sassertion,
1551     sparagraph
1552   }
1553 }

```

(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 31.)

`\stex_if_smsmode_p:`

`\stex_if_smsmode:TF`

```

1554 \bool_new:N \g__stex_smsmode_bool
1555 \bool_set_false:N \g__stex_smsmode_bool
1556 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1557   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1558 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 31.)

`\stex_in_smsmode:nn`

```

1559 \cs_new_protected:Nn \stex_in_smsmode:nn {
1560   \vbox_set:Nn \l_tmpa_box {
1561     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1562     \bool_gset_true:N \g__stex_smsmode_bool
1563     #2
1564     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1565   }
1566   \box_clear:N \l_tmpa_box
1567 }
1568
1569 \quark_new:N \q__stex_smsmode_break
1570
1571 %\ior_new:N \c__stex_smsmode_ior
1572 %\tl_new:N \l__stex_smsmode_filecontent_tl
1573 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1574   % \tl_clear:N \l__stex_smsmode_filecontent_tl
1575   % \ior_open:Nn \c__stex_smsmode_ior {#1}
1576   % \ior_map_inline:Nn \c__stex_smsmode_ior {
1577     % \tl_put_right:Nn \l__stex_smsmode_filecontent_tl { ##1 }
1578   % }
1579   % \ior_close:N \c__stex_smsmode_ior
1580   \stex_filestack_push:n{#1}
1581   \stex_in_smsmode:nn{#1} {
1582     #2
1583     \everyeof{\q__stex_smsmode_break\noexpand}
1584     \expandafter\expandafter\expandafter
1585     \stex_smsmode_do:

```

```

1586 \csname @ @ input\endcsname "#1"\relax
1587 %\expandafter \stex_smsmode_do: \l__stex_smsmode_filecontent_tl
1588 }
1589 \stex_filestack_pop:
1590 }

```

(End definition for `\stex_in_smsmode:nn`. This function is documented on page 32.)

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

```

1591 \cs_new_protected:Npn \stex_smsmode_do: {
1592   \stex_if_smsmode:T {
1593     \__stex_smsmode_do:w
1594   }
1595 }
1596 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1597   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1598     \expandafter\if\expandafter\relax\noexpand#1
1599     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1600   } \else\expandafter\__stex_smsmode_do:w\fi
1601 }{
1602   \__stex_smsmode_do:w % #1
1603 }
1604 }
1605 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1606   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1607     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1608       #1\__stex_smsmode_do:w
1609     }{
1610       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1611         #1
1612       }{
1613         \cs_if_eq:NNTF \begin #1 {
1614           \__stex_smsmode_check_begin:n
1615         }{
1616           \cs_if_eq:NNTF \end #1 {
1617             \__stex_smsmode_check_end:n
1618           }{
1619             \__stex_smsmode_do:w
1620           }
1621         }
1622       }
1623     }
1624   }
1625 }
1626
1627 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1628   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1629     \begin{#1}
1630   }{
1631     \__stex_smsmode_do:w
1632   }
1633 }
1634 \cs_new_protected:Nn \__stex_smsmode_check_end:n {

```

```

1635 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1636   \end{#1}\__stex_smsmode_do:w
1637 }{
1638   \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1639 }
1640 }

```

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

29.2 Inheritance

```

1641 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1642 \cs_new_protected:Nn \stex_import_module_uri:nn {
1643   \str_set:Nx \l_stex_import_archive_str { #1 }
1644   \str_set:Nn \l_stex_import_path_str { #2 }
1645
1646   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1647   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1648   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1649
1650   \stex_modules_current_namespace:
1651   \bool_lazy_all:nTF {
1652     {\str_if_empty_p:N \l_stex_import_archive_str}
1653     {\str_if_empty_p:N \l_stex_import_path_str}
1654     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1655   }{
1656     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1657     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1658   }{
1659     \str_if_empty:NT \l_stex_import_archive_str {
1660       \prop_if_exist:NT \l_stex_current_repository_prop {
1661         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1662       }
1663     }
1664     \str_if_empty:NTF \l_stex_import_archive_str {
1665       \str_if_empty:NF \l_stex_import_path_str {
1666         \str_set:Nx \l_stex_import_ns_str {
1667           \l_stex_module_ns_str / \l_stex_import_path_str
1668         }
1669       }
1670     }{
1671       \stex_require_repository:n \l_stex_import_archive_str
1672       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1673       \l_stex_import_ns_str
1674       \str_if_empty:NF \l_stex_import_path_str {
1675         \str_set:Nx \l_stex_import_ns_str {
1676           \l_stex_import_ns_str / \l_stex_import_path_str
1677         }
1678       }
1679     }
1680   }
1681 }

```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 34.)

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str 1682 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1683 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1684 \str_new:N \l_stex_import_path_str
1685 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nmmn {<ns>} {<archive-ID>} {<path>} {<name>}
1686 \cs_new_protected:Nn \stex_import_require_module:nmmn {
1687   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1688
1689     % archive
1690     \str_set:Nx \l_tmpa_str { #2 }
1691     \str_if_empty:NTF \l_tmpa_str {
1692       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1693     } {
1694       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1695       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1696       \seq_put_right:Nn \l_tmpa_seq { source }
1697     }
1698
1699     % path
1700     \str_set:Nx \l_tmpb_str { #3 }
1701     \str_if_empty:NTF \l_tmpb_str {
1702       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1703
1704       \ltx@ifpackageloaded{babel} {
1705         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1706           { \languagename } \l_tmpb_str {
1707           \msg_error:nnx{stex}{error/unknownlanguage}{\languagename}
1708         }
1709       } {
1710         \str_clear:N \l_tmpb_str
1711       }
1712
1713       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1714       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1715         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1716       }{
1717         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1718         \IfFileExists{ \l_tmpa_str.tex }{
1719           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1720         }{
1721           % try english as default
1722           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1723           \IfFileExists{ \l_tmpa_str.en.tex }{
1724             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1725           }{
1726             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1727           }
1728         }

```

```

1729     }
1730
1731   } {
1732     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1733     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1734
1735     \ltx@ifpackageloaded{babel} {
1736       \exp_args:Nnx \prop_get:NnNF \c_stex_language_abbrevs_prop
1737         { \language } \l_tmpb_str {
1738         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1739       }
1740     } {
1741       \str_clear:N \l_tmpb_str
1742     }
1743
1744     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1745
1746     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1747     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1748       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1749     }{
1750       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1751       \IfFileExists{ \l_tmpa_str/#4.tex }{
1752         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1753       }{
1754         % try english as default
1755         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1756         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1757           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1758         }{
1759           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1760           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1761             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1762           }{
1763             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1764             \IfFileExists{ \l_tmpa_str.tex }{
1765               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1766             }{
1767               % try english as default
1768               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1769               \IfFileExists{ \l_tmpa_str.en.tex }{
1770                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1771               }{
1772                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1773               }
1774             }
1775           }
1776         }
1777       }
1778     }
1779   }
1780
1781   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1782     \seq_clear:N \l_stex_all_modules_seq

```



```

1783     \str_clear:N \l_stex_current_module_str
1784     \str_set:Nx \l_tmpb_str { #2 }
1785     \str_if_empty:NF \l_tmpb_str {
1786       \stex_set_current_repository:n { #2 }
1787     }
1788     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1789   }
1790
1791   \stex_if_module_exists:nF { #1 ? #4 } {
1792     \msg_error:nnx{stex}{error/unknownmodule}{
1793       #1?#4~(in~file~\g__stex_importmodule_file_str)
1794     }
1795   }
1796 }
1797 \stex_activate_module:n { #1 ? #4 }
1798 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 34.)

`\importmodule`

```

1799 \NewDocumentCommand \importmodule { 0{} m } {
1800   \stex_import_module_uri:nn { #1 } { #2 }
1801   \stex_debug:nn{modules}{Importing~module:~
1802     \l_stex_import_ns_str ? \l_stex_import_name_str
1803   }
1804   \stex_if_smsmode:F {
1805     \stex_import_require_module:nnnn
1806     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1807     { \l_stex_import_path_str } { \l_stex_import_name_str }
1808     \stex_annotate_invisible:nnn
1809     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1810   }
1811   \exp_args:Nx \stex_add_to_current_module:n {
1812     \stex_import_require_module:nnnn
1813     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1814     { \l_stex_import_path_str } { \l_stex_import_name_str }
1815   }
1816   \exp_args:Nx \stex_add_import_to_current_module:n {
1817     \l_stex_import_ns_str ? \l_stex_import_name_str
1818   }
1819   \stex_smsmode_do:
1820 }
1821 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

(End definition for `\importmodule`. This function is documented on page 32.)

`\usemodule`

```

1822 \NewDocumentCommand \usemodule { 0{} m } {
1823   \stex_if_smsmode:F {
1824     \stex_import_module_uri:nn { #1 } { #2 }
1825     \stex_import_require_module:nnnn
1826     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1827     { \l_stex_import_path_str } { \l_stex_import_name_str }
1828     \stex_annotate_invisible:nnn
1829     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}

```

```

1830 }
1831 \stex_smsmode_do:
1832 }

(End definition for \usemodule. This function is documented on page 32.)

1833 </package>

```

Chapter 30

STEX -Symbols Implementation

```
1834 <*package>
1835
1836 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1837
```

Warnings and error messages

```
1838
```

30.1 Symbol Declarations

```
1839 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1840 \seq_new:N \l_stex_all_symbols_seq
```

(End definition for \l_stex_all_symbols_seq. This variable is documented on page 36.)

`\STEXsymbol`

```
1841 \NewDocumentCommand \STEXsymbol { m } {
1842   \stex_get_symbol:n { #1 }
1843   \exp_args:No
1844   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1845 }
```

(End definition for \STEXsymbol. This function is documented on page 38.)

symdecl arguments:

```
1846 \keys_define:nn { stex / symdecl } {
1847   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1848   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1849   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1850   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1851   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1852   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1853   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1854   def       .tl_set:N = \l_stex_symdecl_definiens_tl
1855 }
```

```

1856
1857 \bool_new:N \l_stex_symdecl_make_macro_bool
1858
1859 \cs_new_protected:Nn \__stex_symdecl_args:n {
1860   \str_clear:N \l_stex_symdecl_name_str
1861   \str_clear:N \l_stex_symdecl_args_str
1862   \bool_set_false:N \l_stex_symdecl_local_bool
1863   \tl_clear:N \l_stex_symdecl_type_tl
1864   \tl_clear:N \l_stex_symdecl_definiens_tl
1865
1866   \keys_set:nn { stex / symdecl } { #1 }
1867 }

```

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

```

1868
1869 \NewDocumentCommand \symdecl { s O{} m } {
1870   \__stex_symdecl_args:n { #2 }
1871   \IfBooleanTF #1 {
1872     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1873   } {
1874     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1875   }
1876   \stex_symdecl_do:n { #3 }
1877   \stex_smsmode_do:
1878 }
1879 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for `\symdecl`. This function is documented on page 35.)

\stex_symdecl_do:n

```

1880 \cs_new_protected:Nn \stex_symdecl_do:n {
1881   \stex_if_in_module:F {
1882     % TODO throw error? some default namespace?
1883   }
1884
1885   \str_if_empty:NT \l_stex_symdecl_name_str {
1886     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1887   }
1888
1889   \prop_if_exist:cT { l_stex_symdecl_
1890     \l_stex_current_module_str ?
1891     \l_stex_symdecl_name_str
1892   }_prop
1893 }{
1894   % TODO throw error (beware of circular dependencies)
1895 }
1896
1897 \prop_clear:N \l_tmpa_prop
1898 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1899 \seq_clear:N \l_tmpa_seq
1900 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1901 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1902

```

```

1903 \exp_args:No \stex_add_constant_to_current_module:n {
1904   \l_stex_symdecl_name_str
1905 }
1906
1907 % arity/args
1908 \int_zero:N \l_tmpb_int
1909
1910 \bool_set_true:N \l_tmpa_bool
1911 \str_map_inline:Nn \l_stex_symdecl_args_str {
1912   \token_case_meaning:NnF ##1 {
1913     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1914     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1915     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1916     {\tl_to_str:n a} {
1917       \bool_set_false:N \l_tmpa_bool
1918       \int_incr:N \l_tmpb_int
1919     }
1920     {\tl_to_str:n B} {
1921       \bool_set_false:N \l_tmpa_bool
1922       \int_incr:N \l_tmpb_int
1923     }
1924   }{
1925     \msg_set:nnn{stex}{error/wrongargs}{
1926       args~value~in~symbol~declaration~for~
1927       \l_stex_current_module_str ?
1928       \l_stex_symdecl_name_str ~
1929       needs~to~be~
1930       i,~a,~b~or~B,~but~##1~given
1931     }
1932     \msg_error:nn{stex}{error/wrongargs}
1933   }
1934 }
1935 \bool_if:NTF \l_tmpa_bool {
1936   % possibly numeric
1937   \str_if_empty:NTF \l_stex_symdecl_args_str {
1938     \prop_put:Nnn \l_tmpa_prop { args } {}
1939     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1940   }{
1941     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1942     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1943     \str_clear:N \l_tmpa_str
1944     \int_step_inline:nn \l_tmpa_int {
1945       \str_put_right:Nn \l_tmpa_str i
1946     }
1947     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1948   }
1949 } {
1950   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1951   \prop_put:Nnx \l_tmpa_prop { arity }
1952     { \str_count:N \l_stex_symdecl_args_str }
1953 }
1954 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1955
1956

```

```

1957 % semantic macro
1958
1959 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1960   \exp_args:Nx \stex_do_aftergroup:n {
1961     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1962       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1963     }}
1964   }
1965
1966   \bool_if:NF \l_stex_symdecl_local_bool {
1967     \exp_args:Nx \stex_add_to_current_module:n {
1968       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1969         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1970       } }
1971     }
1972   }
1973 }
1974
1975 % add to all symbols
1976
1977 \bool_if:NF \l_stex_symdecl_local_bool {
1978   \exp_args:Nx \stex_add_to_current_module:n {
1979     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1980       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1981     }
1982   }
1983 %   \exp_args:Nx \stex_add_field_to_current_module:n {
1984 %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1985 %   }
1986 }
1987
1988 \stex_debug:nn{symbols}{New~symbol:~
1989   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1990   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1991   Args:~\prop_item:Nn \l_tmpa_prop { args }
1992 }
1993
1994 % circular dependencies require this:
1995
1996 \prop_if_exist:cF {
1997   l_stex_symdecl_
1998   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1999   _prop
2000 } {
2001   \prop_set_eq:cN {
2002     l_stex_symdecl_
2003     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2004     _prop
2005   } \l_tmpa_prop
2006 }
2007
2008 \seq_clear:c {
2009   l_stex_symdecl_
2010   \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2011   _notations
2012 }
2013
2014 \bool_if:NF \l_stex_symdecl_local_bool {
2015   \exp_args:Nx
2016   \stex_add_to_current_module:n {
2017     \seq_clear:c {
2018       l_stex_symdecl_
2019       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2020       _notations
2021     }
2022     \prop_set_from_keyval:cn {
2023       l_stex_symdecl_
2024       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2025       _prop
2026     } {
2027       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2028       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2029       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2030       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2031       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2032       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2033     }
2034   }
2035 }
2036
2037 \stex_if_smsmode:TF {
2038   \bool_if:NF \l_stex_symdecl_local_bool {
2039     % \exp_args:Nx \stex_add_to_sms:n {
2040     %   \prop_set_from_keyval:cn {
2041     %     l_stex_symdecl_
2042     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2043     %     _prop
2044     %   } {
2045     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2046     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2047     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
2048     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2049     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2050     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2051     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2052     %   }
2053     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2054     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2055     %   }
2056     % }
2057   }
2058 }{
2059   \exp_args:Nx \stex_do_aftergroup:n {
2060     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2061       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2062     }
2063   }
2064   \stex_if_do_html:T {

```

```

2065 \stex_annotate_invisible:nnn {symdecl} {
2066   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2067 } {
2068   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2069   \stex_annotate_invisible:nnn{args}{}{
2070     \prop_item:Nn \l_tmpa_prop { args }
2071   }
2072   \stex_annotate_invisible:nnn{macroname}{#1}{}
2073   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2074     \stex_annotate_invisible:nnn{definiens}{}
2075     { $\l_stex_symdecl_definiens_tl$ }
2076   }
2077 }
2078 }
2079 }
2080 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 36.)

`\stex_get_symbol:n`

```

2081 \str_new:N \l_stex_get_symbol_uri_str
2082
2083 \cs_new_protected:Nn \stex_get_symbol:n {
2084   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2085     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2086   }{
2087     % argument is a string
2088     % is it a command name?
2089     \cs_if_exist:cTF { #1 }{
2090       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2091       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2092       \str_if_empty:NNTF \l_tmpa_str {
2093         \exp_args:Nx \cs_if_eq:NNTF {
2094           \tl_head:N \l_tmpa_tl
2095         } \stex_invoke_symbol:n {
2096           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2097         }{
2098           \__stex_symdecl_get_symbol_from_string:n { #1 }
2099         }
2100       } {
2101         \__stex_symdecl_get_symbol_from_string:n { #1 }
2102       }
2103     }{
2104       % argument is not a command name
2105       \__stex_symdecl_get_symbol_from_string:n { #1 }
2106       % \l_stex_all_symbols_seq
2107     }
2108   }
2109 }
2110
2111 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2112   \str_set:Nn \l_tmpa_str { #1 }
2113   \bool_set_false:N \l_tmpa_bool
2114   \stex_if_in_module:T {

```



```

2115 \exp_args:Nno \seq_if_in:cnT {c_stex_module_\l_stex_current_module_str _constants} { \l_
2116 \bool_set_true:N \l_tmpa_bool
2117 \str_set:Nx \l_stex_get_symbol_uri_str {
2118 \l_stex_current_module_str ? #1
2119 }
2120 }
2121 }
2122 \bool_if:NF \l_tmpa_bool {
2123 \tl_set:Nn \l_tmpa_tl {
2124 \msg_set:nnn{stex}{error/unknownsymbol}{
2125 No~symbol~#1~found!
2126 }
2127 \msg_error:nn{stex}{error/unknownsymbol}
2128 }
2129 \str_set:Nn \l_tmpa_str { #1 }
2130 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2131 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2132 \str_set:Nn \l_tmpb_str { ##1 }
2133 \str_if_eq:eeT { \l_tmpa_str } {
2134 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2135 } {
2136 \seq_map_break:n {
2137 \tl_set:Nn \l_tmpa_tl {
2138 \str_set:Nn \l_stex_get_symbol_uri_str {
2139 ##1
2140 }
2141 }
2142 }
2143 }
2144 }
2145 \l_tmpa_tl
2146 }
2147 }
2148
2149 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2150 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2151 { \tl_tail:N \l_tmpa_tl }
2152 \tl_if_single:NTF \l_tmpa_tl {
2153 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2154 \exp_after:wN \str_set:Nn \exp_after:wN
2155 \l_stex_get_symbol_uri_str \l_tmpa_tl
2156 }{
2157 % TODO
2158 % tail is not a single group
2159 }
2160 }{
2161 % TODO
2162 % tail is not a single group
2163 }
2164 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 36.)

30.2 Notations

```

2165 <@@=stex_notation>

      notation arguments:
2166 \keys_define:nn { stex / notation } {
2167   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2168   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2169   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2170   op        .tl_set:N   = \l__stex_notation_op_tl ,
2171   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2172   primary   .default:n  = {true} ,
2173   unknown   .code:n     = \str_set:Nx
2174             \l__stex_notation_variant_str \l_keys_key_str
2175 }
2176
2177 \cs_new_protected:Nn \stex_notation_args:n {
2178   \str_clear:N \l__stex_notation_lang_str
2179   \str_clear:N \l__stex_notation_variant_str
2180   \str_clear:N \l__stex_notation_prec_str
2181   \tl_clear:N \l__stex_notation_op_tl
2182   \bool_set_false:N \l__stex_notation_primary_bool
2183
2184   \keys_set:nn { stex / notation } { #1 }
2185 }

```

\notation

```

2186 \NewDocumentCommand \notation { 0{ } m } {
2187   \stex_notation_args:n { #1 }
2188   \tl_clear:N \l_stex_symdecl_definiens_tl
2189   \stex_get_symbol:n { #2 }
2190   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2191 }
2192 \stex_deactivate_macro:Nn \notation {module-environments}

```

(End definition for \notation. This function is documented on page 36.)

\stex_notation_do:nn

```

2193 \seq_new:N \l__stex_notation_precedences_seq
2194 \tl_new:N \l__stex_notation_opprec_tl
2195 \int_new:N \l__stex_notation_currarg_int
2196
2197 \cs_new_protected:Nn \stex_notation_do:nn {
2198   \let\l_stex_current_symbol_str\relax
2199   \str_set:Nx \l__stex_notation_symbol_str { #1 }
2200   \seq_clear:N \l__stex_notation_precedences_seq
2201   \tl_clear:N \l__stex_notation_opprec_tl
2202   \prop_get:cnN {
2203     l_stex_symdecl_ #1 _prop
2204   } { args } \l__stex_notation_args_str
2205
2206   % precedences
2207   \prop_get:cnN {
2208     l_stex_symdecl_ #1 _prop
2209   } { arity } \l__stex_notation_arity_str

```

```

2210 \str_if_empty:NTF \l__stex_notation_prec_str {
2211   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2212     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2213   }{
2214     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2215   }
2216 } {
2217   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2218     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2219     \int_step_inline:nn { \l__stex_notation_arity_str } {
2220       \exp_args:NNo
2221       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2222     }
2223   }{
2224     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2225     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2226       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2227       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2228         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2229         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2230         \seq_map_inline:Nn \l_tmpa_seq {
2231           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2232         }
2233       }
2234     }{
2235       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2236         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2237       }{
2238         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2239       }
2240     }
2241   }
2242 }
2243
2244 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2245 \int_step_inline:nn { \l__stex_notation_arity_str } {
2246   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2247     \exp_args:NNo
2248     \seq_put_right:No \l__stex_notation_precedences_seq {
2249       \l__stex_notation_opprec_tl
2250     }
2251   }
2252 }
2253
2254 \tl_clear:N \l__stex_notation_dummyargs_tl
2255
2256 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2257   \exp_args:NNe
2258   \cs_set:Npn \l__stex_notation_macrocode_cs {
2259     \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2260     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2261     { \l__stex_notation_opprec_tl }
2262     { \exp_not:n { #2 } }
2263   }

```

```

2264   \_stex_notation_final:
2265 }{
2266   \str_if_in:NnTF \l__stex_notation_args_str b {
2267     \exp_args:Nne \use:nn
2268     {
2269       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2270       \cs_set:Npn \l__stex_notation_arity_str } { {
2271         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2272         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2273         { \l__stex_notation_opprec_tl }
2274         { \exp_not:n { #2 } }
2275       }}
2276 }{
2277   \str_if_in:NnTF \l__stex_notation_args_str B {
2278     \exp_args:Nne \use:nn
2279     {
2280       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2281       \cs_set:Npn \l__stex_notation_arity_str } { {
2282         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2283         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2284         { \l__stex_notation_opprec_tl }
2285         { \exp_not:n { #2 } }
2286       } }
2287 }{
2288   \exp_args:Nne \use:nn
2289   {
2290     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2291     \cs_set:Npn \l__stex_notation_arity_str } { {
2292       \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2293       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2294       { \l__stex_notation_opprec_tl }
2295       { \exp_not:n { #2 } }
2296     } }
2297   }
2298 }
2299
2300 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2301 \int_zero:N \l__stex_notation_currarg_int
2302 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2303 \_stex_notation_arguments:
2304 }
2305 }

```

(End definition for `\stex_notation_do:nn`. This function is documented on page 37.)

`_stex_notation_arguments:` Takes care of annotating the arguments in a notation macro

```

2306 \cs_new_protected:Nn \_stex_notation_arguments: {
2307   \int_incr:N \l__stex_notation_currarg_int
2308   \str_if_empty:NnTF \l__stex_notation_remaining_args_str {
2309     \_stex_notation_final:
2310   }{
2311     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2312     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2313     \str_if_eq:VnTF \l_tmpa_str a {

```

```

2314     \_stex_notation_argument_assoc:n
2315   }{
2316     \str_if_eq:VnTF \l_tmpa_str B {
2317       \_stex_notation_argument_assoc:n
2318     }{
2319       \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2320       \tl_put_right:Nx \l__stex_notation_dummyargs_tl {
2321         { \_stex_term_math_arg:nnn
2322           { \int_use:N \l__stex_notation_currarg_int }
2323           { \l_tmpa_str }
2324           { #####\int_use:N \l__stex_notation_currarg_int }
2325         }
2326       }
2327       \_stex_notation_arguments:
2328     }
2329   }
2330 }
2331 }

```

(End definition for _stex_notation_arguments:.)

_stex_notation_argument_assoc:n

```

2332 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2333
2334   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2335     {\l__stex_notation_arity_str}{
2336       #1
2337     }
2338   \int_zero:N \l_tmpa_int
2339   \tl_clear:N \l_tmpa_tl
2340   \str_map_inline:Nn \l__stex_notation_args_str {
2341     \int_incr:N \l_tmpa_int
2342     \tl_put_right:Nx \l_tmpa_tl {
2343       \str_if_eq:nnTF {##1}{a}{ {} }{
2344         \str_if_eq:nnTF {##1}{B}{ {} }{
2345           {##### \int_use:N \l_tmpa_int}
2346         }
2347       }
2348     }
2349   }
2350   \exp_after:wN\exp_after:wN\exp_after:wN \def
2351   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2352   \exp_after:wN\exp_after:wN\exp_after:wN ##
2353   \exp_after:wN\exp_after:wN\exp_after:wN 1
2354   \exp_after:wN\exp_after:wN\exp_after:wN ##
2355   \exp_after:wN\exp_after:wN\exp_after:wN 2
2356   \exp_after:wN\exp_after:wN\exp_after:wN {
2357     \exp_after:wN \exp_after:wN \exp_after:wN
2358     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2359       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2360     }
2361   }
2362
2363   \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str

```

```

2364 \tl_put_right:Nx \l__stex_notation_dummyargs_tl { {
2365   \stex_term_math_assoc_arg:nnnn
2366   { \int_use:N \l__stex_notation_currarg_int }
2367   { \l_tmpa_str }
2368   { ####\int_use:N \l__stex_notation_currarg_int }
2369   { \l_tmpa_cs {####1} {####2} }
2370 } }
2371 %\cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2372 %\tl_put_right:Nx \l_tmpa_tl {
2373 % { \stex_term_math_assoc_arg:nnnn
2374 % { \int_use:N \l_tmpa_int }
2375 % { \l_tmpb_str }
2376 % \exp_args:No \exp_not:n
2377 % {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2378 % { ####\int_use:N \l_tmpa_int }
2379 % }
2380 %}
2381 \__stex_notation_arguments:
2382 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2383 \cs_new_protected:Nn \__stex_notation_final: {
2384   \exp_args:Nne \use:nn
2385   {
2386     \cs_generate_from_arg_count:cNnn {
2387       stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2388       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2389       _cs
2390     }
2391     \cs_set:Npn \l__stex_notation_arity_str { { {
2392       \exp_after:wN \exp_after:wN \exp_after:wN
2393       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2394       { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2395     } }
2396
2397     \tl_if_empty:NF \l__stex_notation_op_tl {
2398       \cs_set:cpx {
2399         stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2400         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2401         _cs
2402       } {
2403         \stex_term_oms:nnn {
2404           \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2405           \l__stex_notation_lang_str
2406         }{
2407           \l__stex_notation_symbol_str
2408         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2409       }
2410     }
2411
2412     \exp_args:Ne
2413     \stex_add_to_current_module:n {

```

```

2414 \cs_generate_from_arg_count:cNnn {
2415   stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2416   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2417   _cs
2418 } \cs_set:Npn {\l__stex_notation_arity_str} {
2419   \exp_after:wN \exp_after:wN \exp_after:wN
2420   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2421   { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2422 }
2423 \tl_if_empty:NF \l__stex_notation_op_tl {
2424   \cs_set:cpn {
2425     stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2426     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2427     _cs
2428   } {
2429     \stex_term_oms:nnn {
2430       \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2431       \l__stex_notation_lang_str
2432     }{
2433       \l__stex_notation_symbol_str
2434     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2435   }
2436 }
2437 }
2438 \exp_args:Nx
2439 % \stex_do_aftergroup:n {
2440   \seq_put_right:cx {
2441     l_stex_symdecl_ \l__stex_notation_symbol_str
2442     _notations
2443   } {
2444     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2445   }
2446 % }
2447
2448 \stex_debug:nn{symbols}{
2449   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2450   ~for~\l__stex_notation_symbol_str^^J
2451   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2452   Argument~precedences:~
2453   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2454   Notation: \cs_meaning:c {
2455     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2456     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2457     _cs
2458   }
2459 }
2460
2461 %\prop_set_eq:cN {
2462 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2463 %   \c_hash_str \l__stex_notation_lang_str _prop
2464 %} \l_tmpb_prop
2465
2466 \exp_args:Ne
2467 \stex_add_to_current_module:n {

```

```

2468 \seq_put_right:cn {
2469   l_stex_symdecl_ \l__stex_notation_symbol_str
2470   _notations
2471 } {
2472   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2473 }
2474 %\prop_set_from_keyval:cn {
2475 % l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2476 %   \c_hash_str \l__stex_notation_lang_str _prop
2477 %} {
2478 % symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2479 % language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2480 % variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2481 % opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2482 % argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2483 %}
2484 }
2485
2486 \stex_if_smsmode:TF {
2487 %   \exp_args:Nx \stex_add_to_sms:n {
2488 %     \prop_set_from_keyval:cn {
2489 %       l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2490 %       \c_hash_str \l__stex_notation_lang_str _prop
2491 %     } {
2492 %       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2493 %       language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2494 %       variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2495 %       opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2496 %       argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2497 %     }
2498 %   }
2499 %} {
2500
2501 % HTML annotations
2502 \stex_if_do_html:T {
2503   \stex_annotate_invisible:nnn { notation }
2504   { \l__stex_notation_symbol_str } {
2505     \stex_annotate_invisible:nnn { notationfragment }
2506     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str } {}
2507     \stex_annotate_invisible:nnn { precedence }
2508     { \l__stex_notation_prec_str } {}
2509
2510     \int_zero:N \l_tmpa_int
2511     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2512     \tl_clear:N \l_tmpa_tl
2513     \int_step_inline:nn { \l__stex_notation_arity_str } {
2514       \int_incr:N \l_tmpa_int
2515       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2516       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2517       \str_if_eq:VnTF \l_tmpb_str a {
2518         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2519           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2520           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2521         } }

```



```

2522     }{
2523       \str_if_eq:VnTF \l_tmpb_str B {
2524         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2525           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2526           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2527         } }
2528       }{
2529         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2530           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2531         } }
2532       }
2533     }
2534   }
2535   \stex_annotate_invisible:nnn { notationcomp }{}{
2536     \str_set:Nx \l_stex_current_symbol_str { \l__stex_notation_symbol_str }
2537     $ \exp_args:Nno \use:nn { \use:c {
2538       stex_notation_ \l_stex_current_symbol_str
2539       \c_hash_str \l__stex_notation_variant_str
2540       \c_hash_str \l__stex_notation_lang_str _cs
2541     } } { \l_tmpa_tl } $
2542   }
2543 }
2544 }
2545 }
2546 \stex_smsmode_do:
2547 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2548 \keys_define:nn { stex / setnotation } {
2549   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2550   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2551   unknown .code:n = \str_set:Nx
2552     \l__stex_notation_variant_str \l_keys_key_str
2553 }
2554
2555 \cs_new_protected:Nn \_stex_setnotation_args:n {
2556   \str_clear:N \l__stex_notation_lang_str
2557   \str_clear:N \l__stex_notation_variant_str
2558   \keys_set:nn { stex / setnotation } { #1 }
2559 }
2560
2561 \NewDocumentCommand \setnotation {m m} {
2562   \stex_get_symbol:n { #1 }
2563   \_stex_setnotation_args:n { #2 }
2564   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl \l_stex_get_symbol_uri_str _notations }
2565     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2566     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2567       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2568     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notation
2569       { \c_hash_str }
2570     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl \l_stex_get_symbol_uri_str _notations
2571       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }

```

```

2572 \exp_args:Nx \stex_add_to_current_module:n {
2573   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notati
2574     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2575   \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notation
2576     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2577   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _notati
2578     { \c_hash_str }
2579 }
2580 \stex_debug:nn {notations}{
2581   Setting~default~notation~
2582   {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2583   \l_stex_get_symbol_uri_str \
2584   \expandafter\meaning\csname
2585   l_stex_symdecl_\l_stex_get_symbol_uri_str _notations\endcsname
2586 }
2587 }{
2588   % todo throw error
2589 }
2590 \stex_smsmode_do:
2591 }
2592
2593 \cs_new_protected:Nn \stex_copy_notations:nn {
2594   \stex_debug:nn {notations}{
2595     Copying~notations~from~#2~to~#1\
2596     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2597   }
2598   \tl_clear:N \l_tmpa_tl
2599   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2600     \tl_put_right:Nn \l_tmpa_tl { {## #1} }
2601   }
2602   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2603     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2604     \edef \l_tmpa_tl {
2605       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2606       \exp_after:wN\exp_after:wN\exp_after:wN {
2607         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2608       }
2609     }
2610     \exp_args:Nx
2611     \stex_do_aftergroup:n {
2612       \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2613       \cs_generate_from_arg_count:cNnn {
2614         stex_notation_ #1 \c_hash_str ##1 _cs
2615       } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2616         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2617       }
2618     }
2619   }
2620 }
2621
2622 \NewDocumentCommand \copynotation {m m} {
2623   \stex_get_symbol:n { #1 }
2624   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2625   \stex_get_symbol:n { #2 }

```

```

2626 \exp_args:Noo
2627 \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2628 \exp_args:Nx \stex_add_import_to_current_module:n{
2629   \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2630 }
2631 \stex_smsmode_do:
2632 }
2633

```

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

\symdef

```

2634 \keys_define:nn { stex / symdef } {
2635   name .str_set_x:N = \l_stex_symdecl_name_str ,
2636   local .bool_set:N = \l_stex_symdecl_local_bool ,
2637   args .str_set_x:N = \l_stex_symdecl_args_str ,
2638   type .tl_set:N = \l_stex_symdecl_type_tl ,
2639   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2640   op .tl_set:N = \l__stex_notation_op_tl ,
2641   lang .str_set_x:N = \l__stex_notation_lang_str ,
2642   variant .str_set_x:N = \l__stex_notation_variant_str ,
2643   prec .str_set_x:N = \l__stex_notation_prec_str ,
2644   unknown .code:n = \str_set:Nx
2645     \l__stex_notation_variant_str \l_keys_key_str
2646 }
2647
2648 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2649   \str_clear:N \l_stex_symdecl_name_str
2650   \str_clear:N \l_stex_symdecl_args_str
2651   \bool_set_false:N \l_stex_symdecl_local_bool
2652   \tl_clear:N \l_stex_symdecl_type_tl
2653   \tl_clear:N \l_stex_symdecl_definiens_tl
2654   \str_clear:N \l__stex_notation_lang_str
2655   \str_clear:N \l__stex_notation_variant_str
2656   \str_clear:N \l__stex_notation_prec_str
2657   \tl_clear:N \l__stex_notation_op_tl
2658
2659   \keys_set:nn { stex / symdef } { #1 }
2660 }
2661
2662 \NewDocumentCommand \symdef { 0{} m } {
2663   \__stex_notation_symdef_args:n { #1 }
2664   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2665   \stex_symdecl_do:n { #2 }
2666   \exp_args:Nx \stex_notation_do:nn {
2667     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2668   }
2669 }
2670 \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for \symdef. This function is documented on page 37.)

```

2671 \endpackage

```

Chapter 31

STEX -Terms Implementation

```
2672 <*package>
2673
2674 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2675
2676 <@@=stex_terms>
2677
2678 Warnings and error messages
2679 \msg_new:nnn{stex}{error/nonotation}{
2680   Symbol~#1~invoked,~but~has~no~notation~#2!
2681 }
2682 \msg_new:nnn{stex}{error/notationarg}{
2683   Error~in~parsing~notation~#1
2684 }
2685 \msg_new:nnn{stex}{error/noop}{
2686   Symbol~#1~has~no~operator~notation~for~notation~#2
2687 }
```

31.1 Symbol Invocations

Arguments:

```
2687 \keys_define:nn { stex / terms } {
2688   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2689   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2690   unknown .code:n = \str_set:Nx
2691     \l__stex_terms_variant_str \l_keys_key_str
2692 }
2693
2694 \cs_new_protected:Nn \__stex_terms_args:n {
2695   \str_clear:N \l__stex_terms_lang_str
2696   \str_clear:N \l__stex_terms_variant_str
2697   \str_clear:N \l__stex_terms_prec_str
2698   \tl_clear:N \l__stex_terms_op_tl
2699
2700   \keys_set:nn { stex / terms } { #1 }
```

2701 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2702 \cs_new_protected:Nn \stex_invoke_symbol:n {
2703   \if_mode_math:
2704     \exp_after:wN \__stex_terms_invoke_math:n
2705   \else:
2706     \exp_after:wN \__stex_terms_invoke_text:n
2707   \fi: { #1 }
2708 }
```

(End definition for \stex_invoke_symbol:n. This function is documented on page 38.)

__stex_terms_invoke_math:n

```
2709 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2710   \peek_charcode_remove:NTF ! {
2711     \peek_charcode:NTF [ {
2712       \__stex_terms_invoke_op:nw { #1 }
2713     }{
2714       \peek_charcode_remove:NTF ! {
2715         \peek_charcode:NTF [ {
2716           \__stex_terms_invoke_op_custom:nw
2717         }{
2718           % TODO throw error
2719         }
2720       }{
2721         \__stex_terms_invoke_op:nw { #1 } []
2722       }
2723     }
2724   }{
2725     \peek_charcode_remove:NTF * {
2726       \__stex_terms_invoke_text:n { #1 }
2727     }{
2728       \peek_charcode:NTF [ {
2729         \__stex_terms_invoke_math:nw { #1 }
2730       }{
2731         \__stex_terms_invoke_math:nw { #1 } []
2732       }
2733     }
2734   }
2735 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2736 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2737   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2738     \stex_highlight_term:nn{#1}{#2}
2739   }
2740 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2741 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2742   \_stex_terms_args:n { #2 }
2743   \cs_if_exist:cTF {
2744     stex_op_notation_ #1 \c_hash_str
2745     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2746   }{
2747     \csname stex_op_notation_ #1 \c_hash_str
2748       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2749     \endcsname
2750   }{
2751     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2752   }
2753 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2754 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2755   \_stex_terms_args:n { #2 }
2756   \seq_if_empty:cTF {
2757     l_stex_symdecl_ #1 _notations
2758   } {
2759     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2760   } {
2761     \seq_if_in:cxTF {
2762       l_stex_symdecl_ #1 _notations
2763     }
2764     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2765       \str_set:Nn \l_stex_current_symbol_str { #1 }
2766       \stex_debug:nn{terms}{Using~
2767         #1\c_hash_str\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str \\
2768         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2769         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2770         _cs\endcsname
2771       }
2772       \use:c{
2773         stex_notation_ #1 \c_hash_str
2774         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2775         _cs
2776       }
2777     }{
2778       \str_if_empty:NTF \l__stex_terms_variant_str {
2779         \str_if_empty:NTF \l__stex_terms_lang_str {
2780           \seq_get_left:cN {
2781             l_stex_symdecl_ #1 _notations
2782           } \l_tmpa_str
2783           \str_set:Nn \l_stex_current_symbol_str { #1 }
2784           \stex_debug:nn{terms}{Using~
2785             #1\c_hash_str\l_tmpa_str \\
2786             \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2787             \l_tmpa_str
2788             _cs\endcsname
2789           }

```

```

2790         \use:c{
2791             stex_notation_ #1 \c_hash_str \l_tmpa_str
2792             _cs
2793         }
2794     }{
2795         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2796             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2797         }
2798     }
2799 }{
2800     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2801         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2802     }
2803 }
2804 }
2805 }
2806 }

```

(End definition for `__stex_terms_invoke_math:nw`.)

`__stex_terms_invoke_text:n`

```

2807 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2808     \peek_charcode_remove:NTF ! {
2809         \stex_term_custom:nn { #1 } { }
2810     }{
2811         \prop_set_eq:Nc \l_tmpa_prop {
2812             l_stex_symdecl_ #1 _prop
2813         }
2814         \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2815         \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2816     }
2817 }

```

(End definition for `__stex_terms_invoke_text:n`.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2818 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2819 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2820 \int_new:N \l__stex_terms_downprec
2821 \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 39.)

Bracketing:

```

\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
2822 \tl_set:Nn \l__stex_terms_left_bracket_str (
2823 \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

```
2824 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2825   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2826     \bool_set_false:N \l__stex_terms_brackets_done_bool
2827     #2
2828   } {
2829     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2830       \bool_if:NTF \l_stex_inarray_bool { #2 } {
2831         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2832         \dobrackets { #2 }
2833       }
2834     }{ #2 }
2835   }
2836 }
```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```
2837 \bool_new:N \l__stex_terms_brackets_done_bool
2838 %\RequirePackage{scalerel}
2839 \cs_new_protected:Npn \dobrackets #1 {
2840   %\ThisStyle{\if D\m@switch
2841   %   \exp_args:Nnx \use:nn
2842   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2843   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2844   % \else
2845   \exp_args:Nnx \use:nn
2846   {
2847     \bool_set_true:N \l__stex_terms_brackets_done_bool
2848     \int_set:Nn \l__stex_terms_downprec \infprec
2849     \l__stex_terms_left_bracket_str
2850     #1
2851   }
2852   {
2853     \bool_set_false:N \l__stex_terms_brackets_done_bool
2854     \l__stex_terms_right_bracket_str
2855     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2856   }
2857   %\fi}
2858 }
```

(End definition for `\dobrackets`. This function is documented on page 39.)

`\withbrackets`

```
2859 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2860   \exp_args:Nnx \use:nn
2861   {
2862     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2863     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2864     #3
2865   }
2866   {
2867     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2868     {\l__stex_terms_left_bracket_str}
```



```

2869 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2870 {\l__stex_terms_right_bracket_str}
2871 }
2872 }

```

(End definition for `\withbrackets`. This function is documented on page 39.)

`\STEXinvisible`

```

2873 \cs_new_protected:Npn \STEXinvisible #1 {
2874 \stex_annotate_invisible:n { #1 }
2875 }

```

(End definition for `\STEXinvisible`. This function is documented on page 40.)

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2876 \cs_new_protected:Nn \_stex_term_oms:nnn {
2877 \stex_annotate:nnn{ OMID }{ #2 }{
2878 \stex_highlight_term:nn { #1 } { #3 }
2879 }
2880 }
2881
2882 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2883 \__stex_terms_maybe_brackets:nn { #3 }{
2884 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2885 }
2886 }

```

(End definition for `_stex_term_math_oms:nnnn`. This function is documented on page 38.)

`_stex_term_math_oma:nnnn`

```

2887 \cs_new_protected:Nn \_stex_term_oma:nnn {
2888 \stex_annotate:nnn{ OMA }{ #2 }{
2889 \stex_highlight_term:nn { #1 } { #3 }
2890 }
2891 }
2892
2893 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2894 \__stex_terms_maybe_brackets:nn { #3 }{
2895 \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2896 }
2897 }

```

(End definition for `_stex_term_math_oma:nnnn`. This function is documented on page 38.)

`_stex_term_math_omb:nnnn`

```

2898 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2899 \stex_annotate:nnn{ OMBIND }{ #2 }{
2900 \stex_highlight_term:nn { #1 } { #3 }
2901 }
2902 }
2903
2904 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2905 \__stex_terms_maybe_brackets:nn { #3 }{
2906 \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```

```

2907 }
2908 }

```

(End definition for `_stex_term_math_omb:nnnn`. This function is documented on page 38.)

`_stex_term_math_arg:nnn`

```

2909 \cs_new_protected:Nn \_stex_term_arg:nn {
2910   \stex_unhighlight_term:n {
2911     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2912   }
2913 }
2914 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2915   \exp_args:Nnx \use:nn
2916     { \int_set:Nn \l__stex_terms_downprec { #2 }
2917       \_stex_term_arg:nn { #1 }{ #3 }
2918     }
2919   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2920 }

```

(End definition for `_stex_term_math_arg:nnn`. This function is documented on page 38.)

`_stex_term_math_assoc_arg:nnnn`

```

2921 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2922   % TODO sequences
2923   \clist_set:Nn \l_tmpa_clist{ #3 }
2924   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2925     \tl_set:Nn \l_tmpa_tl { #3 }
2926   }{
2927     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
2928     \clist_reverse:N \l_tmpa_clist
2929     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2930
2931     \clist_map_inline:Nn \l_tmpa_clist {
2932       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2933         \exp_args:Nno
2934         \l_tmpa_cs { ##1 } \l_tmpa_tl
2935       }
2936     }
2937   }
2938   \exp_args:Nnno
2939   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2940 }

```

(End definition for `_stex_term_math_assoc_arg:nnnn`. This function is documented on page 38.)

`\stex_term_custom:nn`

```

2941 \cs_new_protected:Nn \stex_term_custom:nn {
2942   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2943   \str_set:Nn \l_tmpa_str { #2 }
2944   \tl_clear:N \l_tmpa_tl
2945   \int_zero:N \l_tmpa_int
2946   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2947   \__stex_terms_custom_loop:
2948 }

```

(End definition for \stex_term_custom:nn. This function is documented on page 39.)

_stex_terms_custom_loop:

```

2949 \cs_new_protected:Nn \_stex_terms_custom_loop: {
2950   \bool_set_false:N \l_tmpa_bool
2951   \bool_while_do:nn {
2952     \str_if_eq_p:ee X {
2953       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2954     }
2955   }{
2956     \int_incr:N \l_tmpa_int
2957   }
2958
2959   \peek_charcode:NTF [ {
2960     % notation/text component
2961     \_stex_terms_custom_component:w
2962   } {
2963     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2964       % all arguments read => finish
2965       \_stex_terms_custom_final:
2966     } {
2967       % arguments missing
2968       \peek_charcode_remove:NTF * {
2969         % invisible, specific argument position or both
2970         \peek_charcode:NTF [ {
2971           % visible specific argument position
2972           \_stex_terms_custom_arg:wn
2973         } {
2974           % invisible
2975           \peek_charcode_remove:NTF * {
2976             % invisible specific argument position
2977             \_stex_terms_custom_arg_inv:wn
2978           } {
2979             % invisible next argument
2980             \_stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2981           }
2982         }
2983       } {
2984         % next normal argument
2985         \_stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2986       }
2987     }
2988   }
2989 }

```

(End definition for _stex_terms_custom_loop:.)

_stex_terms_custom_arg_inv:wn

```

2990 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2991   \bool_set_true:N \l_tmpa_bool
2992   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
2993 }

```

(End definition for _stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```

2994 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
2995   \str_set:Nx \l_tmpb_str {
2996     \str_item:Nn \l_tmpa_str { #1 }
2997   }
2998   \str_case:VnTF \l_tmpb_str {
2999     { X } {
3000       \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
3001     }
3002     { i } { \_stex_terms_custom_set_X:n { #1 } }
3003     { b } { \_stex_terms_custom_set_X:n { #1 } }
3004     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3005     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3006   }{}{
3007     \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
3008   }
3009
3010   \bool_if:nTF \l_tmpa_bool {
3011     \tl_put_right:Nx \l_tmpa_tl {
3012       \stex_annotate_invisible:n {
3013         \_stex_term_arg:nn { \int_eval:n { #1 } }
3014         \exp_not:n { { #2 } }
3015       }
3016     }
3017   } {
3018     \tl_put_right:Nx \l_tmpa_tl {
3019       \_stex_term_arg:nn { \int_eval:n { #1 } }
3020       \exp_not:n { { #2 } }
3021     }
3022   }
3023
3024   \_stex_terms_custom_loop:
3025 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

3026 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3027   \str_set:Nx \l_tmpa_str {
3028     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3029     X
3030     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3031   }
3032 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

3033 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3034   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3035   \_stex_terms_custom_loop:
3036 }

```

(End definition for _stex_terms_custom_component:.)

`_stex_terms_custom_final:`

```

3037 \cs_new_protected:Nn \_stex_terms_custom_final: {
3038   \int_compare:nNnTF \l_tmpb_int = 0 {
3039     \exp_args:Nnno \_stex_term_oms:nnn
3040   }{
3041     \str_if_in:NnTF \l_tmpa_str {b} {
3042       \exp_args:Nnno \_stex_term_ombind:nnn
3043     } {
3044       \exp_args:Nnno \_stex_term_oma:nnn
3045     }
3046   }
3047   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
3048 }

```

(End definition for `_stex_terms_custom_final:.`)

`\symref`
`\symname`

```

3049 \NewDocumentCommand \symref { m m }{
3050   \let\compemph_uri_prev:\compemph@uri
3051   \let\compemph@uri\symrefemph@uri
3052   \STEXsymbol{#1}![#2]
3053   \let\compemph@uri\compemph_uri_prev:
3054 }
3055
3056 \keys_define:nn { stex / symname } {
3057   post .str_set_x:N = \l_stex_symname_post_str
3058 }
3059
3060 \cs_new_protected:Nn \stex_symname_args:n {
3061   \str_clear:N \l_stex_symname_post_str
3062   \keys_set:nn { stex / symname } { #1 }
3063 }
3064
3065 \NewDocumentCommand \symname { 0{} m }{
3066   \stex_symname_args:n { #1 }
3067   \stex_get_symbol:n { #2 }
3068   \str_set:Nx \l_tmpa_str {
3069     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3070   }
3071   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3072
3073   \let\compemph_uri_prev:\compemph@uri
3074   \let\compemph@uri\symrefemph@uri
3075   \exp_args:NNx \use:nn
3076   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3077     \l_tmpa_str \l_stex_symname_post_str
3078   ] }
3079   \let\compemph@uri\compemph_uri_prev:
3080 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 38.)

31.3 Notation Components

3081 <@@=stex_notationcomps>

`\stex_highlight_term:nn`

```

3082
3083 \str_new:N \l_stex_current_symbol_str
3084 \cs_new_protected:Nn \stex_highlight_term:nn {
3085   \exp_args:Nnx
3086   \use:nn {
3087     \str_set:Nx \l_stex_current_symbol_str { #1 }
3088     #2
3089   } {
3090     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3091     { \l_stex_current_symbol_str }
3092   }
3093 }
3094
3095 \cs_new_protected:Nn \stex_unhighlight_term:n {
3096   % \latexml_if:TF {
3097   %   #1
3098   % } {
3099   %   \rustex_if:TF {
3100   %     #1
3101   %   } {
3102     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3103   % }
3104   % }
3105 }
```

(End definition for `\stex_highlight_term:nn`. This function is documented on page 40.)

```

\comp
\compemph@uri 3106 \cs_new_protected:Npn \comp #1 {
\compemph 3107   \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3108     \rustex_if:TF {
\defemph@uri 3109       \stex_annotate:nnn { comp }{\l_stex_current_symbol_str}{ #1 }
\symrefemph 3110     }{
\symrefemph@uri 3111       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3112     }
3113   }
3114 }
3115
3116 \cs_new_protected:Npn \compemph@uri #1 #2 {
3117   \compemph{ #1 }
3118 }
3119
3120
3121 \cs_new_protected:Npn \compemph #1 {
3122   #1
3123 }
3124
3125 \cs_new_protected:Npn \defemph@uri #1 #2 {
3126   \defemph{#1}
3127 }
```

```

3128
3129 \cs_new_protected:Npn \defemph #1 {
3130     \textbf{#1}
3131 }
3132
3133 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3134     \symrefemph{#1}
3135 }
3136
3137 \cs_new_protected:Npn \symrefemph #1 {
3138     \textbf{#1}
3139 }

```

(End definition for `\comp` and others. These functions are documented on page 40.)

`\ellipses`

```

3140 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 40.)

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3141 \bool_new:N \l_stex_inparray_bool
3142 \bool_set_false:N \l_stex_inparray_bool
3143 \NewDocumentCommand \parray { m m } {
3144     \begingroup
3145     \bool_set_true:N \l_stex_inparray_bool
3146     \begin{array}{#1}
3147         #2
3148     \end{array}
3149     \endgroup
3150 }
3151
3152 \NewDocumentCommand \prmatrix { m } {
3153     \begingroup
3154     \bool_set_true:N \l_stex_inparray_bool
3155     \begin{matrix}
3156         #1
3157     \end{matrix}
3158     \endgroup
3159 }
3160
3161 \def \maybepline {
3162     \bool_if:NT \l_stex_inparray_bool {\hline}
3163 }
3164
3165 \def \parrayline #1 #2 {
3166     #1 #2 \bool_if:NT \l_stex_inparray_bool {\}
3167 }
3168
3169 \def \pmrow #1 { \parrayline{#1} }
3170
3171 \def \parraylineh #1 #2 {
3172     #1 #2 \bool_if:NT \l_stex_inparray_bool {\hline}
3173 }
3174

```

```

3175 \def \parraycell #1 {
3176   #1 \bool_if:NT \l_stex_inarray_bool {&}
3177 }

```

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

```

3178 \endpackage

```


Chapter 32

STEX -Structural Features Implementation

```
3179 <*package>
3180
3181 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3182
3183 <@@=stex_features>
3184
3185     Warnings and error messages
3186 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3187     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3188 }
3189 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3190     Symbol~#1~not~assigned~in~interpretmodule~#2
3191 }
```

32.1 Imports with modification

```
3191 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3192     \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3193         \__stex_features_get_symbol_from_cs:n { #1 }
3194     }{
3195         % argument is a string
3196         % is it a command name?
3197         \cs_if_exist:cTF { #1 }{
3198             \cs_set_eq:Nc \l_tmpa_tl { #1 }
3199             \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3200             \str_if_empty:NNTF \l_tmpa_str {
3201                 \exp_args:Nx \cs_if_eq:NNTF {
3202                     \tl_head:N \l_tmpa_tl
3203                 } \stex_invoke_symbol:n {
3204                     \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3205                 }{
3206                     \__stex_features_get_symbol_from_string:n { #1 }
```

```

3207     }
3208   } {
3209     \__stex_features_get_symbol_from_string:n { #1 }
3210   }
3211   ){
3212     % argument is not a command name
3213     \__stex_features_get_symbol_from_string:n { #1 }
3214     % \l_stex_all_symbols_seq
3215   }
3216 }
3217 }
3218
3219 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3220   \str_set:Nn \l_tmpa_str { #1 }
3221   \bool_set_false:N \l_tmpa_bool
3222   \bool_if:NF \l_tmpa_bool {
3223     \tl_set:Nn \l_tmpa_tl {
3224       \msg_set:nnn{stex}{error/unknownsymbol}{
3225         No~symbol~#1~found!
3226       }
3227       \msg_error:nn{stex}{error/unknownsymbol}
3228     }
3229     \str_set:Nn \l_tmpa_str { #1 }
3230     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3231     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3232       \str_set:Nn \l_tmpb_str { ##1 }
3233       \str_if_eq:eeT { \l_tmpa_str } {
3234         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3235       } {
3236         \seq_map_break:n {
3237           \tl_set:Nn \l_tmpa_tl {
3238             \str_set:Nn \l_stex_get_symbol_uri_str {
3239               ##1
3240             }
3241             \__stex_features_get_symbol_check:
3242           }
3243         }
3244       }
3245     }
3246     \l_tmpa_tl
3247   }
3248 }
3249
3250 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3251   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3252   { \tl_tail:N \l_tmpa_tl }
3253   \tl_if_single:NTF \l_tmpa_tl {
3254     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3255       \exp_after:wN \str_set:Nn \exp_after:wN
3256       \l_stex_get_symbol_uri_str \l_tmpa_tl
3257       \__stex_features_get_symbol_check:
3258     }{
3259       % TODO
3260       % tail is not a single group

```

```

3261     }
3262   }{
3263     % TODO
3264     % tail is not a single group
3265   }
3266 }
3267
3268 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3269   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3270   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3271     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3272     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3273     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3274       \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3275         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3276       }
3277     }
3278   }{
3279     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3280       \l_stex_current_copymodule_name_str~(inexplicably)
3281     }
3282   }
3283 }
3284
3285 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3286   \stex_import_module_uri:nn { #1 } { #2 }
3287   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3288   \stex_import_require_module:nnnn
3289     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3290     { \l_stex_import_path_str } { \l_stex_import_name_str }
3291   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3292   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3293   \seq_clear:N \l__stex_features_copymodule_fields_seq
3294   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3295     \seq_map_inline:cn {c_stex_module_###1_constants}{
3296       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3297         ###1 ? #####1
3298       }
3299     }
3300   }
3301   \seq_clear:N \l_tmpa_seq
3302   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3303     name      = \l_stex_current_copymodule_name_str ,
3304     module    = \l_stex_current_module_str ,
3305     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3306     includes  = \l_tmpa_seq ,
3307     fields    = \l_tmpa_seq
3308   }
3309   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3310     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3311   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3312     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3313   \stex_if_smsmode:F {
3314     \begin{stex_annotate_env} {#4} {

```

```

3315     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3316   }
3317   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3318 }
3319 \bool_set_eq:NN \l__stex_features_oldhtml_bool \l_stex_html_do_output_bool
3320 \bool_set_false:N \l_stex_html_do_output_bool
3321 }
3322 \cs_new_protected:Nn \stex_copymodule_end:n {
3323   \def \l_tmpa_cs ##1 ##2 {#1}
3324   \bool_set_eq:NN \l_stex_html_do_output_bool \l__stex_features_oldhtml_bool
3325   \tl_clear:N \l_tmpa_tl
3326   \tl_clear:N \l_tmpb_tl
3327   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3328   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3329     \seq_map_inline:cn {c_stex_module_##1_constants}{
3330       \tl_clear:N \l_tmpc_tl
3331       \l_tmpa_cs{##1}{####1}
3332       \str_if_exist:cTF {l__stex_features_copymodule_##1?####1_name_str} {
3333         \tl_put_right:Nx \l_tmpa_tl {
3334           \prop_set_from_keyval:cn {
3335             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3336           }{
3337             \exp_after:wN \prop_to_keyval:N \csname
3338               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3339             \endcsname
3340           }
3341           \seq_clear:c {
3342             l_stex_symdecl_
3343             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3344             _notations
3345           }
3346         }
3347         \tl_put_right:Nx \l_tmpc_tl {
3348           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_copy
3349           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3350         }
3351         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3352         \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3353           \tl_put_right:Nx \l_tmpc_tl {
3354             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3355           }
3356           \tl_put_right:Nx \l_tmpa_tl {
3357             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{\}
3358             \stex_invoke_symbol:n {
3359               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3360             }
3361           }
3362         }
3363       }
3364     }{
3365       \tl_put_right:Nx \l_tmpc_tl {
3366         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3367       }
3368       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3369 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3370 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3371 \tl_put_right:Nx \l_tmpa_tl {
3372   \prop_set_from_keyval:cn {
3373     l_stex_symdecl\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3374   }{
3375     \prop_to_keyval:N \l_tmpa_prop
3376   }
3377   \seq_clear:c {
3378     l_stex_symdecl_
3379     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3380     _notations
3381   }
3382 }
3383 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3384 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3385   \tl_put_right:Nx \l_tmpc_tl {
3386     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}
3387   }
3388   \tl_put_right:Nx \l_tmpa_tl {
3389     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3390       \stex_invoke_symbol:n {
3391         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3392       }
3393     }
3394   }
3395 }
3396 }
3397 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3398   \tl_put_right:Nx \l_tmpc_tl {
3399     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?####1_def_tl}}
3400   }
3401 }
3402 \tl_put_right:Nx \l_tmpb_tl {
3403   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3404 }
3405 }
3406 }
3407 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3408 \tl_put_left:Nx \l_tmpa_tl {
3409   \prop_set_from_keyval:cn {
3410     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
3411   }{
3412     \prop_to_keyval:N \l_stex_current_copymodule_prop
3413   }
3414 }
3415 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3416 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3417 \exp_args:Nx \stex_do_aftergroup:n {
3418   \exp_args:No \exp_not:n \l_tmpa_tl
3419 }
3420 \l_tmpb_tl
3421 \stex_if_smsmode:F {
3422   \end{stex_annotate_env}

```

```

3423 }
3424 }
3425
3426 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3427   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3428   \stex_deactivate_macro:Nn \symdecl {module~environments}
3429   \stex_deactivate_macro:Nn \symdef {module~environments}
3430   \stex_deactivate_macro:Nn \notation {module~environments}
3431   \stex_reactivate_macro:N \assign
3432   \stex_reactivate_macro:N \renamedec1
3433   \stex_reactivate_macro:N \donotcopy
3434   \stex_smsmode_do:
3435 }{
3436   \stex_copymodule_end:n {}
3437 }
3438
3439 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3440   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3441   \stex_deactivate_macro:Nn \symdecl {module~environments}
3442   \stex_deactivate_macro:Nn \symdef {module~environments}
3443   \stex_deactivate_macro:Nn \notation {module~environments}
3444   \stex_reactivate_macro:N \assign
3445   \stex_reactivate_macro:N \renamedec1
3446   \stex_reactivate_macro:N \donotcopy
3447   \stex_smsmode_do:
3448 }{
3449   \stex_copymodule_end:n {
3450     \tl_if_exist:cF {
3451       l__stex_features_copymodule_##1?##2_def_tl
3452     }{
3453       \msg_error:nnxx{stex}{error/interpretmodule/nodedefiniens}{
3454         ##1?##2
3455       }{\l_stex_current_copymodule_name_str}
3456     }
3457   }
3458 }
3459
3460 \NewDocumentCommand \donotcopy { 0{} m}{
3461   \stex_import_module_uri:nn { #1 } { #2 }
3462   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3463   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3464     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3465     \seq_map_inline:cn {c_stex_module_##1_constants}{
3466       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3467       \bool_lazy_any_p:nT {
3468         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3469         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3470         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3471       }{
3472         % TODO throw error
3473       }
3474     }
3475   }
3476 }

```

```

3477 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3478 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3479 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3480 }
3481
3482 \NewDocumentCommand \assign { m m }{
3483   \stex_get_symbol_in_copymodule:n {#1}
3484   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3485   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3486 }
3487
3488 \keys_define:nn { stex / renamedec1 } {
3489   name .str_set_x:N = \l_stex_renamedec1_name_str
3490 }
3491 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3492   \str_clear:N \l_stex_renamedec1_name_str
3493
3494   \keys_set:nn { stex / renamedec1 } { #1 }
3495 }
3496
3497 \NewDocumentCommand \renamedec1 { O{} m m }{
3498   \__stex_features_renamedec1_args:n { #1 }
3499   \stex_get_symbol_in_copymodule:n {#2}
3500   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3501   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3502   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3503     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3504       \l_stex_get_symbol_uri_str
3505     } }
3506   } {
3507     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3508     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3509     \prop_set_eq:cc {l_stex_symdecl_
3510       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3511       _prop
3512     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3513     \seq_set_eq:cc {l_stex_symdecl_
3514       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3515       _notations
3516     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3517     \prop_put:cnx {l_stex_symdecl_
3518       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3519       _prop
3520     }{ name }{ \l_stex_renamedec1_name_str }
3521     \prop_put:cnx {l_stex_symdecl_
3522       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3523       _prop
3524     }{ module }{ \l_stex_current_module_str }
3525     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3526       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3527     }
3528     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3529       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3530     } }

```

```

3531 }
3532 }
3533 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3534 % \_stex_notation_args:n { #1 }
3535 % \tl_clear:N \l_stex_symdecl_definiens_tl
3536 % \stex_get_symbol_in_copymodule:n { #2 }
3537 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3538 % % todo
3539 %}
3540 \stex_deactivate_macro:Nn \assign {copymodules}
3541 \stex_deactivate_macro:Nn \renamedekl {copymodules}
3542 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3543
3544
3545 \seq_new:N \l_stex_implicit_morphisms_seq
3546 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3547 \stex_import_module_uri:nn { #1 } { #2 }
3548 \stex_debug:nn{implicits}{
3549 Implicit~morphism:~
3550 \l_stex_module_ns_str ? \l__stex_features_name_str
3551 }
3552 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3553 \l_stex_module_ns_str ? \l__stex_features_name_str
3554 }{
3555 \msg_error:nnn{stex}{error/conflictingmodules}{
3556 \l_stex_module_ns_str ? \l__stex_features_name_str
3557 }
3558 }
3559
3560 % TODO
3561
3562
3563
3564 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3565 \l_stex_module_ns_str ? \l__stex_features_name_str
3566 }
3567 }
3568

```

32.2 The feature environment

structural@feature

```

3569
3570 \NewDocumentEnvironment{structural@feature}{ m m m }{
3571 \stex_if_in_module:F {
3572 \msg_set:nnn{stex}{error/nomodule}{
3573 Structural~Feature~has~to~occur~in~a~module:\\
3574 Feature~#2~of~type~#1\\
3575 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3576 }
3577 \msg_error:nn{stex}{error/nomodule}
3578 }
3579

```



```

3580 \str_set:Nx \l_stex_module_name_str {
3581   \prop_item:Nn \l_stex_current_module_prop
3582     { name } / #2 - feature
3583 }
3584
3585 \str_set:Nx \l_stex_module_ns_str {
3586   \prop_item:Nn \l_stex_current_module_prop
3587     { ns }
3588 }
3589
3590
3591 \str_clear:N \l_tmpa_str
3592 \seq_clear:N \l_tmpa_seq
3593 \tl_clear:N \l_tmpa_tl
3594 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3595   origname = #2,
3596   name     = \l_stex_module_name_str ,
3597   ns       = \l_stex_module_ns_str ,
3598   imports  = \exp_not:o { \l_tmpa_seq } ,
3599   constants = \exp_not:o { \l_tmpa_seq } ,
3600   content  = \exp_not:o { \l_tmpa_tl } ,
3601   file     = \exp_not:o { \g_stex_currentfile_seq } ,
3602   lang     = \l_stex_module_lang_str ,
3603   sig      = \l_tmpa_str ,
3604   meta     = \l_tmpa_str ,
3605   feature  = #1 ,
3606 }
3607
3608 \stex_if_smsmode:F {
3609   \begin{stex_annotate_env}{ feature:#1 }{}
3610   \stex_annotate_invisible:nnn{header}{}{ #3 }
3611 }
3612 }{
3613   \str_set:Nx \l_tmpa_str {
3614     c_stex_feature_
3615     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3616     \prop_item:Nn \l_stex_current_module_prop { name }
3617     _prop
3618   }
3619   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3620   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3621   \stex_if_smsmode:TF {
3622     \exp_args:Nx \stex_add_to_sms:n {
3623       \prop_gset_from_keyval:cn {
3624         c_stex_feature_
3625         \prop_item:Nn \l_stex_current_module_prop { ns } ?
3626         \prop_item:Nn \l_stex_current_module_prop { name }
3627         _prop
3628       } {
3629         origname = #2,
3630         name     = \prop_item:cn { \l_tmpa_str } { name } ,
3631         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
3632         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
3633         constants = \prop_item:cn { \l_tmpa_str } { constants } ,

```

```

3634         content    = \prop_item:cn { \l_tmpa_str } { content } ,
3635         file       = \prop_item:cn { \l_tmpa_str } { file } ,
3636         lang       = \prop_item:cn { \l_tmpa_str } { lang } ,
3637         sig        = \prop_item:cn { \l_tmpa_str } { sig } ,
3638         meta       = \prop_item:cn { \l_tmpa_str } { meta } ,
3639         feature    = \prop_item:cn { \l_tmpa_str } { feature }
3640     }
3641 }
3642 } {
3643     \end{stex_annotate_env}
3644 }
3645 }
3646

```

32.3 Features

structure

```

3647
3648 \prop_new:N \l_stex_all_structures_prop
3649
3650 \keys_define:nn { stex / features / structure } {
3651     name          .str_set_x:N = \l__stex_features_structure_name_str ,
3652 }
3653
3654 \cs_new_protected:Nn \__stex_features_structure_args:n {
3655     \str_clear:N \l__stex_features_structure_name_str
3656     \keys_set:nn { stex / features / structure } { #1 }
3657 }
3658
3659 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3660 %   \__stex_features_structure_args:n { ##1 }
3661 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3662 %       \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3663 %   }
3664 %} {
3665 %
3666 %}
3667
3668 \NewDocumentEnvironment{mathstructure}{0{ } m }{
3669     \__stex_features_structure_args:n { #1 }
3670     \str_if_empty:NT \l__stex_features_structure_name_str {
3671         \str_set:Nx \l__stex_features_structure_name_str { #2 }
3672     }
3673     \exp_args:Nnnx
3674     \begin{structural@feature}{ structure }
3675         { \l__stex_features_structure_name_str }{}
3676         \seq_clear:N \l_tmpa_seq
3677         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3678         \stex_smsmode_do:
3679     }{
3680         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3681         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3682         \str_set:Nx \l_tmpa_str {

```

```

3683     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3684     \prop_item:Nn \l_stex_current_module_prop { name }
3685   }
3686   \seq_map_inline:Nn \l_tmpa_seq {
3687     \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3688   }
3689   \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3690   \exp_args:Nnx
3691   \AddToHookNext { env / mathstructure / after }{
3692     \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3693       \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}
3694     }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3695     \STEXexport {
3696       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3697       {\prop_item:Nn \l_stex_current_module_prop { origname }}
3698       {\l_tmpa_str}
3699       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3700       {#2}{\l_tmpa_str}
3701     %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3702     %     \prop_item:Nn \l_stex_current_module_prop { origname },
3703     %     \l_tmpa_str
3704     %   }
3705     %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3706     %     #2,\l_tmpa_str
3707     %   }
3708     %   \tl_set:cx { #2 } {
3709     %     \stex_invoke_structure:n { \l_tmpa_str }
3710   }
3711 }
3712
3713 \end{structural@feature}
3714 % \g_stex_last_feature_prop
3715 }

```

\instantiate

```

3716 \seq_new:N \l__stex_features_structure_field_seq
3717 \str_new:N \l__stex_features_structure_field_str
3718 \str_new:N \l__stex_features_structure_def_tl
3719 \prop_new:N \l__stex_features_structure_prop
3720 \NewDocumentCommand \instantiate { m O{} m }{
3721   \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3722   \prop_set_eq:Nc \l__stex_features_structure_prop {
3723     c_stex_feature_\l_tmpa_str _prop
3724   }
3725   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3726   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3727     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3728     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3729       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3730       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3731         {!} \l_tmpa_tl
3732       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3733         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3734         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl

```

```

3735     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3736   }{
3737     \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3738     \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3739     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3740       \l_tmpa_tl
3741     \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3742       \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3743       \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3744     }{
3745       \tl_clear:N \l_tmpb_tl
3746     }
3747   }
3748 }{
3749   \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3750   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3751     \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3752     \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3753     \tl_clear:N \l_tmpa_tl
3754   }{
3755     % TODO throw error
3756   }
3757 }
3758 % \l_tmpa_str: name
3759 % \l_tmpa_tl: definiens
3760 % \l_tmpb_tl: notation
3761 \tl_if_empty:NT \l__stex_features_structure_field_str {
3762   % TODO throw error
3763 }
3764 \str_clear:N \l_tmpb_str
3765
3766 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3767 \seq_map_inline:Nn \l_tmpa_seq {
3768   \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3769   \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3770   \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3771     \seq_map_break:n {
3772       \str_set:Nn \l_tmpb_str { ####1 }
3773     }
3774   }
3775 }
3776 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3777   \l_tmpb_str
3778
3779 \tl_if_empty:NNTF \l_tmpb_tl {
3780   \tl_if_empty:NF \l_tmpa_tl {
3781     \exp_args:Nx \use:n {
3782       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3783     }
3784   }
3785 }{
3786   \tl_if_empty:NNTF \l_tmpa_tl {
3787     \exp_args:Nx \use:n {
3788       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\

```

```

3789     }
3790
3791   }{
3792     \exp_args:Nx \use:n {
3793       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3794       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3795     }
3796   }
3797 }
3798 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3799 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3800 % #3/\l__stex_features_structure_field_str
3801 % \par
3802 % \expandafter\present\csname
3803 %   \l_stex_symdecl_
3804 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3805 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3806 %   #3/\l__stex_features_structure_field_str
3807 %   _prop
3808 % \endcsname
3809 }
3810
3811 \tl_clear:N \l__stex_features_structure_def_tl
3812
3813 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3814 \seq_map_inline:Nn \l_tmpa_seq {
3815   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3816   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3817   \exp_args:Nx \use:n {
3818     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3819
3820     }
3821   }
3822
3823   \prop_if_exist:cF {
3824     \l_stex_symdecl_
3825     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3826     \prop_item:Nn \l_stex_current_module_prop {name} ?
3827     #3/\l_tmpa_str
3828     _prop
3829   }{
3830     \prop_get:cnN { \l_stex_symdecl_ ##1 _prop } {args}
3831     \l_tmpb_str
3832     \exp_args:Nx \use:n {
3833       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3834     }
3835   }
3836 }
3837
3838 \symdecl*[type={\STEXsymbol{module-type}}{
3839   \_stex_term_math_oms:nnnn {
3840     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3841     \prop_item:Nn \l__stex_features_structure_prop {name}
3842   }-{}{0}-{}

```

```

3843   }}]{#3}
3844
3845   % TODO: -> sms file
3846
3847   \tl_set:cx{ #3 }{
3848     \stex_invoke_structure:nnn {
3849       \prop_item:Nn \l_stex_current_module_prop {ns} ?
3850       \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3851     } {
3852       \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3853       \prop_item:Nn \l__stex_features_structure_prop {name}
3854     }
3855   }
3856   \stex_smsmode_do:
3857 }

```

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

\stex_invoke_structure:nnn

```

3858 % #1: URI of the instance
3859 % #2: URI of the instantiated module
3860 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3861   \tl_if_empty:nTF{ #3 }{
3862     \prop_set_eq:Nc \l__stex_features_structure_prop {
3863       c_stex_feature_ #2 _prop
3864     }
3865     \tl_clear:N \l_tmpa_tl
3866     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3867     \seq_map_inline:Nn \l_tmpa_seq {
3868       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3869       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3870       \cs_if_exist:cT {
3871         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3872       }{
3873         \tl_if_empty:NF \l_tmpa_tl {
3874           \tl_put_right:Nn \l_tmpa_tl {,}
3875         }
3876         \tl_put_right:Nx \l_tmpa_tl {
3877           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3878         }
3879       }
3880     }
3881     \exp_args:No \mathstrut \l_tmpa_tl
3882   }{
3883     \stex_invoke_symbol:n{#1/#3}
3884   }
3885 }

```

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

3886 </package>

Chapter 33

STEX -Statements Implementation

```
3887 <*package>
3888
3889 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3890
3891 \protected\def\ignorespacesandpars{
3892   \begingroup\catcode13=10\relax
3893   \@ifnextchar\par{
3894     \endgroup\expandafter\ignorespacesandpars\@gobble
3895   }{
3896     \endgroup
3897   }
3898 }
3899
3900 <@@=stex_statements>
3901
3902   Warnings and error messages
```

\titleemph

```
3902 \def\titleemph#1{\textbf{#1}}
```

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

33.1 Definitions

definiendum

```
3903 \keys_define:nn {stex / definiendum }{
3904   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3905   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3906   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3907 }
3908 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3909   \str_clear:N \l__stex_statements_definiendum_root_str
3910   \tl_clear:N \l__stex_statements_definiendum_post_tl
3911   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3912 \keys_set:nn { stex / definiendum } { #1 }
3913 }
3914 \NewDocumentCommand \definiendum { 0{} m m } {
3915   \__stex_statements_definiendum_args:n { #1 }
3916   \stex_get_symbol:n { #2 }
3917   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3918   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3919     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3920       \tl_set:Nn \l_tmpa_tl { #3 }
3921     } {
3922       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3923       \tl_set:Nn \l_tmpa_tl {
3924         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3925       }
3926     }
3927   } {
3928     \tl_set:Nn \l_tmpa_tl { #3 }
3929   }
3930
3931   % TODO root
3932   \rustex_if:TF {
3933     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3934   } {
3935     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3936   }
3937 }
3938 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3939
3940 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3941
3942 \NewDocumentCommand \definame { 0{} m } {
3943   \__stex_statements_definiendum_args:n { #1 }
3944   % TODO: root
3945   \stex_get_symbol:n { #2 }
3946   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3947   \str_set:Nx \l_tmpa_str {
3948     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3949   }
3950   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3951   \rustex_if:TF {
3952     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3953       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3954     }
3955   } {
3956     \defemph@uri {
3957       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3958     } { \l_stex_get_symbol_uri_str }
3959   }
3960 }
3961 \stex_deactivate_macro:Nn \definame {definition~environments}

```



```

3962
3963 \NewDocumentCommand \Definame { 0{ } m } {
3964   \_stex_statements_definiendum_args:n { #1 }
3965   \stex_get_symbol:n { #2 }
3966   \str_set:Nx \l_tmpa_str {
3967     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3968   }
3969   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3970   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3971   \rustex_if:TF {
3972     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3973       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3974     }
3975   } {
3976     \defemph@uri {
3977       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
3978     } { \l_stex_get_symbol_uri_str }
3979   }
3980 }
3981 \stex_deactivate_macro:Nn \Definame {definition-environments}
3982
3983 \NewDocumentCommand \Symname { 0{ } m }{
3984   \stex_symname_args:n { #1 }
3985   \stex_get_symbol:n { #2 }
3986   \str_set:Nx \l_tmpa_str {
3987     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3988   }
3989   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3990   \let\compemph_uri_prev:\compemph@uri
3991   \let\compemph@uri\symrefemph@uri
3992   \exp_args:NNx \use:nn
3993   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
3994     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3995     \l_stex_symname_post_str
3996   ] }
3997   \let\compemph@uri\compemph_uri_prev:
3998 }

```

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

sdefinition

```

3999
4000 \keys_define:nn {stex / sdefinition }{
4001   type      .str_set_x:N = \sdefinitiontype,
4002   id        .str_set_x:N = \sdefinitionid,
4003   name      .str_set_x:N = \sdefinitionname,
4004   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4005   title     .tl_set:N     = \sdefinitiontitle
4006 }
4007 \cs_new_protected:Nn \_stex_statements_sdefinition_args:n {
4008   \str_clear:N \sdefinitiontype
4009   \str_clear:N \sdefinitionid
4010   \str_clear:N \sdefinitionname
4011   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```

```

4012 \tl_clear:N \sdefinitiontitle
4013 \keys_set:nn { stex / sdefinition }{ #1 }
4014 }
4015
4016 \NewDocumentEnvironment{sdefinition}{0{}}{
4017   \__stex_statements_sdefinition_args:n{ #1 }
4018   \stex_reactivate_macro:N \definiendum
4019   \stex_reactivate_macro:N \definame
4020   \stex_reactivate_macro:N \Definame
4021   \stex_if_smsmode:F{
4022     \seq_clear:N \l_tmpa_seq
4023     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4024       \str_if_eq:nnF{ ##1 }{ }{
4025         \stex_get_symbol:n { ##1 }
4026         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4027           \l_stex_get_symbol_uri_str
4028         }
4029       }
4030     }
4031     \exp_args:Nnnx
4032     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4033     \str_if_empty:NF \sdefinitiontype {
4034       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4035     }
4036     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4037     \tl_clear:N \l_tmpa_tl
4038     \clist_map_inline:Nn \l_tmpa_clist {
4039       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4040         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4041       }
4042     }
4043     \tl_if_empty:NTF \l_tmpa_tl {
4044       \__stex_statements_sdefinition_start:
4045     }{
4046       \l_tmpa_tl
4047     }
4048   }
4049   \stex_ref_new_doc_target:n \sdefinitionid
4050   \stex_smsmode_do:
4051 }{
4052   \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4053   \stex_if_smsmode:F {
4054     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4055     \tl_clear:N \l_tmpa_tl
4056     \clist_map_inline:Nn \l_tmpa_clist {
4057       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4058         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4059       }
4060     }
4061     \tl_if_empty:NTF \l_tmpa_tl {
4062       \__stex_statements_sdefinition_end:
4063     }{
4064       \l_tmpa_tl
4065     }

```

```

4066     \end{stex_annotate_env}
4067   }
4068 }

```

\stexpatchdefinition

```

4069 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4070   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4071     ~(\sdefinitiontitle)
4072   }~}
4073 }
4074 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4075
4076 \newcommand\stexpatchdefinition[3] [] {
4077   \str_set:Nx \l_tmpa_str{ #1 }
4078   \str_if_empty:NTF \l_tmpa_str {
4079     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4080     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4081   }{
4082     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4083     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4084   }
4085 }

```

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

\inlinedef inline:

```

4086 \keys_define:nn {stex / inlinedef }{
4087   type      .str_set_x:N = \sdefinitiontype,
4088   id        .str_set_x:N = \sdefinitionid,
4089   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4090   name      .str_set_x:N = \sdefinitionname
4091 }
4092 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4093   \str_clear:N \sdefinitiontype
4094   \str_clear:N \sdefinitionid
4095   \str_clear:N \sdefinitionname
4096   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4097   \keys_set:nn { stex / inlinedef }{ #1 }
4098 }
4099 \NewDocumentCommand \inlinedef { 0{} m } {
4100   \begingroup
4101   \__stex_statements_inlinedef_args:n{ #1 }
4102   \stex_ref_new_doc_target:n \sdefinitionid
4103   \stex_reactivate_macro:N \definiendum
4104   \stex_reactivate_macro:N \definame
4105   \stex_reactivate_macro:N \Definame
4106   \stex_if_smsmode:TF{
4107     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4108   }{
4109     \seq_clear:N \l_tmpa_seq
4110     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4111       \str_if_eq:nnF{ ##1 }{}{
4112         \stex_get_symbol:n { ##1 }
4113         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {

```

```

4114         \l_stex_get_symbol_uri_str
4115     }
4116 }
4117 }
4118 \exp_args:Nnx
4119 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4120     \str_if_empty:NF \sdefinitiontype {
4121         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4122     }
4123     #2
4124     \str_if_empty:NF \sdefinitionname { \symdecl*{\sdefinitionname} }
4125 }
4126 }
4127 \endgroup
4128 \stex_smsmode_do:
4129 }

```

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

33.2 Assertions

sassertion

```

4130
4131 \keys_define:nn {stex / sassertion }{
4132     type      .str_set_x:N = \sassertiontype,
4133     id        .str_set_x:N = \sassertionid,
4134     title     .tl_set:N    = \sassertiontitle ,
4135     for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4136     name      .str_set_x:N = \sassertionname
4137 }
4138 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4139     \str_clear:N \sassertiontype
4140     \str_clear:N \sassertionid
4141     \str_clear:N \sassertionname
4142     \clist_clear:N \l__stex_statements_sassertion_for_clist
4143     \tl_clear:N \sassertiontitle
4144     \keys_set:nn { stex / sassertion }{ #1 }
4145 }
4146
4147 %\tl_new:N \g__stex_statements_aftergroup_tl
4148
4149 \NewDocumentEnvironment{sassertion}{0{}}{
4150     \__stex_statements_sassertion_args:n{ #1 }
4151     \stex_if_smsmode:F {
4152         \seq_clear:N \l_tmpa_seq
4153         \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4154             \str_if_eq:nnF{ ##1 }{ }{
4155                 \stex_get_symbol:n { ##1 }
4156                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4157                     \l_stex_get_symbol_uri_str
4158                 }
4159             }
4160         }

```

```

4161 \exp_args:Nnnx
4162 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4163 \str_if_empty:NF \sassertiontype {
4164   \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4165 }
4166 \clist_set:No \l_tmpa_clist \sassertiontype
4167 \tl_clear:N \l_tmpa_tl
4168 \clist_map_inline:Nn \l_tmpa_clist {
4169   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4170     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4171   }
4172 }
4173 \tl_if_empty:NTF \l_tmpa_tl {
4174   \__stex_statements_sassertion_start:
4175 }{
4176   \l_tmpa_tl
4177 }
4178 }
4179 \stex_ref_new_doc_target:n \sassertionid
4180 \stex_smsmode_do:
4181 ){
4182   \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4183   \stex_if_smsmode:F {
4184     \clist_set:No \l_tmpa_clist \sassertiontype
4185     \tl_clear:N \l_tmpa_tl
4186     \clist_map_inline:Nn \l_tmpa_clist {
4187       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4188         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4189       }
4190     }
4191     \tl_if_empty:NTF \l_tmpa_tl {
4192       \__stex_statements_sassertion_end:
4193     }{
4194       \l_tmpa_tl
4195     }
4196   }
4197 }
4198 }

```

\stexpatchassertion

```

4199
4200 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4201   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4202     (\sassertiontitle)
4203   }~}
4204 }
4205 \cs_new_protected:Nn \__stex_statements_sassertion_end: { \par\medskip}
4206
4207 \newcommand\stexpatchassertion[3] [] {
4208   \str_set:Nx \l_tmpa_str{ #1 }
4209   \str_if_empty:NTF \l_tmpa_str {
4210     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4211     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4212   }{

```

```

4213     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4214     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4215   }
4216 }

```

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

`\inlineass` inline:

```

4217 \keys_define:nn {stex / inlineass }{
4218   type      .str_set_x:N = \sassertiontype,
4219   id        .str_set_x:N = \sassertionid,
4220   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4221   name      .str_set_x:N = \sassertionname
4222 }
4223 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4224   \str_clear:N \sassertiontype
4225   \str_clear:N \sassertionid
4226   \str_clear:N \sassertionname
4227   \clist_clear:N \l__stex_statements_sassertion_for_clist
4228   \keys_set:nn { stex / inlineass }{ #1 }
4229 }
4230 \NewDocumentCommand \inlineass { 0{} m } {
4231   \begingroup
4232   \__stex_statements_inlineass_args:n{ #1 }
4233   \stex_ref_new_doc_target:n \sassertionid
4234   \stex_if_smsmode:TF{
4235     \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4236   }{
4237     \seq_clear:N \l_tmpa_seq
4238     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4239       \str_if_eq:nnF{ ##1 }{ }{
4240         \stex_get_symbol:n { ##1 }
4241         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4242           \l_stex_get_symbol_uri_str
4243         }
4244       }
4245     }
4246     \exp_args:Nnx
4247     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4248       \str_if_empty:NF \sassertiontype {
4249         \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4250       }
4251       #2
4252       \str_if_empty:NF \sassertionname { \symdecl*{\sassertionname} }
4253     }
4254   }
4255   \endgroup
4256   \stex_smsmode_do:
4257 }

```

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

33.3 Examples

sexample

```

4258
4259 \keys_define:nn {stex / sexample }{
4260   type      .str_set_x:N = \exampletype,
4261   id        .str_set_x:N = \sexampleid,
4262   title     .tl_set:N     = \sexamplename,
4263   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4264 }
4265 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4266   \str_clear:N \sexampletype
4267   \str_clear:N \sexampleid
4268   \tl_clear:N \sexamplename
4269   \clist_clear:N \l__stex_statements_sexample_for_clist
4270   \keys_set:nn { stex / sexample }{ #1 }
4271 }
4272
4273 \NewDocumentEnvironment{sexample}{0{}}{
4274   \__stex_statements_sexample_args:n{ #1 }
4275   \stex_if_smsmode:F {
4276     \seq_clear:N \l_tmpa_seq
4277     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4278       \str_if_eq:nnF{ ##1 }{}{
4279         \stex_get_symbol:n { ##1 }
4280         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4281           \l_stex_get_symbol_uri_str
4282         }
4283       }
4284     }
4285     \exp_args:Nnnx
4286     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4287     \str_if_empty:NF \sexampletype {
4288       \stex_annotate_invisible:nnn{type}{\sexampletype}{}
4289     }
4290     \clist_set:Nn \l_tmpa_clist \sexampletype
4291     \tl_clear:N \l_tmpa_tl
4292     \clist_map_inline:Nn \l_tmpa_clist {
4293       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4294         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4295       }
4296     }
4297     \tl_if_empty:NTF \l_tmpa_tl {
4298       \__stex_statements_sexample_start:
4299     }{
4300       \l_tmpa_tl
4301     }
4302   }
4303   \stex_ref_new_doc_target:n \sexampleid
4304   \stex_smsmode_do:
4305 }{
4306   \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4307   \stex_if_smsmode:F {
4308     \clist_set:Nn \l_tmpa_clist \sexampletype

```

```

4309 \tl_clear:N \l_tmpa_tl
4310 \clist_map_inline:Nn \l_tmpa_clist {
4311   \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4312     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4313   }
4314 }
4315 \tl_if_empty:NTF \l_tmpa_tl {
4316   \__stex_statements_sexample_end:
4317 }{
4318   \l_tmpa_tl
4319 }
4320 \end{stex_annotate_env}
4321 }
4322 }

```

\stexpatchexample

```

4323
4324 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4325   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
4326     (\sexampltitle)
4327   }~}
4328 }
4329 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
4330
4331 \newcommand\stexpatchexample[3] [] {
4332   \str_set:Nx \l_tmpa_str{ #1 }
4333   \str_if_empty:NTF \l_tmpa_str {
4334     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4335     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4336   }{
4337     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4338     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4339   }
4340 }

```

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

\inlineex inline:

```

4341 \keys_define:nn {stex / inlineex }{
4342   type      .str_set_x:N = \sexampltype,
4343   id        .str_set_x:N = \sexampleid,
4344   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
4345   name      .str_set_x:N = \sexamplname
4346 }
4347 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4348   \str_clear:N \sexampltype
4349   \str_clear:N \sexampleid
4350   \str_clear:N \sexamplname
4351   \clist_clear:N \l__stex_statements_sexample_for_clist
4352   \keys_set:nn { stex / inlineex }{ #1 }
4353 }
4354 \NewDocumentCommand \inlineex { 0{} m } {
4355   \begingroup
4356   \__stex_statements_inlineex_args:n{ #1 }

```



```

4357 \stex_ref_new_doc_target:n \sexampleid
4358 \stex_if_smsmode:TF{
4359   \str_if_empty:NF \sexamplename { \symdecl*{\examplename} }
4360 }{
4361   \seq_clear:N \l_tmpa_seq
4362   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4363     \str_if_eq:nnF{ ##1 }{ }{
4364       \stex_get_symbol:n { ##1 }
4365       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4366         \l_stex_get_symbol_uri_str
4367       }
4368     }
4369   }
4370   \exp_args:Nnx
4371   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {},}{ }{
4372     \str_if_empty:NF \sexamplotype {
4373       \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4374     }
4375     #2
4376     \str_if_empty:NF \sexamplename { \symdecl*{\sexamplename} }
4377   }
4378 }
4379 \endgroup
4380 \stex_smsmode_do:
4381 }

```

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

33.4 Logical Paragraphs

sparagraph

```

4382 \keys_define:nn { stex / sparagraph } {
4383   id      .str_set:N = \sparagraphid ,
4384   title   .tl_set:N  = \l_stex_sparagraph_title_tl ,
4385   type     .str_set:N = \sparagraphtype ,
4386   for      .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4387   from     .tl_set:N  = \sparagraphfrom ,
4388   to       .tl_set:N  = \sparagraphto ,
4389   start    .tl_set:N  = \l_stex_sparagraph_start_tl ,
4390   name     .str_set:N  = \sparagraphname
4391 }
4392
4393 \cs_new_protected:Nn \stex_sparagraph_args:n {
4394   \tl_clear:N \l_stex_sparagraph_title_tl
4395   \tl_clear:N \sparagraphfrom
4396   \tl_clear:N \sparagraphto
4397   \tl_clear:N \l_stex_sparagraph_start_tl
4398   \str_clear:N \sparagraphid
4399   \str_clear:N \sparagraphtype
4400   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4401   \str_clear:N \sparagraphname
4402   \keys_set:nn { stex / sparagraph }{ #1 }
4403 }

```

```

4404 \newif\if@in@omtext\@in@omtextfalse
4405
4406 \NewDocumentEnvironment {sparagraph} { 0{} } {
4407   \stex_sparagraph_args:n { #1 }
4408   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4409     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4410   }{
4411     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4412   }
4413   \@in@omtexttrue
4414   \stex_if_smsmode:F {
4415     \seq_clear:N \l_tmpa_seq
4416     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4417       \str_if_eq:nnF{ ##1 }{}{
4418         \stex_get_symbol:n { ##1 }
4419         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4420           \l_stex_get_symbol_uri_str
4421         }
4422       }
4423     }
4424     \exp_args:Nnnx
4425     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4426     \str_if_empty:NF \sparagraphtype {
4427       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4428     }
4429     \str_if_empty:NF \sparagraphfrom {
4430       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4431     }
4432     \str_if_empty:NF \sparagraphto {
4433       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4434     }
4435     \clist_set:No \l_tmpa_clist \sparagraphtype
4436     \tl_clear:N \l_tmpa_tl
4437     \clist_map_inline:Nn \sparagraphtype {
4438       \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4439         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4440       }
4441     }
4442     \tl_if_empty:NTF \l_tmpa_tl {
4443       \__stex_statements_sparagraph_start:
4444     }{
4445       \l_tmpa_tl
4446     }
4447   }
4448   \stex_ref_new_doc_target:n \sparagraphid
4449   \stex_smsmode_do:
4450   \ignorespacesandpars
4451 }{
4452   \stex_if_smsmode:F {
4453     \clist_set:No \l_tmpa_clist \sparagraphtype
4454     \tl_clear:N \l_tmpa_tl
4455     \clist_map_inline:Nn \l_tmpa_clist {
4456       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
4457         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}

```

```

4458     }
4459   }
4460   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname} }
4461   \tl_if_empty:NTF \l_tmpa_tl {
4462     \__stex_statements_sparagraph_end:
4463   }{
4464     \l_tmpa_tl
4465   }
4466   \end{stex_annotate_env}
4467 }
4468 }

```

\stexpatchparagraph

```

4469
4470 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
4471   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4472     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4473       \titleemph{\l_stex_sparagraph_title_tl}:~
4474     }
4475   }{
4476     \titleemph{\l_stex_sparagraph_start_tl}~
4477   }
4478 }
4479 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4480
4481 \newcommand\stexpatchparagraph[3] [] {
4482   \str_set:Nx \l_tmpa_str{ #1 }
4483   \str_if_empty:NTF \l_tmpa_str {
4484     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4485     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4486   }{
4487     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
4488     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4489   }
4490 }
4491
4492 \keys_define:nn { stex / inlinepara} {
4493   id      .str_set_x:N = \sparagraphid ,
4494   type    .str_set_x:N = \sparagraphtype ,
4495   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4496   from    .tl_set:N    = \sparagraphfrom ,
4497   to      .tl_set:N    = \sparagraphto ,
4498   name    .str_set:N   = \sparagraphname
4499 }
4500 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4501   \tl_clear:N \sparagraphfrom
4502   \tl_clear:N \sparagraphto
4503   \str_clear:N \sparagraphid
4504   \str_clear:N \sparagraphtype
4505   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4506   \str_clear:N \sparagraphname
4507   \keys_set:nn { stex / inlinepara }{ #1 }
4508 }
4509 \NewDocumentCommand \inlinepara { 0{} m } {

```

```

4510 \begingroup
4511 \__stex_statements_inlinepara_args:n{ #1 }
4512 \stex_ref_new_doc_target:n \sparagraphid
4513 \stex_if_smsmode:TF{
4514   \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4515 }{
4516   \seq_clear:N \l_tmpa_seq
4517   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4518     \str_if_eq:nnF{ ##1 }{}{
4519       \stex_get_symbol:n { ##1 }
4520       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4521         \l_stex_get_symbol_uri_str
4522       }
4523     }
4524   }
4525   \exp_args:Nnx
4526   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4527     \str_if_empty:NF \sparagraphtype {
4528       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4529     }
4530     \str_if_empty:NF \sparagraphfrom {
4531       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4532     }
4533     \str_if_empty:NF \sparagraphto {
4534       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4535     }
4536     #2
4537     \str_if_empty:NF \sparagraphname { \symdecl*\sparagraphname } }
4538   }
4539 }
4540 \endgroup
4541 \stex_smsmode_do:
4542 }
4543

```

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

symboldoc

```

4544 \NewDocumentEnvironment{symboldoc}{ m }{
4545   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4546   \seq_clear:N \l_tmpb_seq
4547   \seq_map_inline:Nn \l_tmpa_seq {
4548     \str_if_eq:nnF{ ##1 }{}{
4549       \stex_get_symbol:n { ##1 }
4550       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4551         \l_stex_get_symbol_uri_str
4552       }
4553     }
4554   }
4555   \par
4556   \exp_args:Nnnx
4557   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4558 }{
4559   \end{stex_annotate_env}
4560 }

```

4561 </package>

Chapter 34

The Implementation

34.1 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).¹³

```
4562 <*package>
4563 <@@=stex_sproof>
4564
4565 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4566
```

34.2 Proofs

We first define some keys for the proof environment.

```
4567 \keys_define:nn { stex / spf } {
4568   id          .str_set:N = \l__stex_sproof_spf_id_str,
4569   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4570   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4571   from       .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4572   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4573   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4574   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4575   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4576   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4577   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4578 }
4579 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4580   \str_clear:N \l__stex_sproof_spf_id_str
4581   \tl_clear:N \l__stex_sproof_spf_display_tl
4582   \tl_clear:N \l__stex_sproof_spf_for_tl
4583   \tl_clear:N \l__stex_sproof_spf_from_tl
4584   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4585   \tl_clear:N \l__stex_sproof_spf_type_tl
4586   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4587 \tl_clear:N \l__stex_sproof_spf_continues_tl
4588 \tl_clear:N \l__stex_sproof_spf_functions_tl
4589 \tl_clear:N \l__stex_sproof_spf_method_tl
4590 \keys_set:nn { stex / spf }{ #1 }
4591 }

```

\spf@flow We define this macro, so that we can test whether the **display** key has the value **flow**

```

4592 \def\spf@flow{flow}

```

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

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.

pst@with@label This environment manages⁶ the path labeling of the proof steps in the description environment of the outermost **proof** environment. The argument is the label prefix up to now; which we cache in **\pst@label** (we need evaluate it first, since are in the right place now!). Then we increment the proof depth which is stored in **\count10** (lower counters are used by T_EX for page numbering) and initialize the next level counter **\count\count10** with 1. In the end call for this environment, we just decrease the proof depth counter by 1 again.

```

4593 \newcount\count_ten
4594 \newenvironment{pst@with@label}[1]{
4595   \edef\pst@label{#1}
4596   \advance\count_ten by 1\relax
4597   \count_ten=1
4598 }{
4599   \advance\count_ten by -1\relax
4600 }

```

\the@pst@label **\the@pst@label** evaluates to the current step label.

```

4601 \def\the@pst@label{
4602   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4603 }

```

(End definition for **\the@pst@label**. This function is documented on page ??.)

\setpstlabelstyle **\setpstlabelstyle{metaKey-Val pairs}** makes the labeling style customizable. **\setpstlabelstyle{pr}** will change the labeling style from **P.1.2.3** to **Pr-1-2-3†**. **\setpstlabelstyledefault** will set the labeling style back to default.

```

4604 \keys_define:nn { stex / pstlabel }{
4605   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4606   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4607   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4608 }
4609 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

⁶This gets the labeling right but only works 8 levels deep

```

4610 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4611 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4612 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4613 }
4614 \__stex_sproof_pstlabel_args:n {}
4615 \newcommand\setpstlabelstyle[1]{
4616   \__stex_sproof_pstlabel_args:n {#1}
4617 }
4618 \newcommand\setpstlabelstyledefault{%
4619   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4620 }

```

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

\pstlabelstyle \pstlabelstyle just sets the \pst@make@label macro according to the style.

```

4621 \ExplSyntaxOff
4622 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4623 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4624 \def\pst@make@label@short#1#2{#2}
4625 \def\pst@make@label@empty#1#2{}
4626 \ExplSyntaxOn
4627 \def\pstlabelstyle#1{%
4628   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4629 }%
4630 \pstlabelstyle{long}%

```

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

\next@pst@label \next@pst@label increments the step label at the current level.

```

4631 \def\next@pst@label{%
4632   \global\advance\count\count10 by 1%
4633 }%

```

(End definition for \next@pst@label. This function is documented on page ??.)

\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

```

4634 \def\sproof@box{
4635   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4636 }
4637 \def\spf@proofend{\sproof@box}
4638 \def\sproofend{
4639   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4640     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4641   }
4642 }
4643 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4644 \def\spf@proofsketch@kw{Proof Sketch}
4645 \def\spf@proof@kw{Proof}
4646 \def\spf@step@kw{Step}

```


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

For the other languages, we set up triggers

```

4647 \AddToHook{begindocument}{
4648   \ltx@ifpackageloaded{babel}{
4649     \makeatletter
4650     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4651     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4652       \input{sproof-ngerman.ldf}
4653     }
4654     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4655       \input{sproof-finnish.ldf}
4656     }
4657     \clist_if_in:NnT \l_tmpa_clist {french}{
4658       \input{sproof-french.ldf}
4659     }
4660     \clist_if_in:NnT \l_tmpa_clist {russian}{
4661       \input{sproof-russian.ldf}
4662     }
4663     \makeatother
4664   }{}
4665 }

```

`spfsketch`

```

4666 \newcommand\spfsketch[2][]{
4667   \__stex_sproof_spf_args:n{#1}
4668   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4669     \titleemph{
4670       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4671         \spf@proofsketch@kw
4672       }{
4673         \l__stex_sproof_spf_type_tl
4674       }
4675     }:
4676   }
4677   {~#2}
4678   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4679   \sproofend
4680 }

```

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

`spfeq` This is very similar to `\spfsketch`, but uses a computation array¹⁴¹⁵

```

4681 \newenvironment{spfeq}[2][]{
4682   \__stex_sproof_spf_args:n{#1}
4683   %\sref@target
4684   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4685     \titleemph{
4686       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4687         \spf@proof@kw
4688       }{

```

¹⁴EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

¹⁵EDNOTE: document above

```

4689     \l__stex_sproof_spf_type_tl
4690   }
4691   }:
4692 }
4693 {-#2}
4694 \begin{displaymath}\begin{array}{rcll}
4695 }{
4696   \end{array}\end{displaymath}
4697 }

```

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

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

```

4698 \newenvironment{spf@proof}[2] []{
4699   \l__stex_sproof_spf_args:n{#1}
4700   %\sref@target
4701   \count_ten=10
4702   \par\noindent
4703   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4704     \titleemph{
4705       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4706         \spf@proof@kw
4707       }{
4708         \l__stex_sproof_spf_type_tl
4709       }
4710     }:
4711   }
4712   {-#2}
4713   %\sref@label{id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4714   \def\pst@label{}
4715   \newcount\pst@count% initialize the labeling mechanism
4716   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4717   }{
4718     \end{pst@with@label}\end{description}
4719   }
4720   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4721   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4722 \newcommand\spfidea[2] []{
4723   \l__stex_sproof_spf_args:n{#1}
4724   \titleemph{
4725     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4726       \l__stex_sproof_spf_type_tl
4727     }:
4728   }-#2
4729   \sproofend
4730 }

```

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

The next two environments (proof steps) and comments, are mostly semantical, they take `KeyVal` arguments that specify their semantic role. In draft mode, they read these

values and show them. If the surrounding proof had `display=flow`, then no new `\item` is generated, otherwise it is. In any case, the proof step number (at the current level) is incremented.

EdN:16

```

spfstep 16
4731 \newenvironment{spfstep}[1][]{
4732   \_stex_sproof_spf_args:n{#1}
4733   \@in@omtexttrue
4734   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4735     \item[\the@pst@label]
4736   }
4737   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4738     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4739   }
4740   %\sref@label{id{\pst@label}}
4741   \ignorespacesandpars
4742 }{
4743   \next@pst@label\ignorespacesandpars
4744 }

```

sproofcomment

```

4745 \newenvironment{sproofcomment}[1][]{
4746   \_stex_sproof_spf_args:n{#1}
4747   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4748     \item[\the@pst@label]
4749   }
4750 }{
4751   \next@pst@label
4752 }

```

The next two environments also take a `KeyVal` argument, but also a regular one, which contains a start text. Both environments start a new numbered proof level.

subproof In the `subproof` environment, a new (lower-level) `proproofof` environment is started.

```

4753 \newenvironment{subproof}[2][]{
4754   \_stex_sproof_spf_args:n{#1}
4755   \def\@test{#2}
4756   \ifx\@test\empty\else
4757     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4758       \item[\the@pst@label]
4759     }{#2}
4760   \fi
4761   \begin{pst@with@label}{\pst@label,\number\count_ten}
4762 }{
4763   \end{pst@with@label}\next@pst@label
4764 }

```

spfcases In the `pfcases` environment, the start text is displayed as the first comment of the proof.

```

4765 \newenvironment{spfcases}[2][]{
4766   \def\@test{#1}
4767   \ifx\@test\empty
4768     \begin{subproof}[method=by-cases]{#2}

```

¹⁶EdNOTE: MK: labeling of steps does not work yet.

```

4769 \else
4770   \begin{subproof}[#1,method=by-cases]{#2}
4771 \fi
4772 }{
4773   \end{subproof}
4774 }

```

spfcase In the **pfcase** environment, the start text is displayed specification of the case after the **\item**

```

4775 \newenvironment{spfcase}[2] [] {
4776   \__stex_sproof_spf_args:n{#1}
4777   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4778     \item[\the@pst@label]
4779   }
4780   \def\@test{#2}
4781   \ifx\@test\@empty
4782   \else
4783     {\titleemph{#2}:~}
4784   \fi
4785   \begin{pst@with@label}{\pst@label,\number\count_ten}
4786 }{
4787   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4788     \sproofend
4789   }
4790   \end{pst@with@label}
4791   \next@pst@label
4792 }

```

spfcase similar to **spfcase**, takes a third argument.

```

4793 \newcommand\spfcasesketch[3] [] {
4794   \__stex_sproof_spf_args:n{#1}
4795   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4796     \item[\the@pst@label]
4797   }
4798   \def\@test{#2}
4799   \ifx\@test\@empty
4800   \else
4801     {\titleemph{#2}:~}
4802   \fi#3
4803   \next@pst@label
4804 }%

```

34.3 Justifications

We define the actions that are undertaken, when the keys for justifications are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```

4805 \keys_define:nn { stex / just }{
4806   id          .str_set_x:N = \l__stex_sproof_just_id_str,
4807   method      .tl_set:N   = \l__stex_sproof_just_method_tl,
4808   premises    .tl_set:N   = \l__stex_sproof_just_premises_tl,
4809   args        .tl_set:N   = \l__stex_sproof_just_args_tl
4810 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.¹⁷

`justification`

4811 `\newenvironment{justification}[1] [] {}{}`

`\premise`

4812 `\newcommand\premise[2] [] {#2}`

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

`\justarg`

the `\justarg` macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

4813 `\newcommand\justarg[2] [] {#2}`

4814 `\</package>`

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

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

¹⁷EdNOTE: need to do something about the premise in draft mode.

Chapter 35

STEX -Others Implementation

```
4815 <*package>
4816
4817 %%%%%%%%%% others.dtx %%%%%%%%%%
4818
4819 <@@=stex_others>
    Warnings and error messages
4820 % None

\MSC Math subject classifier

4821 \NewDocumentCommand \MSC {m} {
4822 % TODO
4823 }

(End definition for \MSC. This function is documented on page 21.)
    Patching tikzinput, if loaded
4824 \@ifpackageloaded{tikzinput}{
4825 \RequirePackage{stex-tikzinput}
4826 }{}
4827 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4828 \*package>
4829 \@@=stex_modules>
4830
4831 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4832
4833 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4834 \begingroup
4835 \stex_module_setup:nn{
4836   ns=\c_stex_metatheory_ns_str,
4837   meta=NONE
4838 }{Metatheory}
4839 \stex_reactivate_macro:N \symdecl
4840 \stex_reactivate_macro:N \notation
4841 \stex_reactivate_macro:N \symdef
4842 \ExplSyntaxOff
4843 \csname stex_suppress_html:n\endcsname{
4844   % is-a (a:A, a \in A, a is an A, etc.)
4845   \symdecl[args=ai]{isa}
4846   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
4847   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
4848   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
4849
4850   % bind (\forall, \Pi, \lambda etc.)
4851   \symdecl[args=Bi]{bind}
4852   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
4853   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
4854   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
4855
4856   % dummy variable
4857   \symdecl{dummyvar}
4858   \notation[underscore]{dummyvar}{\comp\_}
4859   \notation[dot]{dummyvar}{\comp\cdot}
4860   \notation[dash]{dummyvar}{\comp{\rm --}}
4861
4862   %fromto (function space, Hom-set, implication etc.)
```

```

4863 \symdecl[args=ai]{fromto}
4864 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
4865 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
4866
4867 % mapto (lambda etc.)
4868 %\symdecl[args=Bi]{mapto}
4869 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4870 %\notation[lambda]{mapto}{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
4871 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
4872
4873 % function/operator application
4874 \symdecl[args=ia]{apply}
4875 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
4876 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
4877
4878 % ‘type’ of all collections (sets, classes, types, kinds)
4879 \symdecl{collection}
4880 \notation[U]{collection}{\comp{\mathcal{U}}}
4881 \notation[set]{collection}{\comp{\textsf{Set}}}
4882
4883 % sequences
4884 \symdecl[args=1]{seqtype}
4885 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4886
4887 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1}_{#2}
4888 \notation[ui,prec=nobrackets]{sequence-index}{#1}^{#2}
4889
4890 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{,\ellipses}}{##1\comp,##2}
4891 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{,\ellipses},#2}{##1\comp,##2}
4892 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{,\ellipses},#2\comp{,\ellipses},#3}
4893
4894 % letin (‘let’, local definitions, variable substitution)
4895 \symdecl[args=bii]{letin}
4896 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2;\comp{\rm in}}{#3}
4897 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4898 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4899
4900 % structures
4901 \symdecl*[args=1]{module-type}
4902 \notation{module-type}{\mathtt{MOD} #1}
4903 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4904 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
4905
4906 }
4907 \ExplSyntaxOn
4908 \stex_add_to_current_module:n{
4909   \let\nappa\apply
4910   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4911   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4912   \def\livar{\csname sequence-index\endcsname[li]}
4913   \def\uivar{\csname sequence-index\endcsname[ui]}
4914   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4915   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4916   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}

```



```
4917 }  
4918 \_stex_modules_end_module:  
4919 \endgroup  
4920 </package>
```

Chapter 37

Tikzinput Implementation

```
4921 <*package>
4922
4923 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4924
4925 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4926 \RequirePackage{l3keys2e}
4927
4928 \keys_define:nn { tikzinput } {
4929   image .bool_set:N = \c_tikzinput_image_bool,
4930   image .default:n = false ,
4931   unknown .code:n = {}
4932 }
4933
4934 \ProcessKeysOptions { tikzinput }
4935
4936 \bool_if:NTF \c_tikzinput_image_bool {
4937   \RequirePackage{graphicx}
4938
4939   \providecommand\usetikzlibrary[]{}
4940   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4941 }{
4942   \RequirePackage{tikz}
4943   \RequirePackage{standalone}
4944
4945   \newcommand \tikzinput [2] [] {
4946     \setkeys{Gin}{#1}
4947     \ifx \Gin@ewidth \Gin@exclamation
4948       \ifx \Gin@eheight \Gin@exclamation
4949         \input { #2 }
4950       \else
4951         \resizebox{!}{ \Gin@eheight }{
4952           \input { #2 }
4953         }
4954       \fi
4955     \else
4956       \ifx \Gin@eheight \Gin@exclamation
4957         \resizebox{ \Gin@ewidth }{!}{
4958           \input { #2 }
```

```

4959     }
4960     \else
4961         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4962             \input { #2 }
4963         }
4964     \fi
4965 \fi
4966 }
4967 }
4968
4969 \newcommand \ctikzinput [2] [] {
4970     \begin{center}
4971         \tikzinput [ #1 ] { #2 }
4972     \end{center}
4973 }
4974
4975 \@ifpackageloaded{stex}{
4976     \RequirePackage{stex-tikzinput}
4977 }{}
4978
4979 </package>
4980 <*stex>
4981 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4982 \RequirePackage{stex}
4983 \RequirePackage{tikzinput}
4984
4985 \newcommand\mhtikzinput [2] [] {%
4986     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4987     \stex_in_repository:nn\Gin@mhrepos{
4988         \tikzinput [ #1 ] {\mhp{##1}{#2}}
4989     }
4990 }
4991 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
4992 </stex>

```

LocalWords: bibfolder jobname.dtx tikzinput.dtx usetikzlibrary Gin@ewidth Gin@eheight
LocalWords: resizebox ctikzinput mhtikzinput Gin@mhrepos mhp{

Chapter 38

document-structure.sty Implementation

38.1 The document-structure Class

The functionality is spread over the `document-structure` class and package. The class provides the `document` environment and the `document-structure` element corresponds to it, whereas the package provides the concrete functionality.

```
4993 \*cls)
4994 \@@=document_structure)
4995 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
4996 \RequirePackage{l3keys2e,expl-keystr-compatible}
```

38.2 Class Options

To initialize the `document-structure` class, we declare and process the necessary options using the `kvoptions` package for key/value options handling. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@cls@class` macro and pass on the macro to `omdoc.sty` for further processing.

`\omdoc@cls@class`

```
4997 \keys_define:nn{ document-structure / pkg }{
4998   class      .str_set_x:N = \c_document_structure_class_str,
4999   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5000   report     .code:n      = {
5001     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5002     \str_set:Nn \c_document_structure_class_str {report}
5003   },
5004   book       .code:n      = {
5005     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5006     \str_set:Nn \c_document_structure_class_str {book}
5007   },
5008   bookpart   .code:n      = {
5009     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5010     \str_set:Nn \c_document_structure_class_str {book}
5011     \str_set:Nn \c_document_structure_topsect_str {chapter}
5012   },
```

```

5013 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5014 unknown     .code:n      = {
5015   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5016 }
5017 }
5018 \ProcessKeysOptions{ document-structure / pkg }
5019 \str_if_empty:NT \c_document_structure_class_str {
5020   \str_set:Nn \c_document_structure_class_str {article}
5021 }
5022 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5023   {\c_document_structure_class_str}
5024

```

38.3 Beefing up the document environment

Now, – unless the option `minimal` is defined – we include the `stex` package

```

5025 \RequirePackage{document-structure}
5026 \bool_if:NF \c_document_structure_minimal_bool {

```

And define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

document For the moment we do not use them on the L^AT_EX level, but the document identifier is picked up by L^AT_EXML.¹⁸

```

5027 \keys_define:nn { document-structure / document }{
5028   id .str_set_x:N = \c_document_structure_document_id_str
5029 }
5030 \let\__document_structure_orig_document=\document
5031 \renewcommand{\document}[1][]{
5032   \keys_set:nn{ document-structure / document }{ #1 }
5033   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5034   \__document_structure_orig_document
5035 }

```

Finally, we end the test for the `minimal` option.

```

5036 }
5037 </cls>

```

38.4 Implementation: document-structure Package

```

5038 <*package>
5039 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5040 \RequirePackage{expl-keystr-compat,13keys2e}

```

38.5 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).

¹⁸EDNOTE: faking documentkeys for now. @HANG, please implement

```

5041
5042 \keys_define:nn{ document-structure / pkg }{
5043   class      .str_set_x:N = \c_document_structure_class_str,
5044   topsect     .str_set_x:N = \c_document_structure_topsect_str,
5045   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5046 }
5047 \ProcessKeysOptions{ document-structure / pkg }
5048 \str_if_empty:NT \c_document_structure_class_str {
5049   \str_set:Nn \c_document_structure_class_str {article}
5050 }
5051 \str_if_empty:NT \c_document_structure_topsect_str {
5052   \str_set:Nn \c_document_structure_topsect_str {section}
5053 }

```

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

```

5054 \RequirePackage{xspace}
5055 \RequirePackage{comment}
5056 \AddToHook{begindocument}{
5057   \ltx@ifpackageloaded{babel}{
5058     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5059     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5060       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5061     }
5062   }{}
5063 }

```

`\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.

```

5064 \int_new:N \l_document_structure_section_level_int
5065 \str_case:VnF \c_document_structure_topsect_str {
5066   {part}}{
5067     \int_set:Nn \l_document_structure_section_level_int {0}
5068   }
5069   {chapter}}{
5070     \int_set:Nn \l_document_structure_section_level_int {1}
5071   }
5072 }{
5073   \str_case:VnF \c_document_structure_class_str {
5074     {book}}{
5075       \int_set:Nn \l_document_structure_section_level_int {0}
5076     }
5077     {report}}{
5078       \int_set:Nn \l_document_structure_section_level_int {0}
5079     }
5080   }{
5081     \int_set:Nn \l_document_structure_section_level_int {2}
5082   }
5083 }

```

38.6 Document Structure

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

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.¹⁹

EdN:19

```
5084 \def\current@section@level{document}%
5085 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5086 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5087 \cs_new_protected:Npn \skipomgroup {
5088   \ifcase\l_document_structure_section_level_int
5089   \or\stepcounter{part}
5090   \or\stepcounter{chapter}
5091   \or\stepcounter{section}
5092   \or\stepcounter{subsection}
5093   \or\stepcounter{subsubsection}
5094   \or\stepcounter{paragraph}
5095   \or\stepcounter{subparagraph}
5096   \fi
5097 }
```

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

`blindomgroup`

```
5098 \newcommand\at@begin@blindomgroup[1]{%
5099 \newenvironment{blindomgroup}
5100 {
5101   \int_incr:N\l_document_structure_section_level_int
5102   \at@begin@blindomgroup\l_document_structure_section_level_int
5103 }{}}
```

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

```
5104 \newcommand\omgroup@nonum[2]{
5105   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5106   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5107 }
```

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

`\omgroup@num` convenience macro: `\omgroup@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `omgroup` 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.

```
5108 \newcommand\omgroup@num[2]{
```

¹⁹EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

5109 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5110   \@nameuse{#1}{#2}
5111 }{
5112   \cs_if_exist:NTF\rdfmata@sectioning{
5113     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5114   }{
5115     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5116   }
5117 }
5118 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5119 }

```

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

omgroup

```

5120 \keys_define:nn { document-structure / omgroup }{
5121   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5122   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5123   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5124   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5125   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5126   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5127   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5128   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5129   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5130   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5131 }
5132 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5133   \str_clear:N \l__document_structure_omgroup_id_str
5134   \str_clear:N \l__document_structure_omgroup_date_str
5135   \clist_clear:N \l__document_structure_omgroup_creators_clist
5136   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5137   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5138   \tl_clear:N \l__document_structure_omgroup_type_tl
5139   \tl_clear:N \l__document_structure_omgroup_short_tl
5140   \tl_clear:N \l__document_structure_omgroup_display_tl
5141   \tl_clear:N \l__document_structure_omgroup_intro_tl
5142   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5143   \keys_set:nn { document-structure / omgroup } { #1 }
5144 }

```

we define a switch for numbering lines and a hook for the beginning of groups: The \at@begin@omgroup macro allows customization. It is run at the beginning of the omgroup, i.e. after the section heading.

```

5145 \newif\if@mainmatter\@mainmattertrue
5146 \newcommand\at@begin@omgroup[3] [] {}

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

5147 \keys_define:nn { document-structure / sectioning }{
5148   name .str_set_x:N = \l__document_structure_sect_name_str ,
5149   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5150   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5151   num .bool_set:N = \l__document_structure_sect_num_bool ,
5152 }

```



```

5153 \cs_new_protected:Nn \__document_structure_sect_args:n {
5154   \str_clear:N \l__document_structure_sect_name_str
5155   \str_clear:N \l__document_structure_sect_ref_str
5156   \bool_set_false:N \l__document_structure_sect_clear_bool
5157   \bool_set_false:N \l__document_structure_sect_num_bool
5158   \keys_set:nn { document-structure / sectioning } { #1 }
5159 }
5160 \newcommand\omdoc@sectioning[3][]{
5161   \__document_structure_sect_args:n {#1}
5162   \let\omdoc@sect@name\l__document_structure_sect_name_str
5163   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5164   \if@mainmatter% numbering not overridden by frontmatter, etc.
5165     \bool_if:NTF \l__document_structure_sect_num_bool {
5166       \omgroup@num{#2}{#3}
5167     }{
5168       \omgroup@nonum{#2}{#3}
5169     }
5170     \def\current@section@level{\omdoc@sect@name}
5171   \else
5172     \omgroup@nonum{#2}{#3}
5173   \fi
5174 }% 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.

```

5175 \newcommand\omgroup@redefine@addtocontents[1]{%
5176   %\edef\__document_structureimport{#1}%
5177   %\@for\@I:=\__document_structureimport\do{%
5178     %\edef\@path{\csname module@\@I @path\endcsname}%
5179     %\@ifundefined{tf@toc}\relax%
5180     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5181   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
5182   %\def\addcontentsline##1##2##3{%
5183     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5184   %\else% hyperref.sty not loaded
5185   %\def\addcontentsline##1##2##3{%
5186     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
5187   %\fi
5188 }% hyperref.sty loaded?

```

now the `omgroup` 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 `omgroups` in the `\omgroup@level` counter.

```

5189 \int_new:N \l_document_structure_omgroup_level_int
5190 \newenvironment{omgroup}[2][]{% keys, title
5191 {
5192   \__document_structure_omgroup_args:n { #1 }%\sref@target%

```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

5193 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5194   \omgroup@redefine@addtocontents{
5195     %\@ifundefined{module@id}\used@modules%
5196     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

5197     }
5198 }

```

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

```

5199 \int_incr:N \l_document_structure_omgroup_level_int
5200 \int_incr:N \l_document_structure_section_level_int
5201 \ifcase\l_document_structure_section_level_int
5202   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5203   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5204   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5205   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5206   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5207   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5208   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5209 \fi
5210 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5211 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5212 }% for customization
5213 {}

```

and finally, we localize the sections

```

5214 \newcommand\omdoc@part@kw{Part}
5215 \newcommand\omdoc@chapter@kw{Chapter}
5216 \newcommand\omdoc@section@kw{Section}
5217 \newcommand\omdoc@subsection@kw{Subsection}
5218 \newcommand\omdoc@subsubsection@kw{Subsubsection}
5219 \newcommand\omdoc@paragraph@kw{paragraph}
5220 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

38.7 Front and Backmatter

Index markup is provided by the `omtext` package [Koh20c], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

5221 \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).

```

5222 \cs_if_exist:NTF\frontmatter{
5223   \let\__document_structure_orig_frontmatter\frontmatter
5224   \let\frontmatter\relax
5225 }{
5226   \tl_set:Nn\__document_structure_orig_frontmatter{
5227     \clearpage
5228     \@mainmatterfalse
5229     \pagenumbering{roman}
5230   }
5231 }

```

```

5232 \cs_if_exist:NTF\backmatter{
5233   \let\__document_structure_orig_backmatter\backmatter
5234   \let\backmatter\relax
5235 }{
5236   \tl_set:Nn\__document_structure_orig_backmatter{
5237     \clearpage
5238     \@mainmatterfalse
5239     \pagenumbering{roman}
5240   }
5241 }

```

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

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

```

5242 \newenvironment{frontmatter}{
5243   \__document_structure_orig_frontmatter
5244 }{
5245   \cs_if_exist:NTF\mainmatter{
5246     \mainmatter
5247   }{
5248     \clearpage
5249     \@mainmattertrue
5250     \pagenumbering{arabic}
5251   }
5252 }

```

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

```

5253 \newenvironment{backmatter}{
5254   \__document_structure_orig_backmatter
5255 }{
5256   \cs_if_exist:NTF\mainmatter{
5257     \mainmatter
5258   }{
5259     \clearpage
5260     \@mainmattertrue
5261     \pagenumbering{arabic}
5262   }
5263 }

```

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

```

5264 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5265 \def \c__document_structure_document_str{document}
5266 \newcommand\afterprematurestop{}
5267 \def\prematurestop@endomgroup{
5268   \unless\ifx\@currenvir\c__document_structure_document_str
5269     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter
5270     \expandafter\prematurestop@endomgroup
5271   \fi
5272 }

```

```

5273 \providecommand\prematurestop{
5274   \message{Stopping~sTeX~processing~prematurely}
5275   \prematurestop@endomgroup
5276   \afterprematurestop
5277   \end{document}
5278 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

5279 \RequirePackage{etoolbox}
5280 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

5281 \newrobustcmd\useSGvar[1]{%
5282   \@ifundefined{sTeX@Gvar@#1}
5283   {\PackageError{document-structure}
5284     {The sTeX Global variable #1 is undefined}
5285     {set it with \protect\setSGvar}}
5286   \@nameuse{sTeX@Gvar@#1}}

```

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

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

```

5287 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5288   \@ifundefined{sTeX@Gvar@#1}
5289   {\PackageError{document-structure}
5290     {The sTeX Global variable #1 is undefined}
5291     {set it with \protect\setSGvar}}
5292   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

Chapter 39

NotesSlides – Implementation

39.1 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.

```
5293 \*cls)
5294 \@@=notesslides)
5295 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5296 \RequirePackage{l3keys2e,expl-keystr-compatible}
5297
5298 \keys_define:nn{notesslides / cls}{
5299   class .code:n = {
5300     \PassOptionsToClass{\CurrentOption}{omdoc}
5301     \str_if_eq:nnT{#1}{book}{
5302       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5303     }
5304     \str_if_eq:nnT{#1}{report}{
5305       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5306     }
5307   },
5308   notes .bool_set:N = \c__notesslides_notes_bool ,
5309   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5310   unknown .code:n = {
5311     \PassOptionsToClass{\CurrentOption}{omdoc}
5312     \PassOptionsToClass{\CurrentOption}{beamer}
5313     \PassOptionsToPackage{\CurrentOption}{notesslides}
5314   }
5315 }
5316 \ProcessKeysOptions{ notesslides / cls }
5317 \bool_if:NTF \c__notesslides_notes_bool {
5318   \PassOptionsToPackage{notes=true}{notesslides}
5319 }{
5320   \PassOptionsToPackage{notes=false}{notesslides}
5321 }
5322 \</cls)
```

now we do the same for the notesslides package.

```

5323 <*package>
5324 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5325 \RequirePackage{l3keys2e,expl-keystr-compat}
5326
5327 \keys_define:nn{notesslides / pkg}{
5328   topsect          .str_set_x:N = \c__notesslides_topsect_str,
5329   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
5330   notes            .bool_set:N = \c__notesslides_notes_bool ,
5331   slides           .code:n       = { \bool_set_false:N \c__notesslides_notes_bool },
5332   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
5333   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
5334   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
5335   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
5336   unknown          .code:n       = {
5337     \PassOptionsToClass{\CurrentOption}{stex}
5338     \PassOptionsToClass{\CurrentOption}{tikzinput}
5339   }
5340 }
5341 \ProcessKeysOptions{ notesslides / pkg }
5342 \newif\ifnotes
5343 \bool_if:NTF \c__notesslides_notes_bool {
5344   \notesttrue
5345 }{
5346   \notesfalse
5347 }
5348

```

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

```

5349 \str_if_empty:NTF \c__notesslides_topsect_str {
5350   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5351 }{
5352   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
5353 }
5354 </package>

```

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

```

5355 <*cls>
5356 \bool_if:NTF \c__notesslides_notes_bool {
5357   \LoadClass{document-structure}
5358 }{
5359   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5360   \newcounter{Item}
5361   \newcounter{paragraph}
5362   \newcounter{subparagraph}
5363   \newcounter{Hfootnote}
5364   \RequirePackage{document-structure}
5365 }

```

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

```

5366 \RequirePackage{notesslides}
5367 </cls>

```

In `notes` mode, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the `STEX` packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the `STEX`-specific ones after we have done some work (e.g. defined the counters `m*`). Only the `stex-logo` package is already needed now for the default theme.

```

5368 \*package>
5369 \bool_if:NT \c__notesslides_notes_bool {
5370   \RequirePackage{a4wide}
5371   \RequirePackage{marginnote}
5372   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5373   \RequirePackage{mdframed}
5374   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5375   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5376 }
5377 \RequirePackage{stex-tikzinput}
5378 \RequirePackage{etoolbox}
5379 \RequirePackage{amssymb}
5380 \RequirePackage{amsmath}
5381 \RequirePackage{comment}
5382 \RequirePackage{textcomp}
5383 \RequirePackage{url}
5384 \RequirePackage{graphicx}
5385 \RequirePackage{pgf}

```

39.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.²⁰

```

5386 \bool_if:NT \c__notesslides_notes_bool {
5387   \renewcommand\usetheme[2][\]{\usepackage[#1]{beamernotestheme#2}}
5388 }

```

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

```

5389 \newcounter{slide}
5390 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5391 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

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

```

5392 \bool_if:NTF \c__notesslides_notes_bool {
5393   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
5394 }{
5395   \excludecomment{note}
5396 }

```

²⁰EdNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

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.

```
5397 \bool_if:NT \c__notesslides_notes_bool {
5398   \newlength{\slideframewidth}
5399   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
5400 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5401   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5402     \bool_set_true:N #1
5403   }{
5404     \bool_set_false:N #1
5405   }
5406 }
5407 \keys_define:nn{notesslides / frame}{
5408   label .str_set_x:N = \l__notesslides_frame_label_str,
5409   allowframebreaks .code:n = {
5410     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5411   },
5412   allowdisplaybreaks .code:n = {
5413     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5414   },
5415   fragile .code:n = {
5416     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5417   },
5418   shrink .code:n = {
5419     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5420   },
5421   squeeze .code:n = {
5422     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5423   },
5424   t .code:n = {
5425     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5426   },
5427 }
5428 \cs_new_protected:Nn \__notesslides_frame_args:n {
5429   \str_clear:N \l__notesslides_frame_label_str
5430   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5431   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5432   \bool_set_true:N \l__notesslides_frame_fragile_bool
5433   \bool_set_true:N \l__notesslides_frame_shrink_bool
5434   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5435   \bool_set_true:N \l__notesslides_frame_t_bool
5436   \keys_set:nn { notesslides / frame }{ #1 }
5437 }
```

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

```
5438 \renewenvironment{frame}[1][]{
5439   \__notesslides_frame_args:n{#1}
5440   \sffamily
5441   \stepcounter{slide}
5442   \def\@currentlabel{\theslide}
5443   \str_if_empty:NF \l__notesslides_frame_label_str {
5444     \label{\l__notesslides_frame_label_str}
```


5445 }

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

```

5446 \def\itemize@level{outer}
5447 \def\itemize@outer{outer}
5448 \def\itemize@inner{inner}
5449 \renewcommand\newpage{\addtocounter{framenum}{1}}
5450 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5451 \renewenvironment{itemize}{
5452   \ifx\itemize@level\itemize@outer
5453     \def\itemize@label{$\rhd$}
5454   \fi
5455   \ifx\itemize@level\itemize@inner
5456     \def\itemize@label{$\scriptstyle\rhd$}
5457   \fi
5458   \begin{list}
5459   {\itemize@label}
5460   {\setlength{\labelsep}{.3em}
5461    \setlength{\labelwidth}{.5em}
5462    \setlength{\leftmargin}{1.5em}
5463   }
5464   \edef\itemize@level{\itemize@inner}
5465 }{
5466   \end{list}
5467 }

```

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

```

5468 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
5469 }{
5470   \medskip\miko@slidelabel\end{mdframed}
5471 }

```

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

`\frametitle`

```

5472 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5473 }

```

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

EdN:21

`\pause` 21

```

5474 \bool_if:NT \c__notesslides_notes_bool {
5475   \newcommand\pause{}
5476 }

```

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

`nparagraph`

```

5477 \bool_if:NTF \c__notesslides_notes_bool {
5478   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}{\end{sparagraph}}
5479 }{
5480   \excludecomment{nparagraph}
5481 }

```

²¹EdNOTE: MK: fake it in notes mode for now

```

nomgroup
5482 \bool_if:NTF \c__notesslides_notes_bool {
5483   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5484 }{
5485   \excludecomment{nomgroup}
5486 }

ntheorem
5487 \bool_if:NTF \c__notesslides_notes_bool {
5488   \newenvironment{ntheorem}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5489 }{
5490   \excludecomment{ntheorem}
5491 }

nassertion
5492 \bool_if:NTF \c__notesslides_notes_bool {
5493   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5494 }{
5495   \excludecomment{nassertion}
5496 }

nsproof
5497 \bool_if:NTF \c__notesslides_notes_bool {
5498   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5499 }{
5500   \excludecomment{nsproof}
5501 }

nexample
5502 \bool_if:NTF \c__notesslides_notes_bool {
5503   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5504 }{
5505   \excludecomment{nexample}
5506 }

\inputref@*skip We customize the hooks for in \inputref.
5507 \def\inputref@preskip{\smallskip}
5508 \def\inputref@postskip{\medskip}

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

\inputref*
5509 \let\orig@inputref\inputref
5510 \def\inputref{\@ifstar\ninputref\orig@inputref}
5511 \newcommand\ninputref[2] [] {
5512   \bool_if:NT \c__notesslides_notes_bool {
5513     \orig@inputref[#1]{#2}
5514   }
5515 }

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

```

39.3 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>}`.

```
5516 \newlength{\slidelogoheight}
5517
5518 \bool_if:NTF \c__notesslides_notes_bool {
5519   \setlength{\slidelogoheight}{.4cm}
5520 }{
5521   \setlength{\slidelogoheight}{1cm}
5522 }
5523 \newsavebox{\slidelogo}
5524 \sbox{\slidelogo}{\TeX}
5525 \newrobustcmd{\setslidelogo}[1]{
5526   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5527 }
```

(End definition for `\setslidelogo`. 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.

```
5528 \def\source{Michael Kohlhase}% customize locally
5529 \newrobustcmd{\setsource}[1]{\def\source{#1}}
```

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

\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.

```
5530 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5531 \newsavebox{\cclogo}
5532 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5533 \newif\ifcchref\cchreffalse
5534 \AtBeginDocument{
5535   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5536 }
5537 \def\licensing{
5538   \ifcchref
5539     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5540   \else
5541     {\usebox{\cclogo}}
5542   \fi
5543 }
5544 \newrobustcmd{\setlicensing}[2][]{
5545   \def@url{#1}
5546   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5547   \ifx@url@empty
5548     \def\licensing{{\usebox{\cclogo}}}
5549   \else
5550     \def\licensing{
```

```

5551     \ifcchref
5552     \href{#1}{\usebox{\cclogo}}
5553     \else
5554     {\usebox{\cclogo}}
5555     \fi
5556   }
5557 \fi
5558 }

```

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

EdN:22

`\slidelabel` Now, we set up the slide label for the article mode.²²

```

5559 \newrobustcmd\miko@slidelabel{
5560   \vbox to \slidelogoheight{
5561     \vss\hbox to \slidewidth
5562     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5563   }
5564 }

```

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

39.4 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.

```

5565 \def\Gin@mhrepos{}
5566 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5567 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
5568 \newrobustcmd\frameimage[2][]{
5569   \stepcounter{slide}
5570   \bool_if:NT \c__notesslides_frameimages_bool {
5571     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5572     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5573     \begin{center}
5574       \bool_if:NTF \c__notesslides_fiboxed_bool {
5575         \fbox{
5576           \ifx\Gin@ewidth\@empty
5577             \ifx\Gin@mhrepos\@empty
5578               \mhgraphics[width=\slidewidth,#1]{#2}
5579             \else
5580               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5581             \fi
5582           \else% Gin@ewidth empty
5583             \ifx\Gin@mhrepos\@empty
5584               \mhgraphics[#1]{#2}
5585             \else
5586               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5587             \fi
5588           \fi% Gin@ewidth empty
5589         }
5590       }{
5591         \ifx\Gin@ewidth\@empty

```

²²EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

5592         \ifx\Gin@mhrepos\empty
5593             \mhgraphics[width=\slidewidth,#1]{#2}
5594         \else
5595             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5596         \fi
5597         \ifx\Gin@mhrepos\empty
5598             \mhgraphics[#1]{#2}
5599         \else
5600             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5601         \fi
5602     \fi% Gin@ewidth empty
5603 }
5604 \end{center}
5605 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5606 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5607 }
5608 } % ifmks@sty@frameimages

```

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

39.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

5609 \sffamily

```

Now, we set up an infrastructure for highlighting phrases in slides. Note that we use content-oriented macros for highlighting rather than directly using color markup. The first thing to do is to adapt the green so that it is dark enough for most beamers

```

5610 \AddToHook{begindocument}{
5611     \definecolor{green}{rgb}{0,.5,0}
5612     \definecolor{purple}{cmyk}{.3,1,0,.17}
5613 }

```

We customize the `\defemph`, `\symrefemph`, `\compemph`, and `\titleemph` macros with colors. Furthermore we customize the `__omtextlec` macro for the appearance of line end comments in `\lec`.

```

5614 % \def\STpresent#1{\textcolor{blue}{#1}}
5615 \def\defemph#1{\textcolor{magenta}{#1}}
5616 \def\symrefemph#1{\textcolor{cyan}{#1}}
5617 \def\compemph#1{\textcolor{blue}{#1}}
5618 \def\titleemph#1{\textcolor{blue}{#1}}
5619 \def\__omtext_lec#1{\textcolor{green}{#1}}

```

I like to use the dangerous bend symbol for warnings, so we provide it here.

`\textwarning` as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```

5620 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5621 \def\smalltextwarning{
5622     \pgfuseimage{miko@small@dbend}
5623     \xspace
5624 }
5625 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5626 \newrobustcmd\textwarning{
5627   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5628   \xspace
5629 }
5630 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5631 \newrobustcmd\bigtextwarning{
5632   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5633   \xspace
5634 }

```

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

```

5635 \newrobustcmd\putgraphicsat[3]{
5636   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5637 }
5638 \newrobustcmd\putat[2]{
5639   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5640 }

```

39.6 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`.

```

5641 \bool_if:NT \c__notesslides_sectocframes_bool {
5642   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5643     \newcounter{chapter}\counterwithin*{section}{chapter}
5644   }{
5645     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5646       \newcounter{chapter}\counterwithin*{section}{chapter}
5647     }
5648   }
5649 }

```

`\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
5650 \def\part@prefix{}
5651 \@ifpackageloaded{document-structure}{\{
5652   \str_case:VnF \__notesslidesstopsect {
5653     {part}{
5654       \int_set:Nn \l_document_structure_section_level_int {0}
5655       \def\thesection{\arabic{chapter}.\arabic{section}}
5656       \def\part@prefix{\arabic{chapter}.}
5657     }
5658     {chapter}{
5659       \int_set:Nn \l_document_structure_section_level_int {1}
5660       \def\thesection{\arabic{chapter}.\arabic{section}}
5661       \def\part@prefix{\arabic{chapter}.}
5662     }
5663   }{
5664     \int_set:Nn \l_document_structure_section_level_int {2}
5665     \def\part@prefix{}

```

```

5666 }
5667 }
5668
5669 \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 omgroup environment that choses the L^AT_EX sectioning macros according to \section@level.

omgroup

```

5670 \renewenvironment{omgroup}[2][]{
5671   \__document_structure_omgroup_args:n { #1 }
5672   \int_incr:N \l_document_structure_omgroup_level_int
5673   \int_incr:N \l_document_structure_section_level_int
5674   \bool_if:NT \c__notesslides_sectocframes_bool {
5675     \stepcounter{slide}
5676     \begin{frame}[noframenumbering]
5677       \vfill\Large\centering
5678       \red{
5679         \ifcase\l_document_structure_section_level_int\or
5680           \stepcounter{part}
5681           \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5682           \def\currentsectionlevel{\omdoc@part@kw}
5683         \or
5684           \stepcounter{chapter}
5685           \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5686           \def\currentsectionlevel{\omdoc@chapter@kw}
5687         \or
5688           \stepcounter{section}
5689           \def\__notesslideslabel{\part@prefix\arabic{section}}
5690           \def\currentsectionlevel{\omdoc@section@kw}
5691         \or
5692           \stepcounter{subsection}
5693           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5694           \def\currentsectionlevel{\omdoc@subsection@kw}
5695         \or
5696           \stepcounter{subsubsection}
5697           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5698           \def\currentsectionlevel{\omdoc@subsubsection@kw}
5699         \or
5700           \stepcounter{paragraph}
5701           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
5702           \def\currentsectionlevel{\omdoc@paragraph@kw}
5703         \else
5704           \def\__notesslideslabel{}
5705           \def\currentsectionlevel{\omdoc@paragraph@kw}
5706         \fi% end ifcase
5707         \__notesslideslabel%\sref@label@id\__notesslideslabel
5708         \quad #2%
5709       }%
5710       \vfill%
5711       \end{frame}%
5712     }
5713     \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

5714 }{}
5715 }

```

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

```

5716 \def\inserttheorembodyfont{\normalfont}
5717 %\bool_if:NF \c__notesslides_notes_bool {
5718 % \defbeamertemplate{theorem begin}{miko}
5719 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5720 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
5721 % \inserttheorempunctuation\inserttheorembodyfont\hspace}
5722 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

5723 % \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.

```

5724 % \expandafter\def\csname Parent2\endcsname{}
5725 %}
5726
5727 \AddToHook{begindocument}{% this does not work for some reasons
5728 \setbeamertemplate{theorems}[ams style]
5729 }
5730 \bool_if:NT \c__notesslides_notes_bool {
5731 \renewenvironment{columns}[1][{}]{%
5732 \par\noindent%
5733 \begin{minipage}%
5734 \slidewidth\centering\leavevmode%
5735 }{}%
5736 \end{minipage}\par\noindent%
5737 }%
5738 \newsavebox\columnbox%
5739 \renewenvironment<>{column}[2][{}]{%
5740 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5741 }{}%
5742 \end{minipage}\end{lrbox}\usebox\columnbox%
5743 }%
5744 }
5745 \bool_if:NTF \c__notesslides_noproblems_bool {
5746 \newenvironment{problems}{}{}
5747 }{
5748 \excludecomment{problems}
5749 }

```

39.7 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.

```

5750 \gdef\printexcursions{}
5751 \newcommand\excursionref[2]{% label, text
5752 \bool_if:NT \c__notesslides_notes_bool {

```



```

5753 \begin{sparagraph}[title=Excursion]
5754 #2 \sref[fallback=the appendix]{#1}.
5755 \end{sparagraph}
5756 }
5757 }
5758 \newcommand\activate@excursion[2][]{
5759 \gappto\printexcursions{\inputref{#1}{#2}}
5760 }
5761 \newcommand\excursion[4][]{% repos, label, path, text
5762 \bool_if:NT \c__notesslides_notes_bool {
5763 \activate@excursion[1]{#3}\excursionref{#2}{#4}
5764 }
5765 }

```

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

\excursiongroup

```

5766 \keys_define:nn{notesslides / excursiongroup }{
5767 id .str_set_x:N = \l__notesslides_excursion_id_str,
5768 intro .tl_set:N = \l__notesslides_excursion_intro_tl,
5769 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5770 }
5771 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5772 \tl_clear:N \l__notesslides_excursion_intro_tl
5773 \str_clear:N \l__notesslides_excursion_id_str
5774 \str_clear:N \l__notesslides_excursion_mhrepos_str
5775 \keys_set:nn {notesslides / excursiongroup }{ #1 }
5776 }
5777 \newcommand\excursiongroup[1][]{
5778 \__notesslides_excursion_args:n{ #1 }
5779 \ifdefempty\printexcursions{}% only if there are excursions
5780 {\begin{note}
5781 \begin{omgroup}[1]{Excursions}%
5782 \ifdefempty\l__notesslides_excursion_intro_tl{\{
5783 \inputref[\l__notesslides_excursion_mhrepos_str]{
5784 \l__notesslides_excursion_intro_tl
5785 }
5786 }
5787 \printexcursions%
5788 \end{omgroup}
5789 \end{note}}
5790 }
5791 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
5792 \</package>

```

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

Chapter 40

The Implementation

40.1 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.

```
5793 <*package>
5794 <@@=problems>
5795 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5796 \RequirePackage{l3keys2e,expl-keystr-compat}
5797
5798 \keys_define:nn { problem / pkg }{
5799   notes      .default:n    = { true },
5800   notes      .bool_set:N   = \c__problems_notes_bool,
5801   gnotes     .default:n    = { true },
5802   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
5803   hints      .default:n    = { true },
5804   hints      .bool_set:N   = \c__problems_hints_bool,
5805   solutions  .default:n    = { true },
5806   solutions  .bool_set:N   = \c__problems_solutions_bool,
5807   pts        .default:n    = { true },
5808   pts        .bool_set:N   = \c__problems_pts_bool,
5809   min        .default:n    = { true },
5810   min        .bool_set:N   = \c__problems_min_bool,
5811   boxed      .default:n    = { true },
5812   boxed      .bool_set:N   = \c__problems_boxed_bool,
5813   unknown    .code:n       = {}
5814 }
5815 \newif\ifsolutions
5816
5817 \ProcessKeysOptions{ problem / pkg }
5818 \bool_if:NTF \c__problems_solutions_bool {
5819   \solutionstrue
5820 }{
5821   \solutionsfalse
5822 }
```

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

```
5823 \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.

```
5824 \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.

```
5825 \def\prob@problem@kw{Problem}
5826 \def\prob@solution@kw{Solution}
5827 \def\prob@hint@kw{Hint}
5828 \def\prob@note@kw{Note}
5829 \def\prob@gnote@kw{Grading}
5830 \def\prob@pt@kw{pt}
5831 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5832 \AddToHook{begindocument}{
5833   \ltx@ifpackageloaded{babel}{
5834     \makeatletter
5835     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5836     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5837       \input{problem-ngerman.ldf}
5838     }
5839     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5840       \input{problem-finnish.ldf}
5841     }
5842     \clist_if_in:NnT \l_tmpa_clist {french}{
5843       \input{problem-french.ldf}
5844     }
5845     \clist_if_in:NnT \l_tmpa_clist {russian}{
5846       \input{problem-russian.ldf}
5847     }
5848     \makeatother
5849   }{ }
5850 }
```

40.2 Problems and Solutions

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

```
5851 \keys_define:nn{ problem / problem }{
5852   id      .str_set:x:N = \l__problems_prob_id_str,
5853   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5854   min     .tl_set:N    = \l__problems_prob_min_tl,
5855   title   .tl_set:N    = \l__problems_prob_title_tl,
5856   type    .tl_set:N    = \l__problems_prob_type_tl,
5857   refnum  .int_set:N    = \l__problems_prob_refnum_int
5858 }
5859 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

5860 \str_clear:N \l__problems_prob_id_str
5861 \tl_clear:N \l__problems_prob_pts_tl
5862 \tl_clear:N \l__problems_prob_min_tl
5863 \tl_clear:N \l__problems_prob_title_tl
5864 \tl_clear:N \l__problems_prob_type_tl
5865 \int_zero_new:N \l__problems_prob_refnum_int
5866 \keys_set:nn { problem / problem }{ #1 }
5867 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5868   \let\l__problems_prob_refnum_int\undefined
5869 }
5870 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5871 \newcounter{problem}
5872 \newcommand\numberproblemsin[1]{\@addtoreset{problem}{#1}}

```

(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.

```

5873 \newcommand\prob@label[1]{#1}

```

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

`\prob@number` We consolidate the problem number into a reusable internal macro

```

5874 \newcommand\prob@number{
5875   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5876     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5877   }{
5878     \int_if_exist:NTF \l__problems_prob_refnum_int {
5879       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5880     }{
5881       \prob@label\theproblem
5882     }
5883   }
5884 }

```

(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.

```

5885 \newcommand\prob@title[3]{%
5886   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5887     #2 \l__problems_inclprob_title_tl #3
5888   }{
5889     \tl_if_exist:NTF \l__problems_prob_title_tl {
5890       #2 \l__problems_prob_title_tl #3
5891     }{
5892       #1
5893     }
5894   }
5895 }

```

(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.

```
5896 \def\prob@heading{
5897   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
5898   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
5899 }
```

(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`

```
5900 \newenvironment{sproblem}[1][]{
5901   \__problems_prob_args:n{#1}%\sref@target%
5902   \@in@omtexttrue% we are in a statement (for inline definitions)
5903   \stepcounter{problem}\record@problem
5904   \def\current@section@level{\prob@problem@kw}
5905   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
5906     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
5907   }{
5908     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
5909   }
5910   \str_if_exist:NTF \l__problems_inclprob_id_str {
5911     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
5912   }{
5913     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
5914   }
5915
5916
5917   \clist_set:No \l_tmpa_clist \sproblemtype
5918   \tl_clear:N \l_tmpa_tl
5919   \clist_map_inline:Nn \l_tmpa_clist {
5920     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
5921       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
5922     }
5923   }
5924   \tl_if_empty:NTF \l_tmpa_tl {
5925     \__problems_sproblem_start:
5926   }{
5927     \l_tmpa_tl
5928   }
5929   \stex_ref_new_doc_target:n \sproblemid
5930 }{
5931   \clist_set:No \l_tmpa_clist \sproblemtype
5932   \tl_clear:N \l_tmpa_tl
5933   \clist_map_inline:Nn \l_tmpa_clist {
5934     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
5935       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
5936     }
5937   }
```

```

5937 }
5938 \tl_if_empty:NTF \l_tmpa_tl {
5939   \__problems_sproblem_end:
5940 }{
5941   \l_tmpa_tl
5942 }
5943
5944
5945 \smallskip
5946 }
5947
5948
5949 \cs_new_protected:Nn \__problems_sproblem_start: {
5950   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5951 }
5952 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
5953
5954 \newcommand\stexpatchproblem[3]{} {
5955   \str_set:Nx \l_tmpa_str{ #1 }
5956   \str_if_empty:NTF \l_tmpa_str {
5957     \tl_set:Nn \__problems_sproblem_start: { #2 }
5958     \tl_set:Nn \__problems_sproblem_end: { #3 }
5959   }{
5960     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
5961     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
5962   }
5963 }
5964
5965
5966 \bool_if:NT \c__problems_boxed_bool {
5967   \surroundwithmdframed{problem}
5968 }

```

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

```

5969 \def\record@problem{
5970   \protected@write\@auxout{}
5971   {
5972     \string\@problem{\prob@number}
5973     {
5974       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5975         \l__problems_inclprob_pts_tl
5976       }{
5977         \l__problems_prob_pts_tl
5978       }
5979     }%
5980     {
5981       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5982         \l__problems_inclprob_min_tl
5983       }{
5984         \l__problems_prob_min_tl
5985       }
5986     }
5987   }
5988 }

```

(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).

```
5989 \def\@problem#1#2#3{}
```

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

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

```
5990 \keys_define:nn { problem / solution }{
5991   id                .str_set_x:N = \l__problems_solution_id_str ,
5992   for               .tl_set:N   = \l__problems_solution_for_tl ,
5993   height           .dim_set:N   = \l__problems_solution_height_dim ,
5994   creators         .clist_set:N = \l__problems_solution_creators_clist ,
5995   contributors     .clist_set:N = \l__problems_solution_contributors_clist ,
5996   srccite          .tl_set:N    = \l__problems_solution_srccite_tl
5997 }
5998 \cs_new_protected:Nn \__problems_solution_args:n {
5999   \str_clear:N \l__problems_solution_id_str
6000   \tl_clear:N \l__problems_solution_for_tl
6001   \tl_clear:N \l__problems_solution_srccite_tl
6002   \clist_clear:N \l__problems_solution_creators_clist
6003   \clist_clear:N \l__problems_solution_contributors_clist
6004   \dim_zero:N \l__problems_solution_height_dim
6005   \keys_set:nn { problem / solution }{ #1 }
6006 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
6007 \newcommand\@startsolution[1][ ]{
6008   \__problems_solution_args:n { #1 }
6009   \@in@omtexttrue% we are in a statement.
6010   \bool_if:NF \c__problems_boxed_bool { \hrule }
6011   \smallskip\noindent
6012   {\textbf\prob@solution@kw : \enspace}
6013   \begin{small}
6014   \def\current@section@level{\prob@solution@kw}
6015   \ignorespacesandpars
6016 }
```

\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.

```
6017 \newcommand\startsolutions{
6018   \specialcomment{solution}{\@startsolution}{
6019     \bool_if:NF \c__problems_boxed_bool {
6020       \hrule\medskip
6021     }
6022     \end{small}%
6023   }
6024   \bool_if:NT \c__problems_boxed_bool {
6025     \surroundwithmdframed{solution}
6026   }
6027 }
```

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

\stopsolutions

```
6028 \newcommand\stopsolutions{\excludecomment{solution}}
```

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

so it only remains to start/stop solutions depending on what option was specified.

```
6029 \ifsolutions
```

```
6030 \startsolutions
```

```
6031 \else
```

```
6032 \stopsolutions
```

```
6033 \fi
```

exnote

```
6034 \bool_if:NTF \c__problems_notes_bool {
```

```
6035 \newenvironment{exnote}[1][]{
```

```
6036 \par\smallskip\hrule\smallskip
```

```
6037 \noindent\textbf{\prob@note@kw : }\small
```

```
6038 }{
```

```
6039 \smallskip\hrule
```

```
6040 }
```

```
6041 }{
```

```
6042 \excludecomment{exnote}
```

```
6043 }
```

hint

```
6044 \bool_if:NTF \c__problems_notes_bool {
```

```
6045 \newenvironment{hint}[1][]{
```

```
6046 \par\smallskip\hrule\smallskip
```

```
6047 \noindent\textbf{\prob@hint@kw :~ }\small
```

```
6048 }{
```

```
6049 \smallskip\hrule
```

```
6050 }
```

```
6051 \newenvironment{exhint}[1][]{
```

```
6052 \par\smallskip\hrule\smallskip
```

```
6053 \noindent\textbf{\prob@hint@kw :~ }\small
```

```
6054 }{
```

```
6055 \smallskip\hrule
```

```
6056 }
```

```
6057 }{
```

```
6058 \excludecomment{hint}
```

```
6059 \excludecomment{exhint}
```

```
6060 }
```

gnote

```
6061 \bool_if:NTF \c__problems_notes_bool {
```

```
6062 \newenvironment{gnote}[1][]{
```

```
6063 \par\smallskip\hrule\smallskip
```

```
6064 \noindent\textbf{\prob@gnote@kw : }\small
```

```
6065 }{
```

```
6066 \smallskip\hrule
```

```
6067 }
```

```
6068 }{
```

```
6069 \excludecomment{gnote}
```

```
6070 }
```


40.3 Multiple Choice Blocks

EdN:23

mcb 23

```

6071 \newenvironment{mcb}{
6072   \begin{enumerate}
6073 }{
6074   \end{enumerate}
6075 }

```

we define the keys for the mcc macro

```

6076 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6077   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6078     \bool_set_true:N #1
6079   }{
6080     \bool_set_false:N #1
6081   }
6082 }
6083 \keys_define:nn { problem / mcc }{
6084   id          .str_set_x:N = \l__problems_mcc_id_str ,
6085   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6086   T           .default:n    = { true } ,
6087   T           .bool_set:N   = \l__problems_mcc_t_bool ,
6088   F           .default:n    = { true } ,
6089   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6090   Ttext       .code:n       = {
6091     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6092   } ,
6093   Ftext       .code:n       = {
6094     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6095   }
6096 }
6097 \cs_new_protected:Nn \l__problems_mcc_args:n {
6098   \str_clear:N \l__problems_mcc_id_str
6099   \tl_clear:N \l__problems_mcc_feedback_tl
6100   \bool_set_true:N \l__problems_mcc_t_bool
6101   \bool_set_true:N \l__problems_mcc_f_bool
6102   \bool_set_true:N \l__problems_mcc_Ttext_bool
6103   \bool_set_false:N \l__problems_mcc_Ftext_bool
6104   \keys_set:nn { problem / mcc }{ #1 }
6105 }

```

\mcc

```

6106 \newcommand\mcc[2][] {
6107   \l__problems_mcc_args:n{ #1 }
6108   \item #2
6109   \ifsolutions
6110     \\\
6111     \bool_if:NT \l__problems_mcc_t_bool {
6112       % TODO!
6113       % \ifcsstring{mcc@T}{T}{ }\{mcc@Ttext}%
6114     }
6115     \bool_if:NT \l__problems_mcc_f_bool {

```

²³EdNOTE: MK: maybe import something better here from a dedicated MC package

```

6116         % TODO!
6117         % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6118     }
6119     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6120         !
6121     }{
6122         \l__problems_mcc_feedback_tl
6123     }
6124     \fi
6125 } %solutions

```

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

40.4 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.

```

6126
6127 \keys_define:nn{ problem / inclproblem }{
6128     id      .str_set:N = \l__problems_inclprob_id_str,
6129     pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6130     min     .tl_set:N  = \l__problems_inclprob_min_tl,
6131     title   .tl_set:N  = \l__problems_inclprob_title_tl,
6132     refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6133     type    .tl_set:N  = \l__problems_inclprob_type_tl,
6134     mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
6135 }
6136 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6137     \str_clear:N \l__problems_prob_id_str
6138     \tl_clear:N \l__problems_inclprob_pts_tl
6139     \tl_clear:N \l__problems_inclprob_min_tl
6140     \tl_clear:N \l__problems_inclprob_title_tl
6141     \tl_clear:N \l__problems_inclprob_type_tl
6142     \int_zero_new:N \l__problems_inclprob_refnum_int
6143     \str_clear:N \l__problems_inclprob_mhrepos_str
6144     \keys_set:nn { problem / inclproblem }{ #1 }
6145     \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6146         \let\l__problems_inclprob_pts_tl\undefined
6147     }
6148     \tl_if_empty:NT \l__problems_inclprob_min_tl {
6149         \let\l__problems_inclprob_min_tl\undefined
6150     }
6151     \tl_if_empty:NT \l__problems_inclprob_title_tl {
6152         \let\l__problems_inclprob_title_tl\undefined
6153     }
6154     \tl_if_empty:NT \l__problems_inclprob_type_tl {
6155         \let\l__problems_inclprob_type_tl\undefined
6156     }
6157     \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6158         \let\l__problems_inclprob_refnum_int\undefined
6159     }
6160 }

```

```

6161
6162 \cs_new_protected:Nn \__problems_inclprob_clear: {
6163   \let\l__problems_inclprob_id_str\undefined
6164   \let\l__problems_inclprob_pts_tl\undefined
6165   \let\l__problems_inclprob_min_tl\undefined
6166   \let\l__problems_inclprob_title_tl\undefined
6167   \let\l__problems_inclprob_type_tl\undefined
6168   \let\l__problems_inclprob_refnum_int\undefined
6169   \let\l__problems_inclprob_mhrepos_str\undefined
6170 }
6171 \__problems_inclprob_clear:
6172
6173 \newcommand\includeproblem[2][ ]{
6174   \__problems_inclprob_args:n{ #1 }
6175   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6176     \input{#2}
6177   }{
6178     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6179       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6180     }
6181   }
6182   \__problems_inclprob_clear:
6183 }

```

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

40.5 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.

```

6184 \AddToHook{enddocument}{
6185   \bool_if:NT \c__problems_pts_bool {
6186     \message{Total:~\arabic{pts}~points}
6187   }
6188   \bool_if:NT \c__problems_min_bool {
6189     \message{Total:~\arabic{min}~minutes}
6190   }
6191 }

```

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

```

6192 \def\pts#1{
6193   \bool_if:NT \c__problems_pts_bool {
6194     \marginpar{#1~\prob@pt@kw}
6195   }
6196 }
6197 \def\min#1{
6198   \bool_if:NT \c__problems_min_bool {
6199     \marginpar{#1~\prob@min@kw}
6200   }
6201 }

```

`\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.

```

6202 \newcounter{pts}
6203 \def\show@pts{
6204   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6205     \bool_if:NT \c__problems_pts_bool {
6206       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6207       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6208     }
6209   }{
6210     \tl_if_exist:NT \l__problems_prob_pts_tl {
6211       \bool_if:NT \c__problems_pts_bool {
6212         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6213         \addtocounter{pts}{\l__problems_prob_pts_tl}
6214       }
6215     }
6216   }
6217 }

```

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

`\show@min`

```

6218 \newcounter{min}
6219 \def\show@min{
6220   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6221     \bool_if:NT \c__problems_min_bool {
6222       \marginpar{\l__problems_inclprob_min_tl\ min}
6223       \addtocounter{min}{\l__problems_inclprob_min_tl}
6224     }
6225   }{
6226     \tl_if_exist:NT \l__problems_prob_min_tl {
6227       \bool_if:NT \c__problems_min_bool {
6228         \marginpar{\l__problems_prob_min_tl\ min}
6229         \addtocounter{min}{\l__problems_prob_min_tl}
6230       }
6231     }
6232   }
6233 }
6234 \</package>

```

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

Chapter 41

Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

41.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
6235 \@@=hwexam>
6236 \*cls>
6237 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6238 \RequirePackage{l3keys2e,expl-keystr-compatible}
6239 \DeclareOption*{
6240   \PassOptionsToClass{\CurrentOption}{document-structure}
6241   \PassOptionsToPackage{\CurrentOption}{stex}
6242   \PassOptionsToPackage{\CurrentOption}{hwexam}
6243   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6244 }
6245 \ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the L^AT_EXML bindings, we make sure the right packages are loaded.

```
6246 \LoadClass{document-structure}
6247 \RequirePackage{stex}
6248 \RequirePackage{hwexam}
6249 \RequirePackage{tikzinput}
6250 \RequirePackage{graphicx}
6251 \RequirePackage{a4wide}
6252 \RequirePackage{amssymb}
6253 \RequirePackage{amstext}
6254 \RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors

```

6255 \newcommand\assig@default@type{\hwexam@assignment@kw}
6256 \def\document@hwexamtype{\assig@default@type}
6257 <@@=document_structure>
6258 \keys_define:nn { document-structure / document }{
6259 id .str_set_x:N = \c_document_structure_document_id_str,
6260 hwexamtype .tl_set:N = \document@hwexamtype
6261 }
6262 <@@=hwexam>
6263 </cls>

```

Chapter 42

Implementation: The hwexam Package

42.1 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.

```
6264 \*package>
6265 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6266 \RequirePackage{l3keys2e,expl-keystr-compat}
6267
6268 \newif\iftest\testfalse
6269 \DeclareOption{test}{\testtrue}
6270 \newif\ifmultiple\multiplefalse
6271 \DeclareOption{multiple}{\multipletrue}
6272 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6273 \ProcessOptions
```

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

```
6274 \RequirePackage{keyval}[1997/11/10]
6275 \RequirePackage{problem}
```

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

```
6276 \newcommand\hwexam@assignment@kw{Assignment}
6277 \newcommand\hwexam@given@kw{Given}
6278 \newcommand\hwexam@due@kw{Due}
6279 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6280 blank~for~extra~space}
6281 \def\hwexam@minutes@kw{minutes}
6282 \newcommand\correction@probs@kw{prob.}
6283 \newcommand\correction@pts@kw{total}
6284 \newcommand\correction@reached@kw{reached}
6285 \newcommand\correction@sum@kw{Sum}
6286 \newcommand\correction@grade@kw{grade}
6287 \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

```

6288 \AddToHook{begindocument}{
6289 \ltx@ifpackageloaded{babel}{
6290 \makeatletter
6291 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6292 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6293 \input{hwexam-ngerman.ldf}
6294 }
6295 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6296 \input{hwexam-finnish.ldf}
6297 }
6298 \clist_if_in:NnT \l_tmpa_clist {french}{
6299 \input{hwexam-french.ldf}
6300 }
6301 \clist_if_in:NnT \l_tmpa_clist {russian}{
6302 \input{hwexam-russian.ldf}
6303 }
6304 \makeatother
6305 }{}
6306 }
6307

```

42.2 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.

```

6308 \newcounter{assignment}
6309 \numberproblemsin{assignment}
6310 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6311 \keys_define:nn { hwexam / assignment } {
6312 id .str_set:N = \l__hwexam_assign_id_str,
6313 number .int_set:N = \l__hwexam_assign_number_int,
6314 title .tl_set:N = \l__hwexam_assign_title_tl,
6315 type .tl_set:N = \l__hwexam_assign_type_tl,
6316 given .tl_set:N = \l__hwexam_assign_given_tl,
6317 due .tl_set:N = \l__hwexam_assign_due_tl,
6318 loadmodules .code:n = {
6319 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6320 }
6321 }
6322 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6323 \str_clear:N \l__hwexam_assign_id_str
6324 \int_set:Nn \l__hwexam_assign_number_int {-1}
6325 \tl_clear:N \l__hwexam_assign_title_tl
6326 \tl_clear:N \l__hwexam_assign_type_tl
6327 \tl_clear:N \l__hwexam_assign_given_tl
6328 \tl_clear:N \l__hwexam_assign_due_tl
6329 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```



```

6330 \keys_set:nn { hwexam / assignment }{ #1 }
6331 }

```

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.

```

6332 \newcommand\given@due[2]{
6333 \bool_lazy_all:nF {
6334 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6335 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6336 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6337 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6338 }{ #1 }
6339
6340 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6341 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6342 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6343 }
6344 }{
6345 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6346 }
6347
6348 \bool_lazy_or:nnF {
6349 \bool_lazy_and_p:nn {
6350 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6351 }{
6352 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6353 }
6354 }{
6355 \bool_lazy_and_p:nn {
6356 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6357 }{
6358 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6359 }
6360 }{ ,~ }
6361
6362 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6363 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6364 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6365 }
6366 }{
6367 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6368 }
6369
6370 \bool_lazy_all:nF {
6371 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6372 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6373 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6374 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6375 }{ #2 }
6376 }

```

`\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.

```

6377 \newcommand\assignment@title[3]{
6378 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6379 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6380 #1
6381 }{
6382 #2\l__hwexam_assign_title_tl#3
6383 }
6384 }{
6385 #2\l__hwexam_inclasssign_title_tl#3
6386 }
6387 }

```

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

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

```

6388 \newcommand\assignment@number{
6389 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6390 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6391 \arabic{assignment}
6392 } {
6393 \int_use:N \l__hwexam_assign_number_int
6394 }
6395 }{
6396 \int_use:N \l__hwexam_inclasssign_number_int
6397 }
6398 }

```

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

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

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

```

6399 \newenvironment{assignment}[1][ ]{
6400 \__hwexam_assignment_args:n { #1 }
6401 %\sref@target
6402 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6403 \global\stepcounter{assignment}
6404 }{
6405 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6406 }
6407 \setcounter{problem}{0}
6408 \def\current@section@level{\document@hwexamtype}
6409 %\sref@label@id{\document@hwexamtype \thesection}
6410 \begin{@assignment}
6411 }{
6412 \end{@assignment}
6413 }

```

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

```

6414 \def\ass@title{
6415 \protect\document@hwexamtype~\arabic{assignment}
6416 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
6417 }
6418 \ifmultiple
6419 \newenvironment{@assignment}{
6420 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6421 \begin{omgroup}[loadmodules]{\ass@title}
6422 }{
6423 \begin{omgroup}{\ass@title}
6424 }
6425 }{
6426 \end{omgroup}
6427 }

```

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

```

6428 \else
6429 \newenvironment{@assignment}{
6430 \begin{center}\bf
6431 \Large@title\strut\
6432 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}\{;\}\{;\}
6433 \large\given@due{--;\}\{;\}--}
6434 \end{center}
6435 }{}
6436 \fi% multiple

```

42.3 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 `inclasssign` keys after the input.

```

6437 \keys_define:nn { hwexam / inclasssignment } {
6438 %id .str_set_x:N = \l__hwexam_assign_id_str,
6439 number .int_set:N = \l__hwexam_inclasssign_number_int,
6440 title .tl_set:N = \l__hwexam_inclasssign_title_tl,
6441 type .tl_set:N = \l__hwexam_inclasssign_type_tl,
6442 given .tl_set:N = \l__hwexam_inclasssign_given_tl,
6443 due .tl_set:N = \l__hwexam_inclasssign_due_tl,
6444 mhrepos .str_set_x:N = \l__hwexam_inclasssign_mhrepos_str
6445 }
6446 \cs_new_protected:Nn \__hwexam_inclasssignment_args:n {
6447 \int_set:Nn \l__hwexam_inclasssign_number_int {-1}
6448 \tl_clear:N \l__hwexam_inclasssign_title_tl
6449 \tl_clear:N \l__hwexam_inclasssign_type_tl
6450 \tl_clear:N \l__hwexam_inclasssign_given_tl
6451 \tl_clear:N \l__hwexam_inclasssign_due_tl
6452 \str_clear:N \l__hwexam_inclasssign_mhrepos_str
6453 \keys_set:nn { hwexam / inclasssignment }{ #1 }
6454 }
6455 \__hwexam_inclasssignment_args:n {}
6456
6457 \newcommand\inputassignment[2][{}]{

```

```

6458 \_hwexam_inclassnment_args:n { #1 }
6459 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6460 \input{#2}
6461 }{
6462 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6463 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6464 }
6465 }
6466 \_hwexam_inclassnment_args:n {}
6467 }
6468 \newcommand\includeassignment[2][]{
6469 \newpage
6470 \inputassignment[#1]{#2}
6471 }

```

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

42.4 Typesetting Exams

\quizheading

```

6472 \ExplSyntaxOff
6473 \newcommand\quizheading[1]{%
6474 \def\@tas{#1}%
6475 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6476 \ifx\@tas\@empty\else%
6477 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6478 \fi%
6479 }
6480 \ExplSyntaxOn

```

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

\testheading

```

6481
6482 \def\hwexamheader{\input{hwexam-default.header}}
6483
6484 \def\hwexamminutes{
6485 \tl_if_empty:NTF \testheading@duration {
6486 {\testheading@min}~\hwexam@minutes@kw
6487 }{
6488 \testheading@duration
6489 }
6490 }
6491
6492 \keys_define:nn { hwexam / testheading } {
6493 min .tl_set:N = \testheading@min,
6494 duration .tl_set:N = \testheading@duration,
6495 reqpts .tl_set:N = \testheading@reqpts,
6496 tools .tl_set:N = \testheading@tools
6497 }
6498 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6499 \tl_clear:N \testheading@min
6500 \tl_clear:N \testheading@duration

```

```

6501 \tl_clear:N \testheading@reqpts
6502 \tl_clear:N \testheading@tools
6503 \keys_set:nn { hwexam / testheading }{ #1 }
6504 }
6505 \newenvironment{testheading}[1][]{
6506   \_hwexam_testheading_args:n{ #1 }
6507   \newcount\check@time\check@time=\testheading@min
6508   \advance\check@time by -\theassignment@totalmin
6509   \newif\if@bonuspoints
6510   \tl_if_empty:NTF \testheading@reqpts {
6511     \@bonuspointsfalse
6512   }{
6513     \newcount\bonus@pts
6514     \bonus@pts=\theassignment@totalpts
6515     \advance\bonus@pts by -\testheading@reqpts
6516     \edef\bonus@pts{\the\bonus@pts}
6517     \@bonuspointstrue
6518   }
6519   \edef\check@time{\the\check@time}
6520
6521   \makeatletter\hwexamheader\makeatother
6522 }{
6523   \newpage
6524 }

```

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

\testspace

```

6525 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6526 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6527 \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.

```

6528 <@=problems>
6529 \renewcommand\@problem[3]{
6530   \stepcounter{assignment@probs}
6531   \def\__problemspts{#2}
6532   \ifx\__problemspts\@empty\else
6533     \addtocounter{assignment@totalpts}{#2}
6534   \fi
6535   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6536   \xdef\correction@probs{\correction@probs & #1}%
6537   \xdef\correction@pts{\correction@pts & #2}
6538   \xdef\correction@reached{\correction@reached &}

```

```

6539 }
6540 <@@=hwexam>

```

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

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

```

6541 \newcounter{assignment@probs}
6542 \newcounter{assignment@totalpts}
6543 \newcounter{assignment@totalmin}
6544 \def\correction@probs{\correction@probs@kw}
6545 \def\correction@pts{\correction@pts@kw}
6546 \def\correction@reached{\correction@reached@kw}
6547 \stepcounter{assignment@probs}
6548 \newcommand\correction@table{
6549 \resizebox{\textwidth}{!}{%
6550 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6551 &\multicolumn{\theassignment@probs}{c|}|%|
6552 {\footnotesize\correction@forgrading@kw} &\\ \hline
6553 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6554 \correction@pts & \theassignment@totalpts & \\ \hline
6555 \correction@reached & & \[.7cm]\hline
6556 \end{tabular}}
6557 \end{package}

```

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

42.5 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}

```