

The sTeX3 Package *

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

2022-02-19

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 2022-02-19)

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 sTeX	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
9.1.1	HTML Annotations	20
9.1.2	Babel Languages	21
9.1.3	Auxiliary Methods	21

10	<code>sTeX</code>-MathHub	22
10.1	Macros and Environments	22
10.1.1	Files, Paths, URIs	22
10.1.2	MathHub Archives	23
10.1.3	Using Content in Archives	24
11	<code>sTeX</code>-References	25
11.1	Macros and Environments	25
11.1.1	Setting Reference Targets	25
11.1.2	Using References	26
12	<code>sTeX</code>-Modules	27
12.1	Macros and Environments	27
12.1.1	The <code>smodule</code> environment	29
13	<code>sTeX</code>-Module Inheritance	31
13.1	Macros and Environments	31
13.1.1	SMS Mode	31
13.1.2	Imports and Inheritance	32
14	<code>sTeX</code>-Symbols	35
14.1	Macros and Environments	35
15	<code>sTeX</code>-Terms	38
15.1	Macros and Environments	38
16	<code>sTeX</code>-Structural Features	41
16.1	Macros and Environments	41
16.1.1	Structures	41
17	<code>sTeX</code>-Statements	42
17.1	Macros and Environments	42
18	<code>sTeX</code>-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	<code>sTeX</code>-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 L^AT_EX	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	HTML Annotations	74
25.5	Babel Languages	77
25.6	Auxiliary Methods	78
26	STeX-MathHub Implementation	79
26.1	Generic Path Handling	79
26.2	PWD and kpsewhich	81
26.3	File Hooks and Tracking	82
26.4	MathHub Repositories	83
26.5	Using Content in Archives	87
27	STeX-References Implementation	92
27.1	Document URIs and URLs	92
27.2	Setting Reference Targets	94
27.3	Using References	96
28	STeX-Modules Implementation	99
28.1	The smodule environment	103
28.2	Invoking modules	108
29	STeX-Module Inheritance Implementation	110
29.1	SMS Mode	110
29.2	Inheritance	113
30	STeX-Symbols Implementation	117
30.1	Symbol Declarations	117
30.2	Notations	124
30.3	Variables	134
31	STeX-Terms Implementation	137
31.1	Symbol Invocations	137
31.2	Terms	140
31.3	Notation Components	147
32	STeX-Structural Features Implementation	150
32.1	Imports with modification	150
32.2	The feature environment	157
32.3	Features	158
33	STeX-Statements Implementation	164
33.1	Definitions	164
33.2	Assertions	169
33.3	Examples	172
33.4	Logical Paragraphs	174

34 The Implementation	180
34.1 Package Options	180
34.2 Proofs	180
34.3 Justifications	186
35 \LaTeX-Others Implementation	188
36 \LaTeX-Metatheory Implementation	189
37 Tikzinput Implementation	192
38 document-structure.sty Implementation	194
38.1 The document-structure Class	194
38.2 Class Options	194
38.3 Beefing up the <code>document</code> environment	195
38.4 Implementation: document-structure Package	195
38.5 Package Options	195
38.6 Document Structure	197
38.7 Front and Backmatter	200
38.8 Global Variables	202
39 NotesSlides – Implementation	203
39.1 Class and Package Options	203
39.2 Notes and Slides	205
39.3 Header and Footer Lines	209
39.4 Frame Images	210
39.5 Colors and Highlighting	211
39.6 Sectioning	212
39.7 Excursions	214
40 The Implementation	216
40.1 Package Options	216
40.2 Problems and Solutions	217
40.3 Multiple Choice Blocks	223
40.4 Including Problems	224
40.5 Reporting Metadata	225
41 Implementation: The hwexam Class	227
41.1 Class Options	227
42 Implementation: The hwexam Package	229
42.1 Package Options	229
42.2 Assignments	230
42.3 Including Assignments	233
42.4 Typesetting Exams	234
42.5 Leftovers	236

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,
- RuS_{TeX} 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 \LaTeX

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

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

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

sms ($\langle\textit{boolean}\rangle$) use *persisted* mode (not yet implemented).

image ($\langle\textit{boolean}\rangle$) passed on to `tikzinput`.

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

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

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

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}{\comp1}
Test: $\rtimes a{\rplus c}{\rtimes de}$
\end{smodule}
```

Module 5: Test: $a \cdot (c + d \cdot e)$

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 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` Both print this \TeX logo.
`\stex`

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}][ $\$a$ ][\comp{and}][ $\$b$ ]
```

$a * b$ 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 $\mathbin{+}$ adds two elements, as in $a\mathbin{+}b$.

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

Example 11

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

$\textcolor{teal}{Multiplication}$ (denoted by $\mathbin{\cdot}$) 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.

\TeX 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}
 $\plus{a}{\times{b}{c}}$  and  $\times{a}{\plus{b}{c}}$ 
```

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

8.1.2 Archives and Imports

Namespaces

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

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

- If `\begin{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

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

9.1 Macros and Environments

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

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

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

9.1.1 HTML Annotations

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

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

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

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

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

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

Annotates the HTML generated by `⟨content⟩` 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> <code>⟨content⟩</code> <code>\end{stex_annotate_env}</code> behaves like <code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code> .
--------------------------------	--

9.1.2 Babel Languages

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

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

9.1.3 Auxiliary Methods

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

Makes the macro `⟨cs⟩` throw an error, indicating that it is only allowed in the context of `⟨environments⟩`.

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

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

Chapter 10

STEX-MathHub

This sub package provides code for handling ST_EX archives, files, file paths and related 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> $\langle path-variable \rangle$ $\{\langle string \rangle\}$ turns the $\langle string \rangle$ into a path by splitting it at <code>/</code> -characters and stores the result in $\langle path-variable \rangle$. Also applies <code>\stex_path_canonicalize:N</code> .
--	--

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

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

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

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

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

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

10.1.2 MathHub Archives

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

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

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

<code>\l_stex_current_repository_prop</code>
--

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

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

<code>\stex_set_current_repository:n</code>

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

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

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

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

10.1.3 Using Content in Archives

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

Chapter 11

STEX-References

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

11.1 Macros and Environments

\STEXreftitle

\STEXreftitle{<some title>}

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

\stex_get_document_uri:

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

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

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

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

11.1.1 Setting Reference Targets

\stex_ref_new_doc_target:n

\stex_ref_new_doc_target:n{<id>}

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

\stex_ref_new_sym_target:n

\stex_ref_new_sym_target:n{<uri>}

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

11.1.2 Using References

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

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

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

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

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

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

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

Chapter 12

STEX-Modules

This sub package contains code related to Modules

12.1 Macros and Environments

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

`\c_stex_module_<URI>_prop`

A property list with the following fields:

name The *name* of the module,

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

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

lang the module's *language*,

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

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

meta the metatheory of the module.

`\c_stex_module_<URI>_code`

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

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

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

12.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\stex_invoke_module:n

Invoked by `\STEXModule`. Needs to be followed either by `!\macro` or `?{\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.

\stex_activate_module:n

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

Chapter 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 1

```
\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 2

```
\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 10:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
                Meaning: >macro:->\protect \bar <
Module 11:      Meaning: >macro:->\stex_invoke_symbol:n {file://stextest?Foo?foo}<
Module 12:      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 3

```

\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 13:

Module 14: Meaning: »macro:->\stex_invoke_symbol:n {file://stextest?UseTest1?foo}«

Module 15: 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](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?collection>, <http://mathhub.info/sTeX?Metatheory?proposition>, <http://mathhub.info/sTeX?Metatheory?seqtype>, <http://mathhub.info/sTeX?Metatheory?seqdots>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseto>, <http://mathhub.info/sTeX?Metatheory?module-type>, <http://mathhub.info/sTeX?Metatheory?mat>, <http://mathhub.info/sTeX?Metatheory?isat>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?dur>, <http://mathhub.info/sTeX?Metatheory?proposition>, <http://mathhub.info/sTeX?Metatheory?seqtype>, <http://mathhub.info/sTeX?Metatheory?collection>, <http://mathhub.info/sTeX?Metatheory?aseqdots>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?let>, <http://mathhub.info/sTeX?Metatheory?module-type>, <http://mathhub.info/sTeX?Metatheory?mat>, <http://mathhub.info/sTeX?Metatheory?structure>, file://stextest?UseTest2?bar

Test 4

```

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

```

Circular dependencies:

Module 16: »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 source folder of the archive, containing a module $\langle name \rangle$. That module should lie directly in the namespace of the archive. (b) If $\langle path \rangle$ is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive. If a module by that namespace exists, it is returned. Otherwise, we call <code>\stex_require_module:nn</code> on the source 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

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

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

- **name**: An (OMDOC) name. By default equal to $\langle\text{macroname}\rangle$.
- **type**: An (ideally semantic) term. 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. $\backslash\text{symdecl}[\text{args=ii}]\{\text{plus}\}$ allows for $\backslash\text{plus}\{2\}\{2\}$.
 - a an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. $\backslash\text{symdecl}[\text{args=a}]\{\text{plus}\}$ allows for $\backslash\text{plus}\{2,2,2\}$.
 - 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. $\backslash\text{symdecl}[\text{args=bi}]\{\text{forall}\}$ allows for $\backslash\text{forall}\{x\in\text{Nat}\}\{x\geq 0\}$.

`\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 5

```
\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 17:      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 6

```
\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}_ {\com
```

Module 18:

`\symdef`

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

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

Test 7

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

Module 19: $a+b+c$

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 8

```
\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 20: $\langle a^b_c \rangle$ and $\langle a^b_c \rangle$.

Test 9

```
\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 $\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$
$[\plus{a,\mult{b,c}}]\text{ and }[\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}]\$
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[{$\displaystyle
\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$}
\end{smodule}
```

Module 21: $\langle a|[b;c;d;e;f]^g \rangle$ and $\langle a|[b;c]^g \rangle$ and $\langle a|[b]^c \rangle$

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

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

$a+(b \cdot c)$ 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 10

```
\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 22:  some aand some band also some chere.
           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 spfcase environments that mark up the cases one by one.
spfcase	The content of a pfcases environment are a sequence of case proofs marked up in the pfcase 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 pfcase environment is the same as that of a proof , i.e. steps , proofcomments , and pfcases environments. \spfcasesketch is a variant of the spfcase 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 to, or what we have achieved so far. As such, it cannot be the target of a \premise .

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 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 an infrastructure for writing OMDOC documents in L^AT_EX. 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 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). 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
<code>id</code>	metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The op-
<code>creators</code>	tional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code>
<code>contributors</code>	for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The
<code>short</code>	<code>short</code> allows to give a short title for the generated section. If the title contains semantic
<code>loadmodules</code>	macros, they need to be protected by <code>\protect</code> , and we need to give the <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 \LaTeX 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, \LaTeX 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 \LaTeX ML generate the correct reference.

`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in \LaTeX . 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 \LaTeX 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 \LaTeX 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 `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.

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

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

ST_EX -Basics Implementation

25.1 The ST_EXDocument 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 sTeX logo:
\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 HTML Annotations

```

77 <@=stex_annotate>
78 \RequirePackage{rustex}

```

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

```

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

```

Conditionals for `LATeXML`:

`\if@latexml`

```

80 \ifcsname if@latexml\endcsname\else
81   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
82 \fi

```

(End definition for `\if@latexml`. This function is documented on page 20.)

`\latexml_if_p:`

`\latexml_if:TF`

```

83 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
84   \if@latexml
85     \prg_return_true:
86   \else:
87     \prg_return_false:
88   \fi:
89 }

```

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

`\l__stex_annotate_arg_tl`

`\c__stex_annotate_emptyarg_tl`

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

```

90 \tl_new:N \l__stex_annotate_arg_tl
91 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
92   \rustex_if:TF {
93     \rustex_direct_HTML:n { \c_ampsand_str lrm; }
94   }{-}
95 }

```

(End definition for `\l__stex_annotate_arg_tl` and `\c__stex_annotate_emptyarg_tl`.)

`_stex_annotate_checkempty:n`

```

96 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
97   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
98   \tl_if_empty:NT \l__stex_annotate_arg_tl {
99     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
100   }
101 }

```

(End definition for `_stex_annotate_checkempty:n`.)

`\stex_if_do_html_p:`
`\stex_if_do_html:TF`

Whether to (locally) produce HTML output

```

102 \bool_new:N \_stex_html_do_output_bool
103 \bool_set_true:N \_stex_html_do_output_bool
104
105 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
106   \bool_if:nTF \_stex_html_do_output_bool
107     \prg_return_true: \prg_return_false:
108 }

```

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

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```

109 \cs_new_protected:Nn \stex_suppress_html:n {
110   \exp_args:Nne \use:nn {
111     \bool_set_false:N \_stex_html_do_output_bool
112     #1
113   }{
114     \stex_if_do_html:T {
115       \bool_set_true:N \_stex_html_do_output_bool
116     }
117   }
118 }

```

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

`\stex_annotate:nnx`

`\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.

```

119 \rustex_if:TF{
120   \cs_new_protected:Nn \stex_annotate:nnn {
121     \_stex_annotate_checkempty:n { #3 }
122     \rustex_annotate_HTML:nn {
123       property="stex:#1" ~
124       resource="#2"
125     } {
126       \mode_if_vertical:TF{
127         \tl_use:N \l__stex_annotate_arg_tl\par
128       }{
129         \tl_use:N \l__stex_annotate_arg_tl
130       }
131     }
132   }
133   \cs_new_protected:Nn \stex_annotate_invisible:n {

```

```

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

```

```

188         \tl_use:N \l__stex_annotate_arg_tl
189     }
190 } {
191     \cs:w latexml@invisible@text\cs_end:{
192         \tl_use:N \l__stex_annotate_arg_tl
193     }
194 }
195 }
196 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
197     \__stex_annotate_checkempty:n { #3 }
198     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
199         \tl_use:N \l__stex_annotate_arg_tl
200     }
201 }
202 \NewDocumentEnvironment{stex_annotate_env} { m m } {
203     \par\begin{latexml@annotateenv}{#1}{#2}
204 }{
205     \par\end{latexml@annotateenv}
206 }
207 }{
208     \cs_new_protected:Nn \stex_annotate:nnn {#3}
209     \cs_new_protected:Nn \stex_annotate_invisible:n {}
210     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
211     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
212 }
213 }

```

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

25.5 Babel Languages

```

214 <@@=stex_language>

```

`\c_stex_languages_prop` We store language abbreviations in two (mutually inverse) property lists:
`\c_stex_language_abbrevs_prop`

```

215 \prop_const_from_keyval:Nn \c_stex_languages_prop {
216     en = english ,
217     de = ngerman ,
218     ar = arabic ,
219     bg = bulgarian ,
220     ru = russian ,
221     fi = finnish ,
222     ro = romanian ,
223     tr = turkish ,
224     fr = french
225 }
226
227 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
228     english = en ,
229     ngerman = de ,
230     arabic = ar ,
231     bulgarian = bg ,
232     russian = ru ,
233     finnish = fi ,

```

```

234   romanian = ro ,
235   turkish  = tr ,
236   french   = fr
237 }
238 % todo: chinese simplified (zhs)
239 %       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:

```

240 \clist_if_empty:NF \c_stex_languages_clist {
241   \clist_clear:N \l_tmpa_clist
242   \clist_map_inline:Nn \c_stex_languages_clist {
243     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
244       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
245     } {
246       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
247     }
248   }
249   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
250   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
251 }

```

25.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

252 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
253   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
254   \def#1{
255     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
256   }
257 }

```

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

`\stex_reactivate_macro:N`

```

258 \cs_new_protected:Nn \stex_reactivate_macro:N {
259   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
260 }

```

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

`\ignorespacesandpars`

```

261 \protected\def\ignorespacesandpars{
262   \begingroup\catcode13=10\relax
263   \@ifnextchar\par{
264     \endgroup\expandafter\ignorespacesandpars\@gobble
265   }{
266     \endgroup
267   }
268 }
269 \</package>

```

(End definition for `\ignorespacesandpars`. This function is documented on page 21.)

Chapter 26

STEX -MathHub Implementation

```
270 <*package>
271
272 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
273
274 <@@=stex_path>
275
276 Warnings and error messages
277 \msg_new:nnn{stex}{error/norepository}{
278   No~archive~#1~found~in~#2
279 }
280 \msg_new:nnn{stex}{error/notinarchive}{
281   Not~currently~in~an~archive,~but~\detokenize{#1}~
282   needs~one!
283 }
284 \msg_new:nnn{stex}{error/nofile}{
285   \detokenize{#1}~could~not~find~file~#2
286 }
287 \msg_new:nnn{stex}{error/twofiles}{
288   \detokenize{#1}~found~two~candidates~for~#2
289 }
290 }
```

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`

```
288 \cs_new_protected:Nn \stex_path_from_string:Nn {
289   \str_set:Nx \l_tmpa_str { #2 }
290   \str_if_empty:NTF \l_tmpa_str {
291     \seq_clear:N #1
292   }{
293     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
294     \sys_if_platform_windows:T{
295       \seq_clear:N \l_tmpa_tl
```

```

296     \seq_map_inline:Nn #1 {
297       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
298       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
299     }
300     \seq_set_eq:NN #1 \l_tmpa_tl
301   }
302   \stex_path_canonicalize:N #1
303 }
304 }
305

```

(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`

```

306 \cs_new_protected:Nn \stex_path_to_string:NN {
307   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
308 }
309
310 \cs_new:Nn \stex_path_to_string:N {
311   \seq_use:Nn #1 /
312 }

```

(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`

```

313 \str_const:Nn \c__stex_path_dot_str {.}
314 \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.

```

315 \cs_new_protected:Nn \stex_path_canonicalize:N {
316   \seq_if_empty:NF #1 {
317     \seq_clear:N \l_tmpa_seq
318     \seq_get_left:NN #1 \l_tmpa_tl
319     \str_if_empty:NT \l_tmpa_tl {
320       \seq_put_right:Nn \l_tmpa_seq {}
321     }
322     \seq_map_inline:Nn #1 {
323       \str_set:Nn \l_tmpa_tl { ##1 }
324       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
325         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
326           \seq_if_empty:NNTF \l_tmpa_seq {
327             \exp_args:Nno \seq_put_right:Nn \l_tmpa_seq {
328               \c__stex_path_up_str
329             }
330           }{
331             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
332             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
333               \exp_args:Nno \seq_put_right:Nn \l_tmpa_seq {
334                 \c__stex_path_up_str
335               }
336             }{

```

```

337         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
338     }
339 }
340 }{
341     \str_if_empty:NF \l_tmpa_tl {
342         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
343     }
344 }
345 }
346 }
347 \seq_gset_eq:NN #1 \l_tmpa_seq
348 }
349 }

```

(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}`

```

350 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
351     \seq_if_empty:NTF #1 {
352         \prg_return_false:
353     }{
354         \seq_get_left:NN #1 \l_tmpa_tl
355         \sys_if_platform_windows:TF{
356             \str_if_in:NnTF \l_tmpa_tl {:}{
357                 \prg_return_true:
358             }{
359                 \prg_return_false:
360             }
361         }{
362             \str_if_empty:NTF \l_tmpa_tl {
363                 \prg_return_true:
364             }{
365                 \prg_return_false:
366             }
367         }
368     }
369 }

```

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

26.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

370 \str_new:N\l_stex_kpsewhich_return_str
371 \cs_new_protected:Nn \stex_kpsewhich:n {
372     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
373     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
374     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
375 }

```

(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`

```

376 \sys_if_platform_windows:TF{
377   \begingroup\escapechar=-1\catcode'\=12
378   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
379   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
380   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
381   }}{
382     \stex_kpsewhich:n{-var-value~PWD}
383   }
384
385   \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
386   \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
387   \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

```

388 <@@=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

```

389 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

390 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
391 \stex_path_from_string:Nn \c_stex_mainfile_seq
392   \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`

```

393 \seq_gclear_new:N\g_stex_currentfile_seq

```

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

`\stex_filestack_push:n`

```

394 \cs_new_protected:Nn \stex_filestack_push:n {
395   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
396   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
397     \stex_path_from_string:Nn\g_stex_currentfile_seq{
398       \c_stex_pwd_str/#1
399     }
400   }
401   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
402   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
403 }

```


(End definition for `\stex_filestack_push:n`. This function is documented on page 23.)

`\stex_filestack_pop:`

```

404 \cs_new_protected:Nn \stex_filestack_pop: {
405   \seq_if_empty:NF\g__stex_files_stack{
406     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
407   }
408   \seq_if_empty:NTF\g__stex_files_stack{
409     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
410   }{
411     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
412     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
413   }
414 }

```

(End definition for `\stex_filestack_pop:`. This function is documented on page 23.)

Hooks for the current file:

```

415 \AddToHook{file/before}{
416   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
417 }
418 \AddToHook{file/after}{
419   \stex_filestack_pop:
420 }

```

26.4 MathHub Repositories

421 `<@=stex_mathhub>`

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

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

```

422 \str_if_empty:NTF\mathhub{
423   \sys_if_platform_windows:TF{
424     \begingroup\escapechar=-1\catcode'\=12
425     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
426     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
427     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
428   }{
429     \stex_kpsewhich:n{-var-value-MATHHUB}
430   }
431   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
432 }
433 \str_if_empty:NTF\c_stex_mathhub_str{
434   \msg_warning:nn{stex}{warning/nomathhub}
435 }{
436   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
437   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
438 }
439 }{
440   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
441   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
442     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
443       \c_stex_pwd_str/\mathhub
444     }
445   }
446 }

```

```

445 }
446 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
447 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
448 }

```

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

```

449 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
450   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
451     \str_set:Nx \l_tmpa_str { #1 }
452     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
453     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
454     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
455     \_stex_mathhub_find_manifest:N \l_tmpa_seq
456     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
457       \msg_error:nnxx{stex}{error/norepository}{#1}{
458         \stex_path_to_string:N \c_stex_mathhub_str
459       }
460     } {
461       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
462     }
463   }
464 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

465 \seq_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

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

```

466 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
467   \seq_set_eq:NN\l_tmpa_seq #1
468   \bool_set_true:N\l_tmpa_bool
469   \bool_while_do:Nn \l_tmpa_bool {
470     \seq_if_empty:NTF \l_tmpa_seq {
471       \bool_set_false:N\l_tmpa_bool
472     }{
473       \file_if_exist:nTF{
474         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
475       }{
476         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
477         \bool_set_false:N\l_tmpa_bool
478       }{
479         \file_if_exist:nTF{
480           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
481         }{
482           \seq_put_right:Nn\l_tmpa_seq{META-INF}
483           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}

```

```

484         \bool_set_false:N\l_tmpa_bool
485     }{
486         \file_if_exist:nTF{
487             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
488         }{
489             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
490             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
491             \bool_set_false:N\l_tmpa_bool
492         }{
493             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
494         }
495     }
496 }
497 }
498 }
499 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
500 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```
501 \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:

```

502 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
503     \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
504     \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
505     \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
506         \str_set:Nn \l_tmpa_str {##1}
507         \exp_args:NNoo \seq_set_split:Nnn
508             \l_tmpb_seq \c_colon_str \l_tmpa_str
509         \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
510             \exp_args:NNe \str_set:Nn \l_tmpb_tl {
511                 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
512             }
513             \exp_args:No \str_case:nnTF \l_tmpa_tl {
514                 {id} {
515                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
516                     { id } \l_tmpb_tl
517                 }
518                 {narration-base} {
519                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
520                     { narr } \l_tmpb_tl
521                 }
522                 {url-base} {
523                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524                     { docurl } \l_tmpb_tl
525                 }
526                 {source-base} {
527                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528                     { ns } \l_tmpb_tl
529                 }

```

```

530     {ns} {
531       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532       { ns } \l_tmpb_tl
533     }
534     {dependencies} {
535       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
536       { deps } \l_tmpb_tl
537     }
538   }{}{}
539 }{}
540 }
541 \ior_close:N \c__stex_mathhub_manifest_ior
542 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

543 \cs_new_protected:Nn \stex_set_current_repository:n {
544   \stex_require_repository:n { #1 }
545   \prop_set_eq:Nc \l_stex_current_repository_prop {
546     c_stex_mathhub_#1_manifest_prop
547   }
548 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 23.)

`\stex_require_repository:n`

```

549 \cs_new_protected:Nn \stex_require_repository:n {
550   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
551     \stex_debug:nn{mathhub}{Opening~archive:~#1}
552     \_stex_mathhub_do_manifest:n { #1 }
553   }
554 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 23.)

`\l_stex_current_repository_prop`

Current MathHub repository

```

555 %\prop_new:N \l_stex_current_repository_prop
556
557 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
558 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
559   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
560 } {
561   \_stex_mathhub_parse_manifest:n { main }
562   \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
563   \l_tmpa_str
564   \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
565   \c_stex_mathhub_main_manifest_prop
566   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
567   \stex_debug:nn{mathhub}{Current~repository:~
568     \prop_item:Nn \l_stex_current_repository_prop {id}
569   }
570 }

```

(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.

```

571 \cs_new_protected:Nn \stex_in_repository:nn {
572   \str_set:Nx \l_tmpa_str { #1 }
573   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
574   \str_if_empty:NTF \l_tmpa_str {
575     \prop_if_exist:NTF \l_stex_current_repository_prop {
576       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
577       \exp_args:Ne \l_tmpa_cs{
578         \prop_item:Nn \l_stex_current_repository_prop { id }
579       }
580     }{
581       \l_tmpa_cs{}
582     }
583   }{
584     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
585     \stex_require_repository:n \l_tmpa_str
586     \str_set:Nx \l_tmpa_str { #1 }
587     \exp_args:Nne \use:nn {
588       \stex_set_current_repository:n \l_tmpa_str
589       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
590     }{
591       \stex_debug:nn{mathhub}{switching~back~to:~
592       \prop_if_exist:NTF \l_stex_current_repository_prop {
593         \prop_item:Nn \l_stex_current_repository_prop { id }::~
594       \meaning\l_stex_current_repository_prop
595       }{
596         no~repository
597       }
598     }
599     \prop_if_exist:NTF \l_stex_current_repository_prop {
600       \stex_set_current_repository:n {
601         \prop_item:Nn \l_stex_current_repository_prop { id }
602       }
603     }{
604       \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
605     }
606   }
607 }
608 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page [23](#).)

26.5 Using Content in Archives

`\mhpath`

```

609 \def \mhpath #1 #2 {
610   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
611     \c_stex_mathhub_str /
612     \prop_item:Nn \l_stex_current_repository_prop { id }
613     / source / #2
614   }{
615     \c_stex_mathhub_str / #1 / source / #2

```

```

616 }
617 }

```

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

`\inputref`
`\mhinput`

```

618 \newif \ifinputref \inputreffalse
619
620 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
621   \stex_in_repository:nn {#1} {
622     \ifinputref
623       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
624     \else
625       \inputreftrue
626       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
627     \inputreffalse
628   \fi
629 }
630 }
631 \NewDocumentCommand \mhinput { 0{} m}{
632   \stex_mhinput:nn{ #1 }{ #2 }
633 }
634
635 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
636   \stex_in_repository:nn {#1} {
637     \bool_lazy_any:nTF {
638       {\rustex_if_p:}
639       {\latexml_if_p:}
640     } {
641       \str_clear:N \l_tmpa_str
642       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
643         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
644       }
645       \stex_annotate_invisible:nnn{inputref}{
646         \l_tmpa_str / #2
647       }{}
648     }{
649       \begingroup
650         \inputreftrue
651         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
652       \endgroup
653     }
654   }
655 }
656 \NewDocumentCommand \inputref { 0{} m}{
657   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
658 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 24.)

`\addmhbibresource`

```

659 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
660   \stex_in_repository:nn {#1} {
661     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
662   }

```

```

663 }
664 \newcommand\addmhbibresource[2][]{
665   \_stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
666 }

```

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

\libinput

```

667 \cs_new_protected:Npn \libinput #1 {
668   \prop_if_exist:NF \l_stex_current_repository_prop {
669     \msg_error:nnn{stex}{error/notinarchive}\libinput
670   }
671   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
672     \msg_error:nnn{stex}{error/notinarchive}\libinput
673   }
674   \seq_clear:N \l__stex_mathhub_libinput_files_seq
675   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
676   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
677
678   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
679     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
680     \IfFileExists{ \l_tmpa_str }{
681       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
682     }{}
683     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
684     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
685   }
686
687   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
688   \IfFileExists{ \l_tmpa_str }{
689     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
690   }{}
691
692   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
693     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
694   }{
695     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
696       \input{ ##1 }
697     }
698   }
699 }

```

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

\libusepackage

```

700 \NewDocumentCommand \libusepackage {0{} m} {
701   \prop_if_exist:NF \l_stex_current_repository_prop {
702     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
703   }
704   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
705     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
706   }
707   \tl_clear:N \l__stex_mathhub_libinput_files_seq
708   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
709   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

```

```

710
711 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
712   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2.sty}
713   \IfFileExists{ \l_tmpa_str }{
714     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
715   }{}
716   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
717   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
718 }
719
720 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2.sty}
721 \IfFileExists{ \l_tmpa_str }{
722   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
723 }{}
724
725 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
726   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
727 }{
728   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
729     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
730       \usepackage[#1]{ #1 }
731     }
732   }{
733     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
734   }
735 }
736 }

```

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

`\mhgraphics`
`\cmhgraphics`

```

737
738 \AddToHook{begindocument}{
739 \ltx@ifpackageloaded{graphicx}{
740   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
741   \newcommand\mhgraphics[2][]{\%
742     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
743     \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
744   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
745 }{}

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 24.)

`\lstinputmhlisting`
`\clstinputmhlisting`

```

746 \ltx@ifpackageloaded{listings}{
747   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
748   \newcommand\lstinputmhlisting[2][]{\%
749     \def\lst@mhrepos{}\setkeys{lst}{#1}%
750     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
751   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
752 }{}
753 }
754
755 </package>

```


(End definition for \lstinputmhlisting and \clstinputmhlisting. These functions are documented on page 24.)

Chapter 27

STEX -References Implementation

```
756 <*package>
757
758 %%%%%%%%%%% references.dtx %%%%%%%%%%%
759
760 <@@=stex_refs>
761
762 Warnings and error messages
```

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

```
762 \iow_new:N \c__stex_refs_refs_iow
763 \AddToHook{begindocument}{
764   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
765 }
766 \AddToHook{enddocument}{
767   \iow_close:N \c__stex_refs_refs_iow
768 }
```

`\STEXreftitle`

```
769 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
770
771 \NewDocumentCommand \STEXreftitle { m } {
772   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
773 }
```

(End definition for `\STEXreftitle`. This function is documented on page 25.)

27.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
774 \str_new:N \l_stex_current_docns_str
```

(End definition for `\l_stex_current_docns_str`. This variable is documented on page 25.)

`\stex_get_document_uri:`

```
775 \cs_new_protected:Nn \stex_get_document_uri: {  
776   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
777   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
778   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str  
779   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str  
780   \seq_put_right:No \l_tmpa_seq \l_tmpb_str  
781  
782   \str_clear:N \l_tmpa_str  
783   \prop_if_exist:NT \l_stex_current_repository_prop {  
784     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {  
785       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}  
786     }  
787   }  
788  
789   \str_if_empty:NTF \l_tmpa_str {  
790     \str_set:Nx \l_stex_current_docns_str {  
791       file:/\stex_path_to_string:N \l_tmpa_seq  
792     }  
793   }{  
794     \bool_set_true:N \l_tmpa_bool  
795     \bool_while_do:Nn \l_tmpa_bool {  
796       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str  
797       \exp_args:No \str_case:nnTF { \l_tmpb_str } {  
798         {source} { \bool_set_false:N \l_tmpa_bool }  
799       }{}{  
800         \seq_if_empty:NT \l_tmpa_seq {  
801           \bool_set_false:N \l_tmpa_bool  
802         }  
803       }  
804     }  
805  
806     \seq_if_empty:NTF \l_tmpa_seq {  
807       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str  
808     }{  
809       \str_set:Nx \l_stex_current_docns_str {  
810         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq  
811       }  
812     }  
813   }  
814 }
```

(End definition for `\stex_get_document_uri:`. This function is documented on page 25.)

`\l_stex_current_docurl_str`

```
815 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 25.)

`\stex_get_document_url:`

```
816 \cs_new_protected:Nn \stex_get_document_url: {  
817   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
818   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
819   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

820 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
821 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
822
823 \str_clear:N \l_tmpa_str
824 \prop_if_exist:NT \l_stex_current_repository_prop {
825   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
826     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
827       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
828     }
829   }
830 }
831
832 \str_if_empty:NTF \l_tmpa_str {
833   \str_set:Nx \l_stex_current_docurl_str {
834     file:/\stex_path_to_string:N \l_tmpa_seq
835   }
836 }{
837   \bool_set_true:N \l_tmpa_bool
838   \bool_while_do:Nn \l_tmpa_bool {
839     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
840     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
841       {source} { \bool_set_false:N \l_tmpa_bool }
842     }{}{
843       \seq_if_empty:NT \l_tmpa_seq {
844         \bool_set_false:N \l_tmpa_bool
845       }
846     }
847   }
848
849   \seq_if_empty:NTF \l_tmpa_seq {
850     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
851   }{
852     \str_set:Nx \l_stex_current_docurl_str {
853       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
854     }
855   }
856 }
857 }

```

(End definition for `\stex_get_document_url`:. This function is documented on page 25.)

27.2 Setting Reference Targets

```

858 \str_const:Nn \c__stex_refs_url_str{URL}
859 \str_const:Nn \c__stex_refs_ref_str{REF}
860 \str_new:N \l__stex_refs_curr_label_str
861 % @currentlabel -> number
862 % @currentlabelname -> title
863 % @currentHref -> name.number <- id of some kind
864 % \theH# -> \arabic{section}
865 % \the# -> number
866 % \hyper@makecurrent{#}
867 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

868 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
869   \stex_get_document_uri:
870   \str_clear:N \l__stex_refs_curr_label_str
871   \str_set:Nx \l_tmpa_str { #1 }
872   \str_if_empty:NT \l_tmpa_str {
873     \int_incr:N \l__stex_refs_unnamed_counter_int
874     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
875   }
876   \str_set:Nx \l__stex_refs_curr_label_str {
877     \l_stex_current_docns_str?\l_tmpa_str
878   }
879   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
880     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
881   }
882   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
883     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
884   }
885   \stex_if_smsmode:TF {
886     \stex_get_document_url:
887     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
888     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
889   }{
890     \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~\expandafter\unexpanded\expandafter{
891       \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
892       \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
893       \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
894     }
895   }

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 25.)

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

```

896 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
897   \str_set:Nn \l_tmpa_str {#1?#2}
898   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
899   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
900     \seq_new:c {g__stex_refs_labels_#2_seq}
901   }
902   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
903     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
904   }
905 }

```

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

```

906 \AtEndDocument{
907   \def\stexauxadddocref#1 #2 {}{}
908 }

```

`\stex_ref_new_sym_target:n`

```

909 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
910   \stex_if_smsmode:TF {
911     \str_if_exist:cF{sref_sym_#1_type}{
912       \stex_get_document_url:
913       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

914     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
915   }
916 }{
917   \str_if_empty:NF \l__stex_refs_curr_label_str {
918     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
919     \immediate\write\@auxout{
920       \exp_not:N\expandafter\def\exp_not:N\csname sref_sym_#1_label_str\exp_not:N\endcsname
921         \l__stex_refs_curr_label_str
922     }
923   }
924 }
925 }
926 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 25.)

27.3 Using References

```

927 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

928
929 \keys_define:nn { stex / sref } {
930   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
931   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
932   pre           .tl_set:N = \l__stex_refs_pre_tl ,
933   post          .tl_set:N = \l__stex_refs_post_tl ,
934 }
935 \cs_new_protected:Nn \__stex_refs_args:n {
936   \tl_clear:N \l__stex_refs_linktext_tl
937   \tl_clear:N \l__stex_refs_fallback_tl
938   \tl_clear:N \l__stex_refs_pre_tl
939   \tl_clear:N \l__stex_refs_post_tl
940   \str_clear:N \l__stex_refs_repo_str
941   \keys_set:nn { stex / sref } { #1 }
942 }

```

The actual macro:

```

943 \NewDocumentCommand \sref { 0{} m}{
944   \__stex_refs_args:n { #1 }
945   \str_if_empty:NTF \l__stex_refs_indocument_str {
946     \str_set:Nx \l_tmpa_str { #2 }
947     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
948     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
949       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
950         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
951           \str_clear:N \l_tmpa_str
952         }
953       }{
954         \str_clear:N \l_tmpa_str
955       }
956     }{
957       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
958       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

959 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
960 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
961   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
962   \str_clear:N \l_tmpa_str
963   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
964     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
965       \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
966     }{
967       \seq_map_break:n {
968         \str_set:Nn \l_tmpa_str { ##1 }
969       }
970     }
971   }
972 }{
973   \str_clear:N \l_tmpa_str
974 }
975 }
976 \str_if_empty:NTF \l_tmpa_str {
977   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
978 }{
979   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
980     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
981       \cs_if_exist:cTF{autoref}{
982         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
983       }{
984         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
985       }
986     }{
987       \ltx@ifpackageloaded{hyperref}{
988         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
989       }{
990         \l__stex_refs_linktext_tl
991       }
992     }
993   }{
994     \ltx@ifpackageloaded{hyperref}{
995       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
996     }{
997       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
998     }
999   }
1000 }
1001 }{
1002   % TODO
1003 }
1004 }

```

(End definition for `\sref`. This function is documented on page 26.)

`\srefsym`

```

1005 \NewDocumentCommand \srefsym { 0{} m}{
1006   \stex_get_symbol:n { #2 }
1007   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1008 }

```

```

1009
1010 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1011   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1012     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1013   }{
1014     \__stex_refs_args:n { #1 }
1015     \str_if_empty:NTF \l__stex_refs_indocument_str {
1016       \tl_if_exist:cTF{sref_sym_#2 _type}{
1017         % doc uri in \l_tmpb_str
1018         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1019         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1020           % reference
1021           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1022             \cs_if_exist:cTF{autoref}{
1023               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1024             }{
1025               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1026             }
1027           }{
1028             \ltx@ifpackageloaded{hyperref}{
1029               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1030             }{
1031               \l__stex_refs_linktext_tl
1032             }
1033           }
1034         }{
1035           % URL
1036           \ltx@ifpackageloaded{hyperref}{
1037             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1038           }{
1039             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1040           }
1041         }
1042       }{
1043         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1044       }
1045     }{
1046       % TODO
1047     }
1048   }
1049 }

```

(End definition for `\srefsym`. This function is documented on page 26.)

`\srefsymuri`

```

1050 \cs_new_protected:Npn \srefsymuri #1 #2 {
1051   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1052 }

```

(End definition for `\srefsymuri`. This function is documented on page 26.)

```

1053 </package>

```


Chapter 28

STEX -Modules Implementation

```
1054 <*package>
1055
1056 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1057
1058 <@@=stex_modules>
1059
1060     Warnings and error messages
1061 \msg_new:nnn{stex}{error/unknownmodule}{
1062     No~module~#1~found
1063 }
1064 \msg_new:nnn{stex}{error/syntax}{
1065     Syntax~error:~#1
1066 }
1067 \msg_new:nnn{stex}{error/siglanguage}{
1068     Module~#1~declares~signature~#2,~but~does~not~
1069     declare~its~language
1070 }
1071 \msg_new:nnn{stex}{warning/deprecated}{
1072     #1~is~deprecated;~please~use~#2~instead!
1073 }
1074 \msg_new:nnn{stex}{error/conflictingmodules}{
1075     Conflicting~imports~for~module~#1
1076 }
1077
1078 \l_stex_current_module_str The current module:
1079 \str_new:N \l_stex_current_module_str
1080
1081 (End definition for \l_stex_current_module_str. This variable is documented on page 28.)
1082
1083 \l_stex_all_modules_seq Stores all available modules
1084 \seq_new:N \l_stex_all_modules_seq
1085
1086 (End definition for \l_stex_all_modules_seq. This variable is documented on page 28.)
```

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1078 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1079   \str_if_empty:NTF \l_stex_current_module_str
1080   \prg_return_false: \prg_return_true:
1081 }

```

(End definition for \stex_if_in_module:TF. This function is documented on page 28.)

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1082 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1083   \prop_if_exist:cTF { c_stex_module_#1_prop }
1084   \prg_return_true: \prg_return_false:
1085 }

```

(End definition for \stex_if_module_exists:nTF. This function is documented on page 28.)

\stex_add_to_current_module:n Only allowed within modules:

```

\STEXexport
1086 \cs_new_protected:Nn \stex_add_to_current_module:n {
1087   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1088 }
1089 \cs_new_protected:Npn \STEXexport {
1090   \begingroup
1091   \newlinechar=-1\relax
1092   \endlinechar=-1\relax
1093   %\catcode'\ = 9\relax
1094   \expandafter\endgroup\__stex_modules_export:n
1095 }
1096 \cs_new_protected:Nn \__stex_modules_export:n {
1097   \ignorespaces #1
1098   \stex_add_to_current_module:n { \ignorespaces #1 }
1099   \stex_smsmode_do:
1100 }
1101 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented on page 28.)

```

\stex_add_constant_to_current_module:n
1102 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1103   \str_set:Nx \l_tmpa_str { #1 }
1104   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1105 }

```

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page 28.)

```

\stex_add_import_to_current_module:n
1106 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1107   \str_set:Nx \l_tmpa_str { #1 }
1108   \exp_args:Nno
1109   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1110     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1111   }
1112 }

```

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

`\stex_collect_imports:n`

```

1113 \cs_new_protected:Nn \stex_collect_imports:n {
1114   \seq_clear:N \l_stex_collect_imports_seq
1115   \__stex_modules_collect_imports:n {#1}
1116 }
1117 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1118   \seq_map_inline:cn {c_stex_module_#1_imports} {
1119     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1120       \__stex_modules_collect_imports:n { ##1 }
1121     }
1122   }
1123   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1124     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1125   }
1126 }

```

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

`\stex_do_up_to_module:n`

```

1127 \int_new:N \l__stex_modules_group_depth_int
1128 \tl_new:N \l__stex_modules_aftergroup_tl
1129 \cs_new_protected:Nn \stex_do_up_to_module:n {
1130   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1131     #1
1132   }{
1133     #1
1134     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1135       \aftergroup\__stex_modules_aftergroup_do:
1136     }
1137   }
1138   \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1139     \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1140       \l__stex_modules_aftergroup_tl
1141       \tl_clear:N \l__stex_modules_aftergroup_tl
1142     }{
1143       \l__stex_modules_aftergroup_tl
1144       \aftergroup\__stex_modules_aftergroup_do:
1145     }
1146   }

```

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

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

1147

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

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

```

1148 \str_new:N \l_stex_modules_ns_str
1149 \str_new:N \l_stex_modules_subpath_str

```

```

1150 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1151   \str_set:Nx \l_tmpa_str { #1 }
1152   \seq_set_eq:NN \l_tmpa_seq #2
1153   % split off file extension
1154   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1155   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1156   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1157   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1158
1159   \bool_set_true:N \l_tmpa_bool
1160   \bool_while_do:Nn \l_tmpa_bool {
1161     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1162     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1163       {source} { \bool_set_false:N \l_tmpa_bool }
1164     }{}{
1165       \seq_if_empty:NT \l_tmpa_seq {
1166         \bool_set_false:N \l_tmpa_bool
1167       }
1168     }
1169   }
1170
1171   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1172   \str_if_empty:NTF \l_stex_modules_subpath_str {
1173     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1174   }{
1175     \str_set:Nx \l_stex_modules_ns_str {
1176       \l_tmpa_str/\l_stex_modules_subpath_str
1177     }
1178   }
1179 }
1180
1181 \cs_new_protected:Nn \stex_modules_current_namespace: {
1182   \str_clear:N \l_stex_modules_subpath_str
1183   \prop_if_exist:NTF \l_stex_current_repository_prop {
1184     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1185     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1186   }{
1187     % split off file extension
1188     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1189     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1190     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1191     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1192     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1193     \str_set:Nx \l_stex_modules_ns_str {
1194       file:\stex_path_to_string:N \l_tmpa_seq
1195     }
1196   }
1197 }

```

(End definition for `\stex_modules_current_namespace:`. This function is documented on page 29.)

28.1 The smodule environment

smodule arguments:

```

1198 \keys_define:nn { stex / module } {
1199   title      .tl_set:N      = \smodulename ,
1200   type       .str_set_x:N   = \smoduletype ,
1201   id         .str_set_x:N   = \smoduleid ,
1202   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1203   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1204   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1205   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1206   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1207   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1208   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1209   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1210 }
1211
1212 \cs_new_protected:Nn \__stex_modules_args:n {
1213   \str_clear:N \smodulename
1214   \str_clear:N \smoduletype
1215   \str_clear:N \smoduleid
1216   \str_clear:N \l_stex_module_ns_str
1217   \str_clear:N \l_stex_module_deprecate_str
1218   \str_clear:N \l_stex_module_lang_str
1219   \str_clear:N \l_stex_module_sig_str
1220   \str_clear:N \l_stex_module_creators_str
1221   \str_clear:N \l_stex_module_contributors_str
1222   \str_clear:N \l_stex_module_meta_str
1223   \str_clear:N \l_stex_module_srccite_str
1224   \keys_set:nn { stex / module } { #1 }
1225 }
1226
1227 % module parameters here? In the body?
1228
```

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

```

1229 \cs_new_protected:Nn \stex_module_setup:nn {
1230   \str_set:Nx \l_stex_module_name_str { #2 }
1231   \__stex_modules_args:n { #1 }
1232
1233   First, we set up the name and namespace of the module.
1234   Are we in a nested module?
1235
1236   \stex_if_in_module:TF {
1237     % Nested module
1238     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1239     { ns } \l_stex_module_ns_str
1240     \str_set:Nx \l_stex_module_name_str {
1241       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1242       { name } / \l_stex_module_name_str
1243     }
1244   }{
1245     % not nested:
1246     \str_if_empty:NT \l_stex_module_ns_str {
1247       \stex_modules_current_namespace:
1248     }
1249   }

```

```

1244 \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1245 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1246 / {\l_stex_module_ns_str}
1247 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1248 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1249 \str_set:Nx \l_stex_module_ns_str {
1250 \stex_path_to_string:N \l_tmpa_seq
1251 }
1252 }
1253 }
1254 }

```

Next, we determine the language of the module:

```

1255 \str_if_empty:NT \l_stex_module_lang_str {
1256 \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1257 \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1258 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1259 \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1260 \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1261 \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1262 inferred~from~file~name}
1263 \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1264 }
1265 }
1266
1267 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1268 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1269 \l_tmpa_str {
1270 \ltx@ifpackageloaded{babel}{
1271 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1272 }{}
1273 } {
1274 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1275 }
1276 }}

```

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

```

1277 \str_if_empty:NTF \l_stex_module_sig_str {
1278 \exp_args:Nnx \prop_gset_from_keyval:cn {
1279 c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1280 } {
1281 name = \l_stex_module_name_str ,
1282 ns = \l_stex_module_ns_str ,
1283 file = \exp_not:o { \g_stex_currentfile_seq } ,
1284 lang = \l_stex_module_lang_str ,
1285 sig = \l_stex_module_sig_str ,
1286 deprecate = \l_stex_module_deprecate_str ,
1287 meta = \l_stex_module_meta_str
1288 }
1289 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1290 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1291 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1292 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1293 \str_if_empty:NT \l_stex_module_meta_str {
1294   \str_set:Nx \l_stex_module_meta_str {
1295     \c_stex_metatheory_ns_str ? Metatheory
1296   }
1297 }
1298 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1299   \bool_set_true:N \l_stex_in_meta_bool
1300   \exp_args:Nx \stex_add_to_current_module:n {
1301     \bool_set_true:N \l_stex_in_meta_bool
1302     \stex_activate_module:n {\l_stex_module_meta_str}
1303     \bool_set_false:N \l_stex_in_meta_bool
1304   }
1305   \stex_activate_module:n {\l_stex_module_meta_str}
1306   \bool_set_false:N \l_stex_in_meta_bool
1307 }
1308 }{
1309   \str_if_empty:NT \l_stex_module_lang_str {
1310     \msg_error:nnxx{stex}{error/siglanguage}{
1311       \l_stex_module_ns_str?\l_stex_module_name_str
1312     }{\l_stex_module_sig_str}
1313   }
1314
1315   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1316   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1317   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1318   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1319   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1320   \str_set:Nx \l_tmpa_str {
1321     \stex_path_to_string:N \l_tmpa_seq /
1322     \l_tmpa_str . \l_stex_module_sig_str .tex
1323   }
1324   \IfFileExists \l_tmpa_str {
1325     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1326       \str_clear:N \l_stex_current_module_str
1327       \seq_clear:N \l_stex_all_modules_seq
1328       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1329     }
1330   }{
1331     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1332   }
1333   \stex_if_smsmode:F {
1334     \stex_activate_module:n {
1335       \l_stex_module_ns_str ? \l_stex_module_name_str
1336     }
1337   }
1338   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1339 }
1340 \str_if_empty:NF \l_stex_module_deprecate_str {
1341   \msg_warning:nnxx{stex}{warning/deprecated}{
1342     Module~\l_stex_current_module_str
1343   }{
1344     \l_stex_module_deprecate_str
1345   }

```

```

1346 }
1347 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 29.)

smodule The module environment.

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

```

1348 \cs_new_protected:Nn \__stex_modules_begin_module: {
1349   \stex_reactivate_macro:N \STEXexport
1350   \stex_reactivate_macro:N \importmodule
1351   \stex_reactivate_macro:N \symdecl
1352   \stex_reactivate_macro:N \notation
1353   \stex_reactivate_macro:N \symdef
1354
1355   \stex_debug:nn{modules}{
1356     New~module:\\
1357     Namespace:~\l_stex_module_ns_str\\
1358     Name:~\l_stex_module_name_str\\
1359     Language:~\l_stex_module_lang_str\\
1360     Signature:~\l_stex_module_sig_str\\
1361     Metatheory:~\l_stex_module_meta_str\\
1362     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1363   }
1364
1365   \seq_put_right:Nx \l_stex_all_modules_seq {
1366     \l_stex_module_ns_str ? \l_stex_module_name_str
1367   }
1368
1369   \stex_if_smsmode:F{
1370     \begin{stex_annotate_env} {theory} {
1371       \l_stex_module_ns_str ? \l_stex_module_name_str
1372     }
1373
1374     \stex_annotate_invisible:nnn{header}{} {
1375       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1376       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1377       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1378         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1379       }
1380       \str_if_empty:NF \smoduletype {
1381         \stex_annotate:nnn{type}{\smoduletype}{}
1382       }
1383     }
1384   }
1385   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1386   % TODO: Inherit metatheory for nested modules?
1387 }
1388 \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{smodule}`

```

1389 \cs_new_protected:Nn \__stex_modules_end_module: {

```



```

1390 \stex_debug:nn{modules}{Closing-module~\prop_item:cn {c_stex_module_\l_stex_current_module
1391 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1392 \iffalse \begin{stex_annotate_env} \fi %%^A make syntax highlighting work again
1393 \NewDocumentEnvironment { smodule } { 0{} m } {
1394   \stex_module_setup:nn{#1}{#2}
1395   \par
1396   \stex_if_smsmode:F{
1397     \tl_clear:N \l_tmpa_tl
1398     \clist_map_inline:Nn \smodulotype {
1399       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1400         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1401       }
1402     }
1403     \tl_if_empty:NTF \l_tmpa_tl {
1404       \__stex_modules_smodule_start:
1405     }{
1406       \l_tmpa_tl
1407     }
1408   }
1409   \__stex_modules_begin_module:
1410   \str_if_empty:NF \smoduleid {
1411     \stex_ref_new_doc_target:n \smoduleid
1412   }
1413   \stex_smsmode_do:
1414 } {
1415   \__stex_modules_end_module:
1416   \stex_if_smsmode:F {
1417     \end{stex_annotate_env}
1418     \clist_set:Nn \l_tmpa_clist \smodulotype
1419     \tl_clear:N \l_tmpa_tl
1420     \clist_map_inline:Nn \l_tmpa_clist {
1421       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1422         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1423       }
1424     }
1425     \tl_if_empty:NTF \l_tmpa_tl {
1426       \__stex_modules_smodule_end:
1427     }{
1428       \l_tmpa_tl
1429     }
1430   }
1431 }

```

\stexpatchmodule

```

1432 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1433 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1434
1435 \newcommand\stexpatchmodule[3] [] {
1436   \str_set:Nx \l_tmpa_str{ #1 }
1437   \str_if_empty:NTF \l_tmpa_str {
1438     \tl_set:Nn \__stex_modules_smodule_start: { #2 }

```

```

1439     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1440   }{
1441     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1442     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1443   }
1444 }

```

(End definition for `\stexpatchmodule`. This function is documented on page 29.)

28.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1445 \NewDocumentCommand \STEXModule { m } {
1446   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1447   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1448   \tl_set:Nn \l_tmpa_tl {
1449     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1450   }
1451   \seq_map_inline:Nn \l_stex_all_modules_seq {
1452     \str_set:Nn \l_tmpb_str { ##1 }
1453     \str_if_eq:eeT { \l_tmpa_str } {
1454       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1455     } {
1456       \seq_map_break:n {
1457         \tl_set:Nn \l_tmpa_tl {
1458           \stex_invoke_module:n { ##1 }
1459         }
1460       }
1461     }
1462   }
1463   \l_tmpa_tl
1464 }
1465
1466 \cs_new_protected:Nn \stex_invoke_module:n {
1467   \stex_debug:nn{modules}{Invoking~module~#1}
1468   \peek_charcode_remove:NTF ! {
1469     \__stex_modules_invoke_uri:nN { #1 }
1470   } {
1471     \peek_charcode_remove:NTF ? {
1472       \__stex_modules_invoke_symbol:nn { #1 }
1473     } {
1474       \msg_error:nnx{stex}{error/syntax}{
1475         ?~or~!~expected~after~
1476         \c_backslash_str STEXModule{#1}
1477       }
1478     }
1479   }
1480 }
1481
1482 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1483   \str_set:Nn #2 { #1 }
1484 }
1485

```

```

1486 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1487   \stex_invoke_symbol:n{#1?#2}
1488 }

```

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

\stex_activate_module:n

```

1489 \bool_new:N \l_stex_in_meta_bool
1490 \bool_set_false:N \l_stex_in_meta_bool
1491 \cs_new_protected:Nn \stex_activate_module:n {
1492   \stex_debug:nn{modules}{Activating~module~#1}
1493   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1494     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1495   }
1496   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1497     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1498     \use:c{ c_stex_module_#1_code }
1499   }
1500 }

```

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

```

1501 \</package>

```

Chapter 29

STEX -Module Inheritance Implementation

```
1502 ⟨*package⟩
1503
1504 %%%%%%%%% inheritance.dtx %%%%%%%%%
1505
```

29.1 SMS Mode

```
1506 ⟨@@=stex_smsmode⟩

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1507 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1508 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1509 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1510
1511 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1512   \makeatletter
1513   \makeatother
1514   \ExplSyntaxOn
1515   \ExplSyntaxOff
1516   \rustexBREAK
1517 }
1518
1519 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1520   \symdef
1521   \importmodule
1522   \notation
1523   \symdecl
1524   \STEXexport
1525   \inlineass
1526   \inlinedef
1527   \inlineex
1528   \endinput
1529   \setnotation
```

```

1530 \copynotation
1531 }
1532
1533 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1534 \tl_to_str:n {
1535     smodule,
1536     copymodule,
1537     interpretmodule,
1538     sdefinition,
1539     sexample,
1540     sassertion,
1541     sparagraph
1542 }
1543 }

```

(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`

```

1544 \bool_new:N \g__stex_smsmode_bool
1545 \bool_set_false:N \g__stex_smsmode_bool
1546 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1547 \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1548 }

```

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

`\stex_in_smsmode:nn`

```

1549 \cs_new_protected:Nn \stex_in_smsmode:nn {
1550 \vbox_set:Nn \l_tmpa_box {
1551 \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1552 \bool_gset_true:N \g__stex_smsmode_bool
1553 #2
1554 \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1555 }
1556 \box_clear:N \l_tmpa_box
1557 }
1558
1559 \quark_new:N \q__stex_smsmode_break
1560
1561 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1562 \stex_filestack_push:n{#1}
1563 \stex_in_smsmode:nn{#1} {
1564 #2
1565 \everyeof{\q__stex_smsmode_break\noexpand}
1566 \expandafter\expandafter\expandafter
1567 \stex_smsmode_do:
1568 \csname @ @ input\endcsname "#1"\relax
1569 }
1570 \stex_filestack_pop:
1571 }

```

(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:

```

1572 \cs_new_protected:Npn \stex_smsmode_do: {
1573   \stex_if_smsmode:T {
1574     \__stex_smsmode_do:w
1575   }
1576 }
1577 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1578   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1579     \expandafter\if\expandafter\relax\noexpand#1
1580     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1581     \else\expandafter\__stex_smsmode_do:w\fi
1582   }{
1583     \__stex_smsmode_do:w % #1
1584   }
1585 }
1586 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1587   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1588     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1589       #1\__stex_smsmode_do:w
1590     }{
1591       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1592         #1
1593       }{
1594         \cs_if_eq:NNTF \begin #1 {
1595           \__stex_smsmode_check_begin:n
1596         }{
1597           \cs_if_eq:NNTF \end #1 {
1598             \__stex_smsmode_check_end:n
1599           }{
1600             \__stex_smsmode_do:w
1601           }
1602         }
1603       }
1604     }
1605   }
1606 }
1607
1608 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1609   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1610     \begin{#1}
1611   }{
1612     \__stex_smsmode_do:w
1613   }
1614 }
1615 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1616   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1617     \end{#1}\__stex_smsmode_do:w
1618   }{
1619     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1620   }
1621 }

```

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

29.2 Inheritance

1622 <@@=stex_importmodule>

\stex_import_module_uri:nn

```

1623 \cs_new_protected:Nn \stex_import_module_uri:nn {
1624   \str_set:Nx \l_stex_import_archive_str { #1 }
1625   \str_set:Nn \l_stex_import_path_str { #2 }
1626
1627   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1628   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1629   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1630
1631   \stex_modules_current_namespace:
1632   \bool_lazy_all:nTF {
1633     {\str_if_empty_p:N \l_stex_import_archive_str}
1634     {\str_if_empty_p:N \l_stex_import_path_str}
1635     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1636   }{
1637     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1638     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1639   }{
1640     \str_if_empty:NT \l_stex_import_archive_str {
1641       \prop_if_exist:NT \l_stex_current_repository_prop {
1642         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1643       }
1644     }
1645     \str_if_empty:NTF \l_stex_import_archive_str {
1646       \str_if_empty:NF \l_stex_import_path_str {
1647         \str_set:Nx \l_stex_import_ns_str {
1648           \l_stex_module_ns_str / \l_stex_import_path_str
1649         }
1650       }
1651     }{
1652       \stex_require_repository:n \l_stex_import_archive_str
1653       \prop_get:cnN { c_stex_mathhub\l_stex_import_archive_str_manifest_prop } { ns }
1654       \l_stex_import_ns_str
1655       \str_if_empty:NF \l_stex_import_path_str {
1656         \str_set:Nx \l_stex_import_ns_str {
1657           \l_stex_import_ns_str / \l_stex_import_path_str
1658         }
1659       }
1660     }
1661   }
1662 }
```

(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	1663 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str	1664 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str	1665 \str_new:N \l_stex_import_path_str
	1666 \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:nnnn      {\<ns>} {\<archive-ID>} {\<path>} {\<name>}}
1667 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1668   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1669
1670     % archive
1671     \str_set:Nx \l_tmpa_str { #2 }
1672     \str_if_empty:NTF \l_tmpa_str {
1673       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1674     } {
1675       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1676       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1677       \seq_put_right:Nn \l_tmpa_seq { source }
1678     }
1679
1680     % path
1681     \str_set:Nx \l_tmpb_str { #3 }
1682     \str_if_empty:NTF \l_tmpb_str {
1683       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1684
1685       \ltx@ifpackageloaded{babel} {
1686         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1687           { \language } \l_tmpb_str {
1688           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1689         }
1690       } {
1691         \str_clear:N \l_tmpb_str
1692       }
1693
1694       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1695       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1696         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1697       }{
1698         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1699         \IfFileExists{ \l_tmpa_str.tex }{
1700           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1701         }{
1702           % try english as default
1703           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1704           \IfFileExists{ \l_tmpa_str.en.tex }{
1705             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1706           }{
1707             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1708           }
1709         }
1710       }
1711
1712     } {
1713       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1714       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1715
1716       \ltx@ifpackageloaded{babel} {
1717         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1718           { \language } \l_tmpb_str {
1719           \msg_error:nnx{stex}{error/unknownlanguage}{\language}

```



```

1720     }
1721 } {
1722     \str_clear:N \l_tmpb_str
1723 }
1724
1725 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1726
1727 \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1728 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1729     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1730 }{
1731     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1732     \IfFileExists{ \l_tmpa_str/#4.tex }{
1733         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1734     }{
1735         % try english as default
1736         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1737         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1738             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1739         }{
1740             \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1741             \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1742                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1743             }{
1744                 \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1745                 \IfFileExists{ \l_tmpa_str.tex }{
1746                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1747                 }{
1748                     % try english as default
1749                     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1750                     \IfFileExists{ \l_tmpa_str.en.tex }{
1751                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1752                     }{
1753                         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1754                     }
1755                 }
1756             }
1757         }
1758     }
1759 }
1760 }
1761
1762 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1763     \seq_clear:N \l_stex_all_modules_seq
1764     \str_clear:N \l_stex_current_module_str
1765     \str_set:Nx \l_tmpb_str { #2 }
1766     \str_if_empty:NF \l_tmpb_str {
1767         \stex_set_current_repository:n { #2 }
1768     }
1769     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1770 }
1771
1772 \stex_if_module_exists:nF { #1 ? #4 } {
1773     \msg_error:nnx{stex}{error/unknownmodule}{

```

```

1774         #1?#4~(in~file~\g__stex_importmodule_file_str)
1775     }
1776 }
1777 }
1778 \stex_activate_module:n { #1 ? #4 }
1779 }

```

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

`\importmodule`

```

1780 \NewDocumentCommand \importmodule { 0{} m } {
1781   \stex_import_module_uri:nn { #1 } { #2 }
1782   \stex_debug:nn{modules}{Importing~module:~
1783     \l_stex_import_ns_str ? \l_stex_import_name_str
1784   }
1785   \stex_if_smsmode:F {
1786     \stex_import_require_module:nnnn
1787     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1788     { \l_stex_import_path_str } { \l_stex_import_name_str }
1789     \stex_annotate_invisible:nnn
1790     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1791   }
1792   \exp_args:Nx \stex_add_to_current_module:n {
1793     \stex_import_require_module:nnnn
1794     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1795     { \l_stex_import_path_str } { \l_stex_import_name_str }
1796   }
1797   \exp_args:Nx \stex_add_import_to_current_module:n {
1798     \l_stex_import_ns_str ? \l_stex_import_name_str
1799   }
1800   \stex_smsmode_do:
1801   \ignorespacesandpars
1802 }
1803 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1804 \NewDocumentCommand \usemodule { 0{} m } {
1805   \stex_if_smsmode:F {
1806     \stex_import_module_uri:nn { #1 } { #2 }
1807     \stex_import_require_module:nnnn
1808     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1809     { \l_stex_import_path_str } { \l_stex_import_name_str }
1810     \stex_annotate_invisible:nnn
1811     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1812   }
1813   \stex_smsmode_do:
1814   \ignorespacesandpars
1815 }

```

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

```

1816 </package>

```

Chapter 30

STEX -Symbols Implementation

```
1817 <*package>
1818
1819 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1820
    Warnings and error messages
1821 \msg_new:nnn{stex}{error/wrongargs}{
1822   args~value~in~symbol~declaration~for~#1~
1823   needs~to~be~i,~a,~b~or~B,~but~#2~given
1824 }
```

30.1 Symbol Declarations

```
1825 <@@=stex_symdecl>

\l_stex_all_symbols_seq Stores all available symbols
1826 \seq_new:N \l_stex_all_symbols_seq

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

\STEXsymbol
1827 \NewDocumentCommand \STEXsymbol { m } {
1828   \stex_get_symbol:n { #1 }
1829   \exp_args:No
1830   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1831 }

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

1832 \keys_define:nn { stex / symdecl } {
1833   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1834   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1835   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1836   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1837   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1838   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
```

```

1839 gfc .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1840 specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1841 def .tl_set:N = \l_stex_symdecl_definiens_tl ,
1842 assoc .choices:nn =
1843 {bin,binl,binr,pre,conj,pwconj}
1844 {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}}
1845 }
1846
1847 \bool_new:N \l_stex_symdecl_make_macro_bool
1848
1849 \cs_new_protected:Nn \__stex_symdecl_args:n {
1850 \str_clear:N \l_stex_symdecl_name_str
1851 \str_clear:N \l_stex_symdecl_args_str
1852 \str_clear:N \l_stex_symdecl_deprecate_str
1853 \str_clear:N \l_stex_symdecl_assoc_type_str
1854 \bool_set_false:N \l_stex_symdecl_local_bool
1855 \tl_clear:N \l_stex_symdecl_type_tl
1856 \tl_clear:N \l_stex_symdecl_definiens_tl
1857
1858 \keys_set:nn { stex / symdecl } { #1 }
1859 }

```

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

```

1860
1861 \NewDocumentCommand \symdecl { s O{} m } {
1862 \__stex_symdecl_args:n { #2 }
1863 \IfBooleanTF #1 {
1864 \bool_set_false:N \l_stex_symdecl_make_macro_bool
1865 } {
1866 \bool_set_true:N \l_stex_symdecl_make_macro_bool
1867 }
1868 \stex_symdecl_do:n { #3 }
1869 \stex_smsmode_do:
1870 }
1871
1872 \cs_new_protected:Nn \stex_symdecl_do:nn {
1873 \__stex_symdecl_args:n{#1}
1874 \bool_set_false:N \l_stex_symdecl_make_macro_bool
1875 \stex_symdecl_do:n{#2}
1876 }
1877
1878 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1879 \cs_new_protected:Nn \stex_symdecl_do:n {
1880 \stex_if_in_module:F {
1881 % TODO throw error? some default namespace?
1882 }
1883
1884 \str_if_empty:NT \l_stex_symdecl_name_str {
1885 \str_set:Nx \l_stex_symdecl_name_str { #1 }

```

```

1886 }
1887
1888 \prop_if_exist:cT { \l_stex_symdecl_
1889   \l_stex_current_module_str ?
1890   \l_stex_symdecl_name_str
1891   _prop
1892 }{
1893   % TODO throw error (beware of circular dependencies)
1894 }
1895
1896 \prop_clear:N \l_tmpa_prop
1897 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1898 \seq_clear:N \l_tmpa_seq
1899 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1900 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1901
1902 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1903   \str_if_empty:NF \l_stex_module_deprecate_str {
1904     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1905   }
1906 }
1907 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1908
1909 \exp_args:No \stex_add_constant_to_current_module:n {
1910   \l_stex_symdecl_name_str
1911 }
1912
1913 % arity/args
1914 \int_zero:N \l_tmpb_int
1915
1916 \bool_set_true:N \l_tmpa_bool
1917 \str_map_inline:Nn \l_stex_symdecl_args_str {
1918   \token_case_meaning:NnF ##1 {
1919     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1920     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1921     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1922     {\tl_to_str:n a} {
1923       \bool_set_false:N \l_tmpa_bool
1924       \int_incr:N \l_tmpb_int
1925     }
1926     {\tl_to_str:n B} {
1927       \bool_set_false:N \l_tmpa_bool
1928       \int_incr:N \l_tmpb_int
1929     }
1930   }{
1931     \msg_error:nnxx{stex}{error/wrongargs}{
1932       \l_stex_current_module_str ?
1933       \l_stex_symdecl_name_str
1934     }{##1}
1935   }
1936 }
1937 \bool_if:NTF \l_tmpa_bool {
1938   % possibly numeric
1939   \str_if_empty:NTF \l_stex_symdecl_args_str {

```

```

1940     \prop_put:Nnn \l_tmpa_prop { args } {}
1941     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1942   }{
1943     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1944     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1945     \str_clear:N \l_tmpa_str
1946     \int_step_inline:nn \l_tmpa_int {
1947       \str_put_right:Nn \l_tmpa_str i
1948     }
1949     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1950   }
1951 } {
1952   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1953   \prop_put:Nnx \l_tmpa_prop { arity }
1954     { \str_count:N \l_stex_symdecl_args_str }
1955 }
1956 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1957
1958
1959 % semantic macro
1960
1961 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1962   \exp_args:Nx \stex_do_up_to_module:n {
1963     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1964       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1965     }}
1966   }
1967
1968   \bool_if:NF \l_stex_symdecl_local_bool {
1969     \exp_args:Nx \stex_add_to_current_module:n {
1970       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1971         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1972       } }
1973     }
1974   }
1975 }
1976
1977 % add to all symbols
1978
1979 \bool_if:NF \l_stex_symdecl_local_bool {
1980   \exp_args:Nx \stex_add_to_current_module:n {
1981     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1982       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1983     }
1984   }
1985   % \exp_args:Nx \stex_add_field_to_current_module:n {
1986   %   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1987   % }
1988 }
1989
1990 \stex_debug:nn{symbols}{New~symbol:~
1991   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1992   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1993   Args:~\prop_item:Nn \l_tmpa_prop { args }

```

```

1994 }
1995
1996 % circular dependencies require this:
1997
1998 \prop_if_exist:cF {
1999   l_stex_symdecl_
2000   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2001   _prop
2002 } {
2003   \prop_set_eq:cN {
2004     l_stex_symdecl_
2005     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2006     _prop
2007   } \l_tmpa_prop
2008 }
2009
2010 \seq_clear:c {
2011   l_stex_symdecl_
2012   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2013   _notations
2014 }
2015
2016 \bool_if:NF \l_stex_symdecl_local_bool {
2017   \exp_args:Nx
2018   \stex_add_to_current_module:n {
2019     \seq_clear:c {
2020       l_stex_symdecl_
2021       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2022       _notations
2023     }
2024     \prop_set_from_keyval:cn {
2025       l_stex_symdecl_
2026       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2027       _prop
2028     } {
2029       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2030       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2031       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2032       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2033       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2034       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2035     }
2036   }
2037 }
2038
2039 \stex_if_smsmode:F {
2040   \exp_args:Nx \stex_do_up_to_module:n {
2041     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2042       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2043     }
2044   }
2045   \stex_if_do_html:T {
2046     \stex_annotate_invisible:nnn {symdecl} {
2047       \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2048 } {
2049   \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st
2050   \stex_annotate_invisible:nnn{args}{}}{
2051     \prop_item:Nn \l_tmpa_prop { args }
2052   }
2053   \stex_annotate_invisible:nnn{macroname}{#1}{}
2054   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2055     \stex_annotate_invisible:nnn{definiens}{}
2056     {\l_stex_symdecl_definiens_tl$}
2057   }
2058   \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2059     \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2060   }
2061 }
2062 }
2063 }
2064 }

```

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

`\stex_get_symbol:n`

```

2065 \str_new:N \l_stex_get_symbol_uri_str
2066
2067 \cs_new_protected:Nn \stex_get_symbol:n {
2068   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2069     \__stex_symdecl_get_symbol_from_cs:n { #1 }
2070   }{
2071     % argument is a string
2072     % is it a command name?
2073     \cs_if_exist:cTF { #1 }{
2074       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2075       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2076       \str_if_empty:NTF \l_tmpa_str {
2077         \exp_args:Nx \cs_if_eq:NNTF {
2078           \tl_head:N \l_tmpa_tl
2079         } \stex_invoke_symbol:n {
2080           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
2081         }{
2082           \__stex_symdecl_get_symbol_from_string:n { #1 }
2083         }
2084       } {
2085         \__stex_symdecl_get_symbol_from_string:n { #1 }
2086       }
2087     }{
2088       % argument is not a command name
2089       \__stex_symdecl_get_symbol_from_string:n { #1 }
2090       % \l_stex_all_symbols_seq
2091     }
2092   }
2093   \str_if_eq:eeF {
2094     \prop_item:cn {
2095       \l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2096     }{ deprecate }
2097   }{}{

```



```

2098 \msg_warning:nnxx{stex}{warning/deprecated}{
2099 Symbol~\l_stex_get_symbol_uri_str
2100 }{
2101 \prop_item:cn {l_stex_symdecl_l_stex_get_symbol_uri_str _prop}{ deprecate }
2102 }
2103 }
2104 }
2105
2106 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2107 \str_set:Nn \l_tmpa_str { #1 }
2108 \bool_set_false:N \l_tmpa_bool
2109 \stex_if_in_module:T {
2110 \exp_args:Nno \seq_if_in:cnT {c_stex_module_l_stex_current_module_str _constants} { \l_
2111 \bool_set_true:N \l_tmpa_bool
2112 \str_set:Nx \l_stex_get_symbol_uri_str {
2113 \l_stex_current_module_str ? #1
2114 }
2115 }
2116 }
2117 \bool_if:NF \l_tmpa_bool {
2118 \tl_set:Nn \l_tmpa_tl {
2119 \msg_set:nnn{stex}{error/unknownsymbol}{
2120 No~symbol~#1~found!
2121 }
2122 \msg_error:nn{stex}{error/unknownsymbol}
2123 }
2124 \str_set:Nn \l_tmpa_str { #1 }
2125 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2126 \seq_map_inline:Nn \l_stex_all_symbols_seq {
2127 \str_set:Nn \l_tmpb_str { ##1 }
2128 \str_if_eq:eeT { \l_tmpa_str } {
2129 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2130 } {
2131 \seq_map_break:n {
2132 \tl_set:Nn \l_tmpa_tl {
2133 \str_set:Nn \l_stex_get_symbol_uri_str {
2134 ##1
2135 }
2136 }
2137 }
2138 }
2139 }
2140 \l_tmpa_tl
2141 }
2142 }
2143
2144 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2145 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2146 { \tl_tail:N \l_tmpa_tl }
2147 \tl_if_single:NTF \l_tmpa_tl {
2148 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2149 \exp_after:wN \str_set:Nn \exp_after:wN
2150 \l_stex_get_symbol_uri_str \l_tmpa_tl
2151 }{

```

```

2152      % TODO
2153      % tail is not a single group
2154    }
2155  }{
2156    % TODO
2157    % tail is not a single group
2158  }
2159 }

```

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

30.2 Notations

```

2160 <@@=stex_notation>

notation arguments:
2161 \keys_define:nn { stex / notation } {
2162   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2163   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2164   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2165   op        .tl_set:N   = \l__stex_notation_op_tl ,
2166   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2167   primary   .default:n   = {true} ,
2168   unknown   .code:n      = \str_set:Nx
2169               \l__stex_notation_variant_str \l_keys_key_str
2170 }
2171
2172 \cs_new_protected:Nn \_stex_notation_args:n {
2173   \str_clear:N \l__stex_notation_lang_str
2174   \str_clear:N \l__stex_notation_variant_str
2175   \str_clear:N \l__stex_notation_prec_str
2176   \tl_clear:N \l__stex_notation_op_tl
2177   \bool_set_false:N \l__stex_notation_primary_bool
2178
2179   \keys_set:nn { stex / notation } { #1 }
2180 }

```

\notation

```

2181 \NewDocumentCommand \notation { s O{} m } {
2182   \_stex_notation_args:n { #2 }
2183   \tl_clear:N \l_stex_symdecl_definiens_tl
2184   \stex_get_symbol:n { #3 }
2185   \tl_set:Nn \l__stex_notation_after_do_tl {
2186     \__stex_notation_final:
2187     \IfBooleanTF#1{
2188       \stex_setnotation:n {\l__stex_notation_symbol_str}
2189     }{}
2190     \stex_smsmode_do:
2191   }
2192   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2193 }
2194 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

`\stex_notation_do:nn`

```

2195 \seq_new:N \l__stex_notation_precedences_seq
2196 \tl_new:N \l__stex_notation_opprec_tl
2197 \int_new:N \l__stex_notation_currarg_int
2198
2199 \cs_new_protected:Nn \stex_notation_do:nn {
2200   \let\l_stex_current_symbol_str\relax
2201   \str_set:Nx \l__stex_notation_symbol_str { #1 }
2202   \seq_clear:N \l__stex_notation_precedences_seq
2203   \tl_clear:N \l__stex_notation_opprec_tl
2204   \prop_get:cnN {
2205     l_stex_symdecl_ #1 _prop
2206   } { args } \l__stex_notation_args_str
2207
2208   % precedences
2209   \prop_get:cnN {
2210     l_stex_symdecl_ #1 _prop
2211   } { arity } \l__stex_notation_arity_str
2212   \str_if_empty:NTF \l__stex_notation_prec_str {
2213     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2214       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2215     }{
2216       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2217     }
2218   } {
2219     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2220       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2221       \int_step_inline:nn { \l__stex_notation_arity_str } {
2222         \exp_args:NNo
2223         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2224       }
2225     }{
2226       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2227       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2228         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2229         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2230           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2231             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2232           \seq_map_inline:Nn \l_tmpa_seq {
2233             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2234           }
2235         }
2236       }{
2237         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2238           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2239         }{
2240           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2241         }
2242       }
2243     }
2244   }
2245
2246   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2247   \int_step_inline:nn { \l__stex_notation_arity_str } {

```

```

2248 \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
2249 \exp_args:NNo
2250 \seq_put_right:No \l__stex_notation_precedences_seq {
2251 \l__stex_notation_opprec_tl
2252 }
2253 }
2254 }
2255
2256 \tl_clear:N \l__stex_notation_dummyargs_tl
2257
2258 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2259 \exp_args:NNe
2260 \cs_set:Npn \l__stex_notation_macrocode_cs {
2261 \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2262 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2263 { \l__stex_notation_opprec_tl }
2264 { \exp_not:n { #2 } }
2265 }
2266 \l__stex_notation_after_do_tl
2267 }{
2268 \str_if_in:NnTF \l__stex_notation_args_str b {
2269 \exp_args:Nne \use:nn
2270 {
2271 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2272 \cs_set:Npn \l__stex_notation_arity_str } { {
2273 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2274 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2275 { \l__stex_notation_opprec_tl }
2276 { \exp_not:n { #2 } }
2277 } }
2278 }{
2279 \str_if_in:NnTF \l__stex_notation_args_str B {
2280 \exp_args:Nne \use:nn
2281 {
2282 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2283 \cs_set:Npn \l__stex_notation_arity_str } { {
2284 \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2285 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2286 { \l__stex_notation_opprec_tl }
2287 { \exp_not:n { #2 } }
2288 } }
2289 }{
2290 \exp_args:Nne \use:nn
2291 {
2292 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2293 \cs_set:Npn \l__stex_notation_arity_str } { {
2294 \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2295 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2296 { \l__stex_notation_opprec_tl }
2297 { \exp_not:n { #2 } }
2298 } }
2299 }
2300 }
2301

```

```

2302 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2303 \int_zero:N \l__stex_notation_currarg_int
2304 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2305 \__stex_notation_arguments:
2306 }
2307 }

```

(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

```

2308 \cs_new_protected:Nn \__stex_notation_arguments: {
2309 \int_incr:N \l__stex_notation_currarg_int
2310 \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2311 \l__stex_notation_after_do_tl
2312 }{
2313 \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2314 \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2315 \str_if_eq:VnTF \l_tmpa_str a {
2316 \__stex_notation_argument_assoc:n
2317 }{
2318 \str_if_eq:VnTF \l_tmpa_str B {
2319 \__stex_notation_argument_assoc:n
2320 }{
2321 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2322 \tl_put_right:Nx \l__stex_notation_dummyargs_tl {
2323 { \stex_term_math_arg:nnn
2324 { \int_use:N \l__stex_notation_currarg_int }
2325 { \l_tmpa_str }
2326 { ###\int_use:N \l__stex_notation_currarg_int }
2327 }
2328 }
2329 \__stex_notation_arguments:
2330 }
2331 }
2332 }
2333 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2334 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2335
2336 \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2337 {\l__stex_notation_arity_str}{
2338 #1
2339 }
2340 \int_zero:N \l_tmpa_int
2341 \tl_clear:N \l_tmpa_tl
2342 \str_map_inline:Nn \l__stex_notation_args_str {
2343 \int_incr:N \l_tmpa_int
2344 \tl_put_right:Nx \l_tmpa_tl {
2345 \str_if_eq:nnTF {##1}{a}{ {} }{
2346 \str_if_eq:nnTF {##1}{B}{ {} }{
2347 {##### \int_use:N \l_tmpa_int}

```

```

2348     }
2349   }
2350 }
2351 }
2352 \exp_after:wN\exp_after:wN\exp_after:wN \def
2353 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2354 \exp_after:wN\exp_after:wN\exp_after:wN ##
2355 \exp_after:wN\exp_after:wN\exp_after:wN 1
2356 \exp_after:wN\exp_after:wN\exp_after:wN ##
2357 \exp_after:wN\exp_after:wN\exp_after:wN 2
2358 \exp_after:wN\exp_after:wN\exp_after:wN {
2359   \exp_after:wN \exp_after:wN \exp_after:wN
2360   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2361     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2362   }
2363 }
2364
2365 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2366 \tl_put_right:Nx \l__stex_notation_dummyargs_tl { {
2367   \stex_term_math_assoc_arg:nnnn
2368   { \int_use:N \l__stex_notation_currarg_int }
2369   { \l_tmpa_str }
2370   { ####\int_use:N \l__stex_notation_currarg_int }
2371   { \l_tmpa_cs {####1} {####2} }
2372 } }
2373 %\cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2374 %\tl_put_right:Nx \l_tmpa_tl {
2375 % { \stex_term_math_assoc_arg:nnnn
2376 %   { \int_use:N \l_tmpa_int }
2377 %   { \l_tmpb_str }
2378 %   \exp_args:No \exp_not:n
2379 %   {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2380 %   { ####\int_use:N \l_tmpa_int }
2381 % }
2382 %}
2383 \__stex_notation_arguments:
2384 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2385 \cs_new_protected:Nn \__stex_notation_final: {
2386   \exp_args:Nne \use:nn
2387   {
2388     \cs_generate_from_arg_count:cNnn {
2389       stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2390       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2391       _cs
2392     }
2393     \cs_set:Npn \l__stex_notation_arity_str } { {
2394       \exp_after:wN \exp_after:wN \exp_after:wN
2395       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2396       { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2397     } }

```

```

2398
2399 \tl_if_empty:NF \l__stex_notation_op_tl {
2400   \cs_set:cpx {
2401     stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2402     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2403     _cs
2404   } {
2405     \stex_term_oms:nnn {
2406       \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2407       \l__stex_notation_lang_str
2408     }{
2409       \l__stex_notation_symbol_str
2410     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2411   }
2412 }
2413
2414 \exp_args:Ne
2415 \stex_add_to_current_module:n {
2416   \cs_generate_from_arg_count:cNnn {
2417     stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2418     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2419     _cs
2420   } \cs_set:Npn {\l__stex_notation_arity_str} {
2421     \exp_after:wN \exp_after:wN \exp_after:wN
2422     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2423     { \exp_after:wN \l__stex_notation_macrocode_cs \l__stex_notation_dummyargs_tl }
2424   }
2425   \tl_if_empty:NF \l__stex_notation_op_tl {
2426     \cs_set:cpn {
2427       stex_op_notation_ \l__stex_notation_symbol_str \c_hash_str
2428       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2429       _cs
2430     } {
2431       \stex_term_oms:nnn {
2432         \l__stex_notation_symbol_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2433         \l__stex_notation_lang_str
2434       }{
2435         \l__stex_notation_symbol_str
2436       }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2437     }
2438   }
2439 }
2440 \exp_args:Nx
2441 % \stex_do_up_to_module:n {
2442   \seq_put_right:cx {
2443     l_stex_symdecl_ \l__stex_notation_symbol_str
2444     _notations
2445   } {
2446     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2447   }
2448 % }
2449
2450 \stex_debug:nn{symbols}{
2451   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str

```

```

2452 ~for~\l__stex_notation_symbol_str^^J
2453 Operator~precedence:~\l__stex_notation_opprec_tl^^J
2454 Argument~precedences:~
2455   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2456 Notation: \cs_meaning:c {
2457   stex_notation_ \l__stex_notation_symbol_str \c_hash_str
2458   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2459   _cs
2460 }
2461 }
2462
2463 %\prop_set_eq:cN {
2464 %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2465 %   \c_hash_str \l__stex_notation_lang_str _prop
2466 %} \l_tmpb_prop
2467
2468 \exp_args:Ne
2469 \stex_add_to_current_module:n {
2470   \seq_put_right:cn {
2471     l_stex_symdecl_ \l__stex_notation_symbol_str
2472     _notations
2473   } {
2474     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2475   }
2476   %\prop_set_from_keyval:cn {
2477   %   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2478   %   \c_hash_str \l__stex_notation_lang_str _prop
2479   %} {
2480   %   symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2481   %   language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2482   %   variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2483   %   opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2484   %   argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2485   %}
2486 }
2487
2488 \stex_if_smsmode:F {
2489
2490   % HTML annotations
2491   \stex_if_do_html:T {
2492     \stex_annotate_invisible:nnn { notation }
2493     { \l__stex_notation_symbol_str } {
2494       \stex_annotate_invisible:nnn { notationfragment }
2495       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2496       \stex_annotate_invisible:nnn { precedence }
2497       { \l__stex_notation_prec_str }{}
2498
2499       \int_zero:N \l_tmpa_int
2500       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2501       \tl_clear:N \l_tmpa_tl
2502       \int_step_inline:nn { \l__stex_notation_arity_str }{
2503         \int_incr:N \l_tmpa_int
2504         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2505         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_

```



```

2506 \str_if_eq:VnTF \l_tmpb_str a {
2507 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2508 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2509 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2510 } }
2511 }{
2512 \str_if_eq:VnTF \l_tmpb_str B {
2513 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2514 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2515 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2516 } }
2517 }{
2518 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2519 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2520 } }
2521 }
2522 }
2523 }
2524 \stex_annotate_invisible:nnn { notationcomp }{}{
2525 \str_set:Nx \l_stex_current_symbol_str { \l__stex_notation_symbol_str }
2526 $ \exp_args:Nno \use:nn { \use:c {
2527 stex_notation_ \l_stex_current_symbol_str
2528 \c_hash_str \l__stex_notation_variant_str
2529 \c_hash_str \l__stex_notation_lang_str _cs
2530 } } { \l_tmpa_tl } $
2531 }
2532 }
2533 }
2534 }
2535 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2536 \keys_define:nn { stex / setnotation } {
2537 lang .tl_set_x:N = \l__stex_notation_lang_str ,
2538 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2539 unknown .code:n = \str_set:Nx
2540 \l__stex_notation_variant_str \l_keys_key_str
2541 }
2542
2543 \cs_new_protected:Nn \stex_setnotation_args:n {
2544 \str_clear:N \l__stex_notation_lang_str
2545 \str_clear:N \l__stex_notation_variant_str
2546 \keys_set:nn { stex / setnotation } { #1 }
2547 }
2548
2549 \cs_new_protected:Nn \stex_setnotation:n {
2550 \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2551 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2552 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2553 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2554 \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2555 { \c_hash_str }

```

```

2556 \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2557 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2558 \exp_args:Nx \stex_add_to_current_module:n {
2559   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2560   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2561   \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2562   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2563   \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2564   { \c_hash_str }
2565 }
2566 \stex_debug:nn {notations}{
2567   Setting~default~notation~
2568   {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2569   #1 \
2570   \expandafter\meaning\csname
2571   l_stex_symdecl_#1 _notations\endcsname
2572 }
2573 }{
2574   % todo throw error
2575 }
2576 }
2577
2578 \NewDocumentCommand \setnotation {m m} {
2579   \stex_get_symbol:n { #1 }
2580   \_stex_setnotation_args:n { #2 }
2581   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2582   \stex_smsmode_do:
2583 }
2584
2585 \cs_new_protected:Nn \stex_copy_notations:nn {
2586   \stex_debug:nn {notations}{
2587     Copying~notations~from~#2~to~#1\
2588     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2589   }
2590   \tl_clear:N \l_tmpa_tl
2591   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2592     \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2593   }
2594   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2595     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2596     \edef \l_tmpa_tl {
2597       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2598       \exp_after:wN\exp_after:wN\exp_after:wN {
2599         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2600       }
2601     }
2602     \exp_args:Nx
2603     \stex_do_up_to_module:n {
2604       \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2605       \cs_generate_from_arg_count:cNnn {
2606         stex_notation_ #1 \c_hash_str ##1 _cs
2607       } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2608         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2609       }

```

```

2610     }
2611   }
2612 }
2613
2614 \NewDocumentCommand \copynotation {m m} {
2615   \stex_get_symbol:n { #1 }
2616   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2617   \stex_get_symbol:n { #2 }
2618   \exp_args:Noo
2619   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2620   \exp_args:Nx \stex_add_import_to_current_module:n{
2621     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2622   }
2623   \stex_smsmode_do:
2624 }
2625

```

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

\symdef

```

2626 \keys_define:nn { stex / symdef } {
2627   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2628   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2629   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2630   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2631   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2632   op        .tl_set:N    = \l__stex_notation_op_tl ,
2633   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2634   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2635   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2636   assoc     .choices:nn =
2637     {bin,binl,binr,pre,conj,pwconj}
2638     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2639   unknown   .code:n      = \str_set:Nx
2640     \l__stex_notation_variant_str \l_keys_key_str
2641 }
2642
2643 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2644   \str_clear:N \l_stex_symdecl_name_str
2645   \str_clear:N \l_stex_symdecl_args_str
2646   \str_clear:N \l_stex_symdecl_assoctype_str
2647   \bool_set_false:N \l_stex_symdecl_local_bool
2648   \tl_clear:N \l_stex_symdecl_type_tl
2649   \tl_clear:N \l_stex_symdecl_definiens_tl
2650   \str_clear:N \l__stex_notation_lang_str
2651   \str_clear:N \l__stex_notation_variant_str
2652   \str_clear:N \l__stex_notation_prec_str
2653   \tl_clear:N \l__stex_notation_op_tl
2654
2655   \keys_set:nn { stex / symdef } { #1 }
2656 }
2657
2658 \NewDocumentCommand \symdef { O{} m } {
2659   \__stex_notation_symdef_args:n { #1 }

```

```

2660 \bool_set_true:N \l_stex_symdecl_make_macro_bool
2661 \stex_symdecl_do:n { #2 }
2662 \tl_set:Nn \l__stex_notation_after_do_tl {
2663   \__stex_notation_final:
2664   \stex_smsmode_do:
2665 }
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.)

30.3 Variables

```

2671 <@@=stex_variables>
2672
2673 \keys_define:nn { stex / vardef } {
2674   name .str_set_x:N = \l__stex_variables_name_str ,
2675   args .str_set_x:N = \l__stex_variables_args_str ,
2676   type .tl_set:N    = \l__stex_variables_type_tl ,
2677   def .tl_set:N     = \l__stex_variables_def_tl ,
2678   op .tl_set:N      = \l__stex_variables_op_tl ,
2679   prec .str_set_x:N = \l__stex_variables_prec_str ,
2680   assoc .choices:nn =
2681     {bin,binl,binr,pre,conj,pwconj}
2682     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2683   bind .choices:nn =
2684     {forall,exists}
2685     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2686 }
2687
2688 \cs_new_protected:Nn \__stex_variables_args:n {
2689   \str_clear:N \l__stex_variables_name_str
2690   \str_clear:N \l__stex_variables_args_str
2691   \str_clear:N \l__stex_variables_prec_str
2692   \str_clear:N \l__stex_variables_assoctype_str
2693   \str_clear:N \l__stex_variables_bind_str
2694   \tl_clear:N \l__stex_variables_type_tl
2695   \tl_clear:N \l__stex_variables_def_tl
2696   \tl_clear:N \l__stex_variables_op_tl
2697
2698   \keys_set:nn { stex / vardef } { #1 }
2699 }
2700
2701 \NewDocumentCommand \vardecl {0{} m m} {
2702   \__stex_variables_args:n {#1}
2703   \str_if_empty:NT \l__stex_variables_name_str {
2704     \str_set:Nx \l__stex_variables_name_str { #2 }
2705   }
2706   \prop_clear:N \l_tmpa_prop
2707   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2708

```

```

2709 \int_zero:N \l_tmpb_int
2710 \bool_set_true:N \l_tmpa_bool
2711 \str_map_inline:Nn \l__stex_variables_args_str {
2712   \token_case_meaning:NnF ##1 {
2713     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2714     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2715     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2716     {\tl_to_str:n a} {
2717       \bool_set_false:N \l_tmpa_bool
2718       \int_incr:N \l_tmpb_int
2719     }
2720     {\tl_to_str:n B} {
2721       \bool_set_false:N \l_tmpa_bool
2722       \int_incr:N \l_tmpb_int
2723     }
2724   }{
2725     \msg_error:nxxx{stex}{error/wrongargs}{
2726       variable~\l__stex_variables_name_str
2727     }{##1}
2728   }
2729 }
2730 \bool_if:NTF \l_tmpa_bool {
2731   % possibly numeric
2732   \str_if_empty:NTF \l__stex_variables_args_str {
2733     \prop_put:Nnn \l_tmpa_prop { args } {}
2734     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2735   }{
2736     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2737     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2738     \str_clear:N \l_tmpa_str
2739     \int_step_inline:nn \l_tmpa_int {
2740       \str_put_right:Nn \l_tmpa_str i
2741     }
2742     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2743   }
2744 } {
2745   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2746   \prop_put:Nnx \l_tmpa_prop { arity }
2747     { \str_count:N \l__stex_variables_args_str }
2748 }
2749 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2750 \tl_set:cn { #2 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2751
2752
2753
2754
2755
2756
2757 \prop_set_eq:cn { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2758 }
2759
2760
2761
2762

```

2763 `</package>`

Chapter 31

STEX -Terms Implementation

```
2764 <*package>
2765
2766 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2767
2768 <@@=stex_terms>
2769
2770 Warnings and error messages
2771 \msg_new:nnn{stex}{error/nonotation}{
2772   Symbol~#1~invoked,~but~has~no~notation~#2!
2773 }
2774 \msg_new:nnn{stex}{error/notationarg}{
2775   Error~in~parsing~notation~#1
2776 }
2777 \msg_new:nnn{stex}{error/noop}{
2778   Symbol~#1~has~no~operator~notation~for~notation~#2
2779 }
```

31.1 Symbol Invocations

Arguments:

```
2779 \keys_define:nn { stex / terms } {
2780   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2781   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2782   unknown .code:n = \str_set:Nx
2783     \l__stex_terms_variant_str \l_keys_key_str
2784 }
2785
2786 \cs_new_protected:Nn \__stex_terms_args:n {
2787   \str_clear:N \l__stex_terms_lang_str
2788   \str_clear:N \l__stex_terms_variant_str
2789   \str_clear:N \l__stex_terms_prec_str
2790   \tl_clear:N \l__stex_terms_op_tl
2791 }
2792 \keys_set:nn { stex / terms } { #1 }
```

2793 }

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2794 \cs_new_protected:Nn \stex_invoke_symbol:n {
2795   \str_if_eq:eeF {
2796     \prop_item:cn {
2797       l_stex_symdecl_#1_prop
2798     }{ deprecate }
2799   }{}{
2800     \msg_warning:nxxx{stex}{warning/deprecated}{
2801       Symbol~#1
2802     }{
2803       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2804     }
2805   }
2806   \if_mode_math:
2807     \exp_after:wN \__stex_terms_invoke_math:n
2808   \else:
2809     \exp_after:wN \__stex_terms_invoke_text:n
2810   \fi: { #1 }
2811 }
```

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

`__stex_terms_invoke_math:n`

```

2812 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2813   \peek_charcode_remove:NTF ! {
2814     \peek_charcode:NTF [ {
2815       \__stex_terms_invoke_op:nw { #1 }
2816     }{
2817       \peek_charcode_remove:NTF ! {
2818         \peek_charcode:NTF [ {
2819           \__stex_terms_invoke_op_custom:nw
2820         }{
2821           % TODO throw error
2822         }
2823       }{
2824         \__stex_terms_invoke_op:nw { #1 } []
2825       }
2826     }
2827   }{
2828     \peek_charcode_remove:NTF * {
2829       \__stex_terms_invoke_text:n { #1 }
2830     }{
2831       \peek_charcode:NTF [ {
2832         \__stex_terms_invoke_math:nw { #1 }
2833       }{
2834         \__stex_terms_invoke_math:nw { #1 } []
2835       }
2836     }
2837   }
2838 }
```

(End definition for `__stex_terms_invoke_math:n`.)

_stex_terms_invoke_op_custom:nw

```

2839 \cs_new_protected:Npn \_stex_terms_invoke_op_custom:nw #1 [#2] {
2840   \_stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2841     \stex_highlight_term:nn{#1}{#2}
2842   }
2843 }

```

(End definition for _stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2844 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2845   \_stex_terms_args:n { #2 }
2846   \cs_if_exist:cTF {
2847     stex_op_notation_ #1 \c_hash_str
2848     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2849   }{
2850     \csname stex_op_notation_ #1 \c_hash_str
2851       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2852     \endcsname
2853   }{
2854     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_t
2855   }
2856 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2857 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2858   \_stex_terms_args:n { #2 }
2859   \seq_if_empty:cTF {
2860     l_stex_symdecl_ #1 _notations
2861   } {
2862     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2863   } {
2864     \seq_if_in:cxTF {
2865       l_stex_symdecl_ #1 _notations
2866     }
2867     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2868       \str_set:Nn \l_stex_current_symbol_str { #1 }
2869       \stex_debug:nn{terms}{Using~
2870         #1\c_hash_str\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str \
2871       \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2872       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2873       _cs\endcsname
2874     }
2875     \use:c{
2876       stex_notation_ #1 \c_hash_str
2877       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2878       _cs
2879     }
2880   }{
2881     \str_if_empty:NTF \l__stex_terms_variant_str {
2882       \str_if_empty:NTF \l__stex_terms_lang_str {
2883         \seq_get_left:cN {

```

```

2884         l_stex_symdecl_ #1 _notations
2885     } \l_tmpa_str
2886     \str_set:Nn \l_stex_current_symbol_str { #1 }
2887     \stex_debug:nn{terms}{Using~
2888         #1\c_hash_str\l_tmpa_str \
2889         \expandafter\meaning\csname stex_notation_ #1 \c_hash_str
2890         \l_tmpa_str
2891         _cs\endcsname
2892     }
2893     \use:c{
2894         stex_notation_ #1 \c_hash_str \l_tmpa_str
2895         _cs
2896     }
2897   }{
2898     \msg_error:nnxx{stex}{error/nonotation}{#1}{
2899       ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2900     }
2901   }
2902 }{
2903   \msg_error:nnxx{stex}{error/nonotation}{#1}{
2904     ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2905   }
2906 }
2907 }
2908 }
2909 }

```

(End definition for _stex_terms_invoke_math:nw.)

_stex_terms_invoke_text:n

```

2910 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2911   \peek_charcode_remove:NTF ! {
2912     \stex_term_custom:nn { #1 } { }
2913   }{
2914     \prop_set_eq:Nc \l_tmpa_prop {
2915       l_stex_symdecl_ #1 _prop
2916     }
2917     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2918     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2919   }
2920 }

```

(End definition for _stex_terms_invoke_text:n.)

31.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2921 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2922 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2923 \int_new:N \l__stex_terms_downprec
2924 \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`

```
2925 \tl_set:Nn \l__stex_terms_left_bracket_str (
2926 \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

```
2927 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2928   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2929     \bool_set_false:N \l__stex_terms_brackets_done_bool
2930     #2
2931   } {
2932     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2933       \bool_if:NTF \l__stex_inarray_bool { #2 } {
2934         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2935         \dobrackets { #2 }
2936       }
2937     }{ #2 }
2938   }
2939 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```
2940 \bool_new:N \l__stex_terms_brackets_done_bool
2941 %\RequirePackage{scalerel}
2942 \cs_new_protected:Npn \dobrackets #1 {
2943   %\ThisStyle{\if D\m@switch
2944   %   \exp_args:Nnx \use:nn
2945   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2946   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2947   % \else
2948   \exp_args:Nnx \use:nn
2949   {
2950     \bool_set_true:N \l__stex_terms_brackets_done_bool
2951     \int_set:Nn \l__stex_terms_downprec \infprec
2952     \l__stex_terms_left_bracket_str
2953     #1
2954   }
2955   {
2956     \bool_set_false:N \l__stex_terms_brackets_done_bool
2957     \l__stex_terms_right_bracket_str
2958     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2959   }
2960   %\fi}
2961 }
```

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

\withbrackets

```
2962 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2963   \exp_args:Nnx \use:nn
2964   {
2965     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2966     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2967     #3
2968   }
2969   {
2970     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2971       {\l__stex_terms_left_bracket_str}
2972     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2973       {\l__stex_terms_right_bracket_str}
2974   }
2975 }
```

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

\STEXinvisible

```
2976 \cs_new_protected:Npn \STEXinvisible #1 {
2977   \stex_annotate_invisible:n { #1 }
2978 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
2979 \cs_new_protected:Nn \_stex_term_oms:nnn {
2980   \stex_annotate:nnn{ OMID }{ #2 }{
2981     \stex_highlight_term:nn { #1 } { #3 }
2982   }
2983 }
2984
2985 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2986   \_stex_terms_maybe_brackets:nn { #3 }{
2987     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2988   }
2989 }
```

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

_stex_term_math_oma:nnnn

```
2990 \cs_new_protected:Nn \_stex_term_oma:nnn {
2991   \stex_annotate:nnn{ OMA }{ #2 }{
2992     \stex_highlight_term:nn { #1 } { #3 }
2993   }
2994 }
2995
2996 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2997   \_stex_terms_maybe_brackets:nn { #3 }{
2998     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2999   }
3000 }
```

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

`_stex_term_math_omb:nnnn`

```

3001 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3002   \stex_annotate:nnn{ OMBIND }{ #2 }{
3003     \stex_highlight_term:nn { #1 } { #3 }
3004   }
3005 }
3006
3007 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3008   \__stex_terms_maybe_brackets:nn { #3 }{
3009     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3010   }
3011 }

```

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

`_stex_term_math_arg:nnn`

```

3012 \cs_new_protected:Nn \_stex_term_arg:nn {
3013   \stex_unhighlight_term:n {
3014     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3015   }
3016 }
3017 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3018   \exp_args:Nnx \use:nn
3019   { \int_set:Nn \l__stex_terms_downprec { #2 }
3020     \_stex_term_arg:nn { #1 }{ #3 }
3021   }
3022   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3023 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

3024 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3025   % TODO sequences
3026   \clist_set:Nn \l_tmpa_clist{ #3 }
3027   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3028     \tl_set:Nn \l_tmpa_tl { #3 }
3029   }{
3030     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3031     \clist_reverse:N \l_tmpa_clist
3032     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3033
3034     \clist_map_inline:Nn \l_tmpa_clist {
3035       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3036         \exp_args:Nno
3037         \l_tmpa_cs { ##1 } \l_tmpa_tl
3038       }
3039     }
3040   }
3041   \exp_args:Nnno
3042   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
3043 }

```

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

`\stex_term_custom:nn`

```

3044 \cs_new_protected:Nn \stex_term_custom:nn {
3045   \str_set:Nn \l__stex_terms_custom_uri { #1 }
3046   \str_set:Nn \l_tmpa_str { #2 }
3047   \tl_clear:N \l_tmpa_tl
3048   \int_zero:N \l_tmpa_int
3049   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
3050   \__stex_terms_custom_loop:
3051 }

```

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

`__stex_terms_custom_loop:`

```

3052 \cs_new_protected:Nn \__stex_terms_custom_loop: {
3053   \bool_set_false:N \l_tmpa_bool
3054   \bool_while_do:nn {
3055     \str_if_eq_p:ee X {
3056       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
3057     }
3058   }{
3059     \int_incr:N \l_tmpa_int
3060   }
3061
3062   \peek_charcode:NTF [ {
3063     % notation/text component
3064     \__stex_terms_custom_component:w
3065   } {
3066     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
3067       % all arguments read => finish
3068       \__stex_terms_custom_final:
3069     } {
3070       % arguments missing
3071       \peek_charcode_remove:NTF * {
3072         % invisible, specific argument position or both
3073         \peek_charcode:NTF [ {
3074           % visible specific argument position
3075           \__stex_terms_custom_arg:wn
3076         } {
3077           % invisible
3078           \peek_charcode_remove:NTF * {
3079             % invisible specific argument position
3080             \__stex_terms_custom_arg_inv:wn
3081           } {
3082             % invisible next argument
3083             \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
3084           }
3085         }
3086       } {
3087         % next normal argument
3088         \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
3089       }
3090     }
3091   }
3092 }

```

(End definition for _stex_terms_custom_loop:.)

_stex_terms_custom_arg_inv:wn

```

3093 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
3094   \bool_set_true:N \l_tmpa_bool
3095   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
3096 }

```

(End definition for _stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```

3097 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
3098   \str_set:Nx \l_tmpb_str {
3099     \str_item:Nn \l_tmpa_str { #1 }
3100   }
3101   \str_case:VnTF \l_tmpb_str {
3102     { X } {
3103       \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3104     }
3105     { i } { \_stex_terms_custom_set_X:n { #1 } }
3106     { b } { \_stex_terms_custom_set_X:n { #1 } }
3107     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3108     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
3109   }{}{
3110     \msg_error:nnx{stex}{error/notationarg}{\l_stex_terms_custom_uri}
3111   }
3112
3113   \bool_if:nTF \l_tmpa_bool {
3114     \tl_put_right:Nx \l_tmpa_tl {
3115       \stex_annotate_invisible:n {
3116         \_stex_term_arg:nn { \int_eval:n { #1 } }
3117         \exp_not:n { { #2 } }
3118       }
3119     }
3120   } {
3121     \tl_put_right:Nx \l_tmpa_tl {
3122       \_stex_term_arg:nn { \int_eval:n { #1 } }
3123       \exp_not:n { { #2 } }
3124     }
3125   }
3126
3127   \_stex_terms_custom_loop:
3128 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

3129 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
3130   \str_set:Nx \l_tmpa_str {
3131     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
3132     X
3133     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
3134   }
3135 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

3136 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
3137   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
3138   \_stex_terms_custom_loop:
3139 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

3140 \cs_new_protected:Nn \_stex_terms_custom_final: {
3141   \int_compare:nNnTF \l_tmpb_int = 0 {
3142     \exp_args:Nnno \_stex_term_oms:nnn
3143   }{
3144     \str_if_in:NnTF \l_tmpa_str {b} {
3145       \exp_args:Nnno \_stex_term_ombind:nnn
3146     } {
3147       \exp_args:Nnno \_stex_term_oma:nnn
3148     }
3149   }
3150   { \l_stex_terms_custom_uri } { \l_stex_terms_custom_uri } { \l_tmpa_tl }
3151 }

```

(End definition for _stex_terms_custom_final:.)

\symref
\symname

```

3152 \NewDocumentCommand \symref { m m }{
3153   \let\compemph_uri_prev:\compemph@uri
3154   \let\compemph@uri\symrefemph@uri
3155   \STEXsymbol{#1}!{#2}
3156   \let\compemph@uri\compemph_uri_prev:
3157 }
3158
3159 \keys_define:nn { stex / symname } {
3160   post .str_set_x:N = \l_stex_symname_post_str
3161 }
3162
3163 \cs_new_protected:Nn \stex_symname_args:n {
3164   \str_clear:N \l_stex_symname_post_str
3165   \keys_set:nn { stex / symname } { #1 }
3166 }
3167
3168 \NewDocumentCommand \symname { 0{ } m }{
3169   \stex_symname_args:n { #1 }
3170   \stex_get_symbol:n { #2 }
3171   \str_set:Nx \l_tmpa_str {
3172     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3173   }
3174   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3175
3176   \let\compemph_uri_prev:\compemph@uri
3177   \let\compemph@uri\symrefemph@uri
3178   \exp_args:NNx \use:nn

```



```

3179 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
3180 \l_tmpa_str \l_stex_symname_post_str
3181 ] }
3182 \let\compemph@uri\compemph_uri_prev:
3183 }

```

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

31.3 Notation Components

```

3184 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3185
3186 \str_new:N \l_stex_current_symbol_str
3187 \cs_new_protected:Nn \stex_highlight_term:nn {
3188   \exp_args:Nnx
3189   \use:nn {
3190     \str_set:Nx \l_stex_current_symbol_str { #1 }
3191     #2
3192   } {
3193     \str_set:Nx \exp_not:N \l_stex_current_symbol_str
3194     { \l_stex_current_symbol_str }
3195   }
3196 }
3197
3198 \cs_new_protected:Nn \stex_unhighlight_term:n {
3199   % \latexml_if:TF {
3200   %   #1
3201   % } {
3202   %   \rustex_if:TF {
3203   %     #1
3204   %   } {
3205     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3206   % }
3207   % }
3208 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
3209 \cs_new_protected:Npn \comp #1 {
3210   \str_if_empty:NF \l_stex_current_symbol_str {
3211     \rustex_if:TF {
3212       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3213     }{
3214       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3215     }
3216   }
3217 }
3218
3219 \cs_new_protected:Npn \compemph@uri #1 #2 {
3220   \compemph{ #1 }
3221 }

```

```

3222
3223
3224 \cs_new_protected:Npn \compemph #1 {
3225     #1
3226 }
3227
3228 \cs_new_protected:Npn \defemph@uri #1 #2 {
3229     \defemph{#1}
3230 }
3231
3232 \cs_new_protected:Npn \defemph #1 {
3233     \textbf{#1}
3234 }
3235
3236 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3237     \symrefemph{#1}
3238 }
3239
3240 \cs_new_protected:Npn \symrefemph #1 {
3241     \textbf{#1}
3242 }

```

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

\ellipses

```

3243 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3244 \bool_new:N \l_stex_inarray_bool
3245 \bool_set_false:N \l_stex_inarray_bool
3246 \NewDocumentCommand \parray { m m } {
3247     \begin{group}
3248     \bool_set_true:N \l_stex_inarray_bool
3249     \begin{array}{#1}
3250         #2
3251     \end{array}
3252     \end{group}
3253 }
3254
3255 \NewDocumentCommand \prmatrix { m } {
3256     \begin{group}
3257     \bool_set_true:N \l_stex_inarray_bool
3258     \begin{matrix}
3259         #1
3260     \end{matrix}
3261     \end{group}
3262 }
3263
3264 \def \maybepline {
3265     \bool_if:NT \l_stex_inarray_bool {\hline}
3266 }
3267
3268 \def \parrayline #1 #2 {

```

```

3269   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}\}
3270 }
3271
3272 \def \pmrow #1 { \parrayline{ }{ #1 } }
3273
3274 \def \parraylineh #1 #2 {
3275   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}\hline}
3276 }
3277
3278 \def \parraycell #1 {
3279   #1 \bool_if:NT \l_stex_inarray_bool {&}
3280 }

```

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

```

3281 \endpackage

```

Chapter 32

STEX -Structural Features Implementation

```
3282 <*package>
3283
3284 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3285
3286 <@@=stex_features>
3287
3288   Warnings and error messages
3289   \msg_new:nnn{stex}{error/copymodule/notallowed}{
3290     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3291   }
3292   \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3293     Symbol~#1~not~assigned~in~interpretmodule~#2
3294   }
3295
```

32.1 Imports with modification

```
3294 \cs_new_protected:Nn \stex_get_symbol_in_copymodule:n {
3295   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3296     \__stex_features_get_symbol_from_cs:n { #1 }
3297   }{
3298     % argument is a string
3299     % is it a command name?
3300     \cs_if_exist:cTF { #1 }{
3301       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3302       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3303       \str_if_empty:NNTF \l_tmpa_str {
3304         \exp_args:Nx \cs_if_eq:NNTF {
3305           \tl_head:N \l_tmpa_tl
3306         } \stex_invoke_symbol:n {
3307           \exp_args:No \__stex_features_get_symbol_from_cs:n { \use:c { #1 } }
3308         }{
3309           \__stex_features_get_symbol_from_string:n { #1 }
3310         }
3311       }
3312     }
3313   }
3314 }
```

```

3310     }
3311   } {
3312     \__stex_features_get_symbol_from_string:n { #1 }
3313   }
3314   ){
3315     % argument is not a command name
3316     \__stex_features_get_symbol_from_string:n { #1 }
3317     % \l_stex_all_symbols_seq
3318   }
3319 }
3320 }
3321
3322 \cs_new_protected:Nn \__stex_features_get_symbol_from_string:n {
3323   \str_set:Nn \l_tmpa_str { #1 }
3324   \bool_set_false:N \l_tmpa_bool
3325   \bool_if:NF \l_tmpa_bool {
3326     \tl_set:Nn \l_tmpa_tl {
3327       \msg_set:nnn{stex}{error/unknownsymbol}{
3328         No~symbol~#1~found!
3329       }
3330       \msg_error:nn{stex}{error/unknownsymbol}
3331     }
3332     \str_set:Nn \l_tmpa_str { #1 }
3333     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3334     \seq_map_inline:Nn \l__stex_features_copymodule_fields_seq {
3335       \str_set:Nn \l_tmpb_str { ##1 }
3336       \str_if_eq:eeT { \l_tmpa_str } {
3337         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3338       } {
3339         \seq_map_break:n {
3340           \tl_set:Nn \l_tmpa_tl {
3341             \str_set:Nn \l_stex_get_symbol_uri_str {
3342               ##1
3343             }
3344             \__stex_features_get_symbol_check:
3345           }
3346         }
3347       }
3348     }
3349     \l_tmpa_tl
3350   }
3351 }
3352
3353 \cs_new_protected:Nn \__stex_features_get_symbol_from_cs:n {
3354   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3355   { \tl_tail:N \l_tmpa_tl }
3356   \tl_if_single:NTF \l_tmpa_tl {
3357     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3358       \exp_after:wN \str_set:Nn \exp_after:wN
3359       \l_stex_get_symbol_uri_str \l_tmpa_tl
3360       \__stex_features_get_symbol_check:
3361     }{
3362       % TODO
3363       % tail is not a single group

```

```

3364     }
3365   }{
3366     % TODO
3367     % tail is not a single group
3368   }
3369 }
3370
3371 \cs_new_protected:Nn \__stex_features_get_symbol_check: {
3372   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq {?} \l_stex_get_symbol_uri_str
3373   \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 3 {
3374     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
3375     \str_set:Nx \l_tmpa_str {\seq_use:Nn \l_tmpa_seq ?}
3376     \seq_if_in:NoF \l__stex_features_copymodule_modules_seq \l_tmpa_str {
3377       \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3378         \l_stex_current_copymodule_name_str\Allowed:~\seq_use:Nn \l__stex_features_copymodule_modules_seq \l_tmpa_str
3379       }
3380     }
3381   }{
3382     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3383       \l_stex_current_copymodule_name_str~(inexplicably)
3384     }
3385   }
3386 }
3387
3388 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3389   \stex_import_module_uri:nn { #1 } { #2 }
3390   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3391   \stex_import_require_module:nnnn
3392     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3393     { \l_stex_import_path_str } { \l_stex_import_name_str }
3394   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3395   \seq_set_eq:NN \l__stex_features_copymodule_modules_seq \l_stex_collect_imports_seq
3396   \seq_clear:N \l__stex_features_copymodule_fields_seq
3397   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3398     \seq_map_inline:cn {c_stex_module_###1_constants}{
3399       \exp_args:NNx \seq_put_right:Nn \l__stex_features_copymodule_fields_seq {
3400         ###1 ? #####1
3401       }
3402     }
3403   }
3404   \seq_clear:N \l_tmpa_seq
3405   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3406     name      = \l_stex_current_copymodule_name_str ,
3407     module    = \l_stex_current_module_str ,
3408     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3409     includes  = \l_tmpa_seq ,
3410     fields    = \l_tmpa_seq
3411   }
3412   \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3413     as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3414   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_features_copymodule_modules_seq
3415     \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_features_copymodule_fields_seq {,~}
3416   \stex_if_smsmode:F {
3417     \begin{stex_annotate_env} {#4} {

```

```

3418     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3419   }
3420   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{\}
3421 }
3422 \bool_set_eq:NN \l__stex_features_oldhtml_bool \stex_html_do_output_bool
3423 \bool_set_false:N \stex_html_do_output_bool
3424 }
3425 \cs_new_protected:Nn \stex_copymodule_end:n {
3426   \def \l_tmpa_cs ##1 ##2 {#1}
3427   \bool_set_eq:NN \stex_html_do_output_bool \l__stex_features_oldhtml_bool
3428   \tl_clear:N \l_tmpa_tl
3429   \tl_clear:N \l_tmpb_tl
3430   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3431   \seq_map_inline:Nn \l__stex_features_copymodule_modules_seq {
3432     \seq_map_inline:cn {c_stex_module_##1_constants}{
3433       \tl_clear:N \l_tmpc_tl
3434       \l_tmpa_cs{##1}{####1}
3435       \str_if_exist:cTF {\l__stex_features_copymodule_##1?####1_name_str} {
3436         \tl_put_right:Nx \l_tmpa_tl {
3437           \prop_set_from_keyval:cn {
3438             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3439           }{
3440             \exp_after:wN \prop_to_keyval:N \csname
3441               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_features_copymodule_#
3442             \endcsname
3443           }
3444           \seq_clear:c {
3445             l_stex_symdecl_
3446             \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_name_s
3447             _notations
3448           }
3449         }
3450         \tl_put_right:Nx \l_tmpc_tl {
3451           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_features_cop
3452           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_features_copymodule_##1?####1_n
3453         }
3454         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_features_
3455         \str_if_exist:cT {\l_stex_features_copymodule_##1?####1_macroname_str} {
3456           \tl_put_right:Nx \l_tmpc_tl {
3457             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3458           }
3459           \tl_put_right:Nx \l_tmpa_tl {
3460             \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{\
3461             \stex_invoke_symbol:n {
3462               \l_stex_current_module_str ? \use:c{l__stex_features_copymodule_##1?####1_na
3463             }
3464           }
3465         }
3466       }
3467     }{
3468       \tl_put_right:Nx \l_tmpc_tl {
3469         \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_n
3470       }
3471       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}

```

```

3472 \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3473 \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3474 \tl_put_right:Nx \l_tmpa_tl {
3475   \prop_set_from_keyval:cn {
3476     l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3477   }{
3478     \prop_to_keyval:N \l_tmpa_prop
3479   }
3480   \seq_clear:c {
3481     l_stex_symdecl_
3482     \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3483     _notations
3484   }
3485 }
3486 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule
3487 \str_if_exist:cT {l__stex_features_copymodule_##1?####1_macroname_str} {
3488   \tl_put_right:Nx \l_tmpc_tl {
3489     \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_features_copymodule_##1?#
3490   }
3491   \tl_put_right:Nx \l_tmpa_tl {
3492     \tl_set:cx {\use:c{l__stex_features_copymodule_##1?####1_macroname_str}}{
3493       \stex_invoke_symbol:n {
3494         \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3495       }
3496     }
3497   }
3498 }
3499 }
3500 \tl_if_exist:cT {l__stex_features_copymodule_##1?####1_def_tl}{
3501   \tl_put_right:Nx \l_tmpc_tl {
3502     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_features_copymodule_##1?#
3503   }
3504 }
3505 \tl_put_right:Nx \l_tmpb_tl {
3506   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3507 }
3508 }
3509 }
3510 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3511 \tl_put_left:Nx \l_tmpa_tl {
3512   \prop_set_from_keyval:cn {
3513     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
3514   }{
3515     \prop_to_keyval:N \l_stex_current_copymodule_prop
3516   }
3517 }
3518 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
3519 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3520 \exp_args:Nx \stex_do_up_to_module:n {
3521   \exp_args:No \exp_not:n \l_tmpa_tl
3522 }
3523 \l_tmpb_tl
3524 \stex_if_smsmode:F {
3525   \end{stex_annotate_env}

```



```

3526 }
3527 }
3528
3529 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3530   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3531   \stex_deactivate_macro:Nn \symdecl {module~environments}
3532   \stex_deactivate_macro:Nn \symdef {module~environments}
3533   \stex_deactivate_macro:Nn \notation {module~environments}
3534   \stex_reactivate_macro:N \assign
3535   \stex_reactivate_macro:N \renamedec1
3536   \stex_reactivate_macro:N \donotcopy
3537   \stex_smsmode_do:
3538 }{
3539   \stex_copymodule_end:n {}
3540 }
3541
3542 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3543   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3544   \stex_deactivate_macro:Nn \symdecl {module~environments}
3545   \stex_deactivate_macro:Nn \symdef {module~environments}
3546   \stex_deactivate_macro:Nn \notation {module~environments}
3547   \stex_reactivate_macro:N \assign
3548   \stex_reactivate_macro:N \renamedec1
3549   \stex_reactivate_macro:N \donotcopy
3550   \stex_smsmode_do:
3551 }{
3552   \stex_copymodule_end:n {
3553     \tl_if_exist:cF {
3554       l__stex_features_copymodule_##1?##2_def_tl
3555     }{
3556       \msg_error:nxxx{stex}{error/interpretmodule/nodedefiniens}{
3557         ##1?##2
3558       }{\l_stex_current_copymodule_name_str}
3559     }
3560   }
3561 }
3562
3563 \NewDocumentCommand \donotcopy { 0{} m}{
3564   \stex_import_module_uri:nn { #1 } { #2 }
3565   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3566   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3567     \seq_remove_all:Nn \l_stex_features_copymodule_modules_seq { ##1 }
3568     \seq_map_inline:cn {c_stex_module_##1_constants}{
3569       \seq_remove_all:Nn \l_stex_features_copymodule_fields_seq { ##1 ? #####1 }
3570       \bool_lazy_any_p:nT {
3571         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_name_str}}
3572         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_macroname_str}}
3573         { \cs_if_exist_p:c {l__stex_features_copymodule_##1?####1_def_tl}}
3574       }{
3575         % TODO throw error
3576       }
3577     }
3578   }
3579 }

```

```

3580 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3581 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3582 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3583 }
3584
3585 \NewDocumentCommand \assign { m m }{
3586   \stex_get_symbol_in_copymodule:n {#1}
3587   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3588   \tl_set:cn {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3589 }
3590
3591 \keys_define:nn { stex / renamedec1 } {
3592   name .str_set_x:N = \l_stex_renamedec1_name_str
3593 }
3594 \cs_new_protected:Nn \__stex_features_renamedec1_args:n {
3595   \str_clear:N \l_stex_renamedec1_name_str
3596
3597   \keys_set:nn { stex / renamedec1 } { #1 }
3598 }
3599
3600 \NewDocumentCommand \renamedec1 { O{} m m }{
3601   \__stex_features_renamedec1_args:n { #1 }
3602   \stex_get_symbol_in_copymodule:n {#2}
3603   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
3604   \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
3605   \str_if_empty:NTF \l_stex_renamedec1_name_str {
3606     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3607       \l_stex_get_symbol_uri_str
3608     } }
3609   } {
3610     \str_set:cx {l__stex_features_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex_r
3611     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
3612     \prop_set_eq:cc {l_stex_symdecl_
3613       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3614     _prop
3615     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
3616     \seq_set_eq:cc {l_stex_symdecl_
3617       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3618     _notations
3619     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
3620     \prop_put:cnx {l_stex_symdecl_
3621       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3622     _prop
3623     }{ name }{ \l_stex_renamedec1_name_str }
3624     \prop_put:cnx {l_stex_symdecl_
3625       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3626     _prop
3627     }{ module }{ \l_stex_current_module_str }
3628     \exp_args:NNx \seq_put_left:Nn \l__stex_features_copymodule_fields_seq {
3629       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3630     }
3631     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
3632       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
3633     } }

```

```

3634 }
3635 }
3636 %\NewDocumentCommand \notation_in_copymodules: { 0{} m } {
3637 % \_stex_notation_args:n { #1 }
3638 % \tl_clear:N \l_stex_symdecl_definiens_tl
3639 % \stex_get_symbol_in_copymodule:n { #2 }
3640 % \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
3641 % % todo
3642 %}
3643 \stex_deactivate_macro:Nn \assign {copymodules}
3644 \stex_deactivate_macro:Nn \renamedec1 {copymodules}
3645 \stex_deactivate_macro:Nn \donotcopy {copymodules}
3646
3647
3648 \seq_new:N \l_stex_implicit_morphisms_seq
3649 \NewDocumentCommand \implicitmorphism { 0{} m m }{
3650 \stex_import_module_uri:nn { #1 } { #2 }
3651 \stex_debug:nn{implicits}{
3652 Implicit~morphism:~
3653 \l_stex_module_ns_str ? \l__stex_features_name_str
3654 }
3655 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
3656 \l_stex_module_ns_str ? \l__stex_features_name_str
3657 }{
3658 \msg_error:nnn{stex}{error/conflictingmodules}{
3659 \l_stex_module_ns_str ? \l__stex_features_name_str
3660 }
3661 }
3662
3663 % TODO
3664
3665
3666
3667 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
3668 \l_stex_module_ns_str ? \l__stex_features_name_str
3669 }
3670 }
3671

```

32.2 The feature environment

structural@feature

```

3672
3673 \NewDocumentEnvironment{structural@feature}{ m m m }{
3674 \stex_if_in_module:F {
3675 \msg_set:nnn{stex}{error/nomodule}{
3676 Structural~Feature~has~to~occur~in~a~module:\\
3677 Feature~#2~of~type~#1\\
3678 In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
3679 }
3680 \msg_error:nn{stex}{error/nomodule}
3681 }
3682

```

```

3683 \str_set:Nx \l_stex_module_name_str {
3684   \prop_item:Nn \l_stex_current_module_prop
3685     { name } / #2 - feature
3686 }
3687
3688 \str_set:Nx \l_stex_module_ns_str {
3689   \prop_item:Nn \l_stex_current_module_prop
3690     { ns }
3691 }
3692
3693
3694 \str_clear:N \l_tmpa_str
3695 \seq_clear:N \l_tmpa_seq
3696 \tl_clear:N \l_tmpa_tl
3697 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
3698   origname = #2,
3699   name      = \l_stex_module_name_str ,
3700   ns        = \l_stex_module_ns_str ,
3701   imports   = \exp_not:o { \l_tmpa_seq } ,
3702   constants = \exp_not:o { \l_tmpa_seq } ,
3703   content   = \exp_not:o { \l_tmpa_tl } ,
3704   file      = \exp_not:o { \g_stex_currentfile_seq } ,
3705   lang      = \l_stex_module_lang_str ,
3706   sig       = \l_tmpa_str ,
3707   meta      = \l_tmpa_str ,
3708   feature   = #1 ,
3709 }
3710
3711 \stex_if_smsmode:F {
3712   \begin{stex_annotate_env}{ feature:#1 }{}
3713   \stex_annotate_invisible:nnn{header}{}{ #3 }
3714 }
3715 }{
3716   \str_set:Nx \l_tmpa_str {
3717     c_stex_feature_
3718     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3719     \prop_item:Nn \l_stex_current_module_prop { name }
3720     _prop
3721   }
3722   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
3723   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3724   \stex_if_smsmode:F {
3725     \end{stex_annotate_env}
3726   }
3727 }
3728

```

32.3 Features

structure

```

3729
3730 \prop_new:N \l_stex_all_structures_prop
3731

```

```

3732 \keys_define:nn { stex / features / structure } {
3733   name          .str_set_x:N = \l__stex_features_structure_name_str ,
3734 }
3735
3736 \cs_new_protected:Nn \__stex_features_structure_args:n {
3737   \str_clear:N \l__stex_features_structure_name_str
3738   \keys_set:nn { stex / features / structure } { #1 }
3739 }
3740
3741 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3742 %   \__stex_features_structure_args:n { ##1 }
3743 %   \str_if_empty:NT \l__stex_features_structure_name_str {
3744 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3745 %   }
3746 % } {
3747 %
3748 %}
3749
3750 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
3751   \__stex_features_structure_args:n { #1 }
3752   \str_if_empty:NT \l__stex_features_structure_name_str {
3753     \str_set:Nx \l__stex_features_structure_name_str { #2 }
3754   }
3755   \exp_args:Nnnx
3756   \begin{structural@feature}{ structure }
3757     { \l__stex_features_structure_name_str }{}
3758     \seq_clear:N \l_tmpa_seq
3759     \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
3760     \stex_smsmode_do:
3761   }{
3762     \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
3763     \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
3764     \str_set:Nx \l_tmpa_str {
3765       \prop_item:Nn \l_stex_current_module_prop { ns } ?
3766       \prop_item:Nn \l_stex_current_module_prop { name }
3767     }
3768     \seq_map_inline:Nn \l_tmpa_seq {
3769       \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
3770     }
3771     \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
3772     \exp_args:Nnx
3773     \AddToHookNext { env / mathstructure / after }{
3774       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3775         \stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
3776       }, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
3777     \STEXexport {
3778       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3779       { \prop_item:Nn \l_stex_current_module_prop { origname } }
3780       { \l_tmpa_str }
3781       \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3782       { #2 } { \l_tmpa_str }
3783     %   \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3784     %     \prop_item:Nn \l_stex_current_module_prop { origname },
3785     %     \l_tmpa_str

```

```

3786 %     }
3787 %     \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3788 %         #2,\l_tmpa_str
3789 %     }
3790 %     \tl_set:cx { #2 } {
3791 %         \stex_invoke_structure:n { \l_tmpa_str }
3792 %     }
3793 % }
3794
3795 \end{structural@feature}
3796 % \g_stex_last_feature_prop
3797 }

```

\instantiate

```

3798 \seq_new:N \l__stex_features_structure_field_seq
3799 \str_new:N \l__stex_features_structure_field_str
3800 \str_new:N \l__stex_features_structure_def_tl
3801 \prop_new:N \l__stex_features_structure_prop
3802 \NewDocumentCommand \instantiate { m O{} m }{
3803     \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3804     \prop_set_eq:Nc \l__stex_features_structure_prop {
3805         c_stex_feature_\l_tmpa_str _prop
3806     }
3807     \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3808     \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3809         \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3810         \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3811             \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3812             \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3813                 {!} \l_tmpa_tl
3814             \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3815                 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3816                 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3817                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3818             }{
3819                 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3820                 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3821                 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3822                     \l_tmpa_tl
3823                 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3824                     \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3825                     \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3826                 }{
3827                     \tl_clear:N \l_tmpb_tl
3828                 }
3829             }
3830         }{
3831             \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3832             \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3833                 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3834                 \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3835                 \tl_clear:N \l_tmpa_tl
3836             }{
3837                 % TODO throw error

```

```

3838     }
3839   }
3840   % \l_tmpa_str: name
3841   % \l_tmpa_tl: definiens
3842   % \l_tmpb_tl: notation
3843   \tl_if_empty:NT \l__stex_features_structure_field_str {
3844     % TODO throw error
3845   }
3846   \str_clear:N \l_tmpb_str
3847
3848   \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3849   \seq_map_inline:Nn \l_tmpa_seq {
3850     \seq_set_split:Nnn \l_tmpb_seq ? { ###1 }
3851     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3852     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3853       \seq_map_break:n {
3854         \str_set:Nn \l_tmpb_str { ###1 }
3855       }
3856     }
3857   }
3858   \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3859   \l_tmpb_str
3860
3861   \tl_if_empty:NTF \l_tmpb_tl {
3862     \tl_if_empty:NF \l_tmpa_tl {
3863       \exp_args:Nx \use:n {
3864         \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3865       }
3866     }
3867   }{
3868     \tl_if_empty:NTF \l_tmpa_tl {
3869       \exp_args:Nx \use:n {
3870         \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3871       }
3872     }{
3873       \exp_args:Nx \use:n {
3874         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3875         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3876       }
3877     }
3878   }
3879 }
3880 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3881 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3882 % #3/\l__stex_features_structure_field_str
3883 % \par
3884 % \expandafter\present\csname
3885 %   l_stex_symdecl_
3886 %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3887 %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3888 %   #3/\l__stex_features_structure_field_str
3889 %   _prop
3890 % \endcsname
3891 }

```

```

3892
3893 \tl_clear:N \l__stex_features_structure_def_tl
3894
3895 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3896 \seq_map_inline:Nn \l_tmpa_seq {
3897   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3898   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3899   \exp_args:Nx \use:n {
3900     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3901
3902     }
3903   }
3904
3905   \prop_if_exist:cF {
3906     l_stex_symdecl_
3907     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3908     \prop_item:Nn \l_stex_current_module_prop {name} ?
3909     #3/\l_tmpa_str
3910     _prop
3911   }{
3912     \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3913     \l_tmpb_str
3914     \exp_args:Nx \use:n {
3915       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3916     }
3917   }
3918 }
3919
3920 \symdecl*[type={\STEXsymbol{module-type}}{
3921   \_stex_term_math_oms:nnnn {
3922     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3923     \prop_item:Nn \l__stex_features_structure_prop {name}
3924     }{0}{0}
3925   }}{#3}
3926
3927 % TODO: -> sms file
3928
3929 \tl_set:cx{ #3 }{
3930   \stex_invoke_structure:nnn {
3931     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3932     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3933   } {
3934     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3935     \prop_item:Nn \l__stex_features_structure_prop {name}
3936   }
3937 }
3938 \stex_smsmode_do:
3939 }

```

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

\stex_invoke_structure:nnn

```

3940 % #1: URI of the instance
3941 % #2: URI of the instantiated module

```



```

3942 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3943   \tl_if_empty:nTF{ #3 }{
3944     \prop_set_eq:Nc \l__stex_features_structure_prop {
3945       c_stex_feature_ #2 _prop
3946     }
3947     \tl_clear:N \l_tmpa_tl
3948     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3949     \seq_map_inline:Nn \l_tmpa_seq {
3950       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3951       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3952       \cs_if_exist:cT {
3953         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3954       }{
3955         \tl_if_empty:NF \l_tmpa_tl {
3956           \tl_put_right:Nn \l_tmpa_tl {,}
3957         }
3958         \tl_put_right:Nx \l_tmpa_tl {
3959           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3960         }
3961       }
3962     }
3963     \exp_args:No \mathstrut \l_tmpa_tl
3964   }{
3965     \stex_invoke_symbol:n{#1/#3}
3966   }
3967 }

```

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

```

3968 </package>

```

Chapter 33

STEX -Statements Implementation

```
3969 <*package>
3970
3971 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3972
3973 <@@=stex_statements>
    Warnings and error messages
3974
\titleemph
3975 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

33.1 Definitions

```
definiendum
3976 \keys_define:nn {stex / definiendum }{
3977   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3978   root      .str_set_x:N    = \l__stex_statements_definiendum_root_str,
3979   gfa       .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
3980 }
3981 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3982   \str_clear:N \l__stex_statements_definiendum_root_str
3983   \tl_clear:N \l__stex_statements_definiendum_post_tl
3984   \str_clear:N \l__stex_statements_definiendum_gfa_str
3985   \keys_set:nn { stex / definiendum }{ #1 }
3986 }
3987 \NewDocumentCommand \definiendum { O{} m m } {
3988   \__stex_statements_definiendum_args:n { #1 }
3989   \stex_get_symbol:n { #2 }
3990   \stex_ref_new_sym_target:n \l__stex_get_symbol_uri_str
3991   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3992     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3993       \tl_set:Nn \l_tmpa_tl { #3 }

```

```

3994   } {
3995     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3996     \tl_set:Nn \l_tmpa_tl {
3997       \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3998     }
3999   }
4000 } {
4001   \tl_set:Nn \l_tmpa_tl { #3 }
4002 }
4003
4004 % TODO root
4005 \rustex_if:TF {
4006   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4007 } {
4008   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4009 }
4010 }
4011 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4012
4013 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
4014
4015 \NewDocumentCommand \definame { 0{ } m } {
4016   \__stex_statements_definiendum_args:n { #1 }
4017   % TODO: root
4018   \stex_get_symbol:n { #2 }
4019   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4020   \str_set:Nx \l_tmpa_str {
4021     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4022   }
4023   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4024   \rustex_if:TF {
4025     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4026       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4027     }
4028   } {
4029     \defemph@uri {
4030       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4031     } { \l_stex_get_symbol_uri_str }
4032   }
4033 }
4034 \stex_deactivate_macro:Nn \definame {definition~environments}
4035
4036 \NewDocumentCommand \Definame { 0{ } m } {
4037   \__stex_statements_definiendum_args:n { #1 }
4038   \stex_get_symbol:n { #2 }
4039   \str_set:Nx \l_tmpa_str {
4040     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4041   }
4042   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4043   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str

```

```

4044 \rustex_if:TF {
4045   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4046     \l_tmpa_str\l__stex_statements_definiendum_post_tl
4047   }
4048 } {
4049   \defemph@uri {
4050     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4051   } { \l_stex_get_symbol_uri_str }
4052 }
4053 }
4054 \stex_deactivate_macro:Nn \Definame {definition-environments}
4055
4056 \NewDocumentCommand \Symname { 0{ } m }{
4057   \stex_symname_args:n { #1 }
4058   \stex_get_symbol:n { #2 }
4059   \str_set:Nx \l_tmpa_str {
4060     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4061   }
4062   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4063   \let\compemph_uri_prev:\compemph@uri
4064   \let\compemph@uri\symrefemph@uri
4065   \exp_args:NNx \use:nn
4066   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
4067     \exp_after:wN \stex_capitalize:n \l_tmpa_str
4068     \l_stex_symname_post_str
4069   ] }
4070   \let\compemph@uri\compemph_uri_prev:
4071 }

```

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

sdefinition

```

4072
4073 \keys_define:nn {stex / sdefinition }{
4074   type      .str_set_x:N = \sdefinitiontype,
4075   id        .str_set_x:N = \sdefinitionid,
4076   name      .str_set_x:N = \sdefinitionname,
4077   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4078   title     .tl_set:N     = \sdefinitiontitle
4079 }
4080 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4081   \str_clear:N \sdefinitiontype
4082   \str_clear:N \sdefinitionid
4083   \str_clear:N \sdefinitionname
4084   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4085   \tl_clear:N \sdefinitiontitle
4086   \keys_set:nn { stex / sdefinition }{ #1 }
4087 }
4088
4089 \NewDocumentEnvironment{sdefinition}{0{}}{
4090   \__stex_statements_sdefinition_args:n{ #1 }
4091   \stex_reactivate_macro:N \definiendum
4092   \stex_reactivate_macro:N \definame
4093   \stex_reactivate_macro:N \Definame

```

```

4094 \stex_if_smsmode:F{
4095   \seq_clear:N \l_tmpa_seq
4096   \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4097     \str_if_eq:nnF{ ##1 }{}{
4098       \stex_get_symbol:n { ##1 }
4099       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4100         \l_stex_get_symbol_uri_str
4101       }
4102     }
4103   }
4104   \exp_args:Nnnx
4105   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4106   \str_if_empty:NF \sdefinitiontype {
4107     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4108   }
4109   \clist_set:No \l_tmpa_clist \sdefinitiontype
4110   \tl_clear:N \l_tmpa_tl
4111   \clist_map_inline:Nn \l_tmpa_clist {
4112     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4113       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4114     }
4115   }
4116   \tl_if_empty:NTF \l_tmpa_tl {
4117     \__stex_statements_sdefinition_start:
4118   }{
4119     \l_tmpa_tl
4120   }
4121 }
4122 \stex_ref_new_doc_target:n \sdefinitionid
4123 \stex_smsmode_do:
4124 }{
4125   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4126   \stex_if_smsmode:F {
4127     \clist_set:No \l_tmpa_clist \sdefinitiontype
4128     \tl_clear:N \l_tmpa_tl
4129     \clist_map_inline:Nn \l_tmpa_clist {
4130       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4131         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4132       }
4133     }
4134     \tl_if_empty:NTF \l_tmpa_tl {
4135       \__stex_statements_sdefinition_end:
4136     }{
4137       \l_tmpa_tl
4138     }
4139     \end{stex_annotate_env}
4140   }
4141 }

```

\stexpatchdefinition

```

4142 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4143   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4144     ~(\sdefinitiontitle)
4145   }~}

```

```

4146 }
4147 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par \medskip }
4148
4149 \newcommand\stexpatchdefinition[3] [] {
4150   \str_set:Nx \l_tmpa_str { #1 }
4151   \str_if_empty:NTF \l_tmpa_str {
4152     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4153     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4154   } {
4155     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname { #2 }
4156     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname { #3 }
4157   }
4158 }

```

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

\inlinedef inline:

```

4159 \keys_define:nn {stex / inlinedef }{
4160   type      .str_set_x:N = \sdefinitiontype,
4161   id        .str_set_x:N = \sdefinitionid,
4162   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4163   name      .str_set_x:N = \sdefinitionname
4164 }
4165 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4166   \str_clear:N \sdefinitiontype
4167   \str_clear:N \sdefinitionid
4168   \str_clear:N \sdefinitionname
4169   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4170   \keys_set:nn { stex / inlinedef } { #1 }
4171 }
4172 \NewDocumentCommand \inlinedef { 0{} m } {
4173   \begingroup
4174   \__stex_statements_inlinedef_args:n { #1 }
4175   \stex_reactivate_macro:N \definiendum
4176   \stex_reactivate_macro:N \definame
4177   \stex_reactivate_macro:N \Definame
4178   \stex_ref_new_doc_target:n \sdefinitionid
4179   \stex_if_smsmode:TF{
4180     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4181   }{
4182     \seq_clear:N \l_tmpa_seq
4183     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4184       \str_if_eq:nnF{ ##1 }{}{
4185         \stex_get_symbol:n { ##1 }
4186         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4187           \l_stex_get_symbol_uri_str
4188         }
4189       }
4190     }
4191     \exp_args:Nnx
4192     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4193       \str_if_empty:NF \sdefinitiontype {
4194         \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{
4195

```

```

4196         #2
4197         \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{ }\sdefinitionname } }
4198     }
4199 }
4200 \endgroup
4201 \stex_smsmode_do:
4202 }

```

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

33.2 Assertions

sassertion

```

4203
4204 \keys_define:nn {stex / sassertion }{
4205     type      .str_set_x:N = \sassertiontype,
4206     id        .str_set_x:N = \sassertionid,
4207     title     .tl_set:N    = \sassertiontitle ,
4208     for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4209     name      .str_set_x:N = \sassertionname
4210 }
4211 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4212     \str_clear:N \sassertiontype
4213     \str_clear:N \sassertionid
4214     \str_clear:N \sassertionname
4215     \clist_clear:N \l__stex_statements_sassertion_for_clist
4216     \tl_clear:N \sassertiontitle
4217     \keys_set:nn { stex / sassertion }{ #1 }
4218 }
4219
4220 %\tl_new:N \g__stex_statements_aftergroup_tl
4221
4222 \NewDocumentEnvironment{sassertion}{0{}}{
4223     \__stex_statements_sassertion_args:n{ #1 }
4224     \stex_if_smsmode:F {
4225         \seq_clear:N \l_tmpa_seq
4226         \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4227             \str_if_eq:nnF{ ##1 }{ }{
4228                 \stex_get_symbol:n { ##1 }
4229                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4230                     \l_stex_get_symbol_uri_str
4231                 }
4232             }
4233         }
4234         \exp_args:Nnnx
4235         \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
4236         \str_if_empty:NF \sassertiontype {
4237             \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
4238         }
4239         \clist_set:N \l_tmpa_clist \sassertiontype
4240         \tl_clear:N \l_tmpa_tl
4241         \clist_map_inline:Nn \l_tmpa_clist {
4242             \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{

```

```

4243         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4244     }
4245 }
4246 \tl_if_empty:NTF \l_tmpa_tl {
4247     \__stex_statements_sassertion_start:
4248 }{
4249     \l_tmpa_tl
4250 }
4251 }
4252 \str_if_empty:NTF \sassertionid {
4253     \str_if_empty:NF \sassertionname {
4254         \stex_ref_new_doc_target:n {}
4255     }
4256 } {
4257     \stex_ref_new_doc_target:n \sassertionid
4258 }
4259 \stex_smsmode_do:
4260 }{
4261     \str_if_empty:NF \sassertionname {
4262         \stex_symdecl_do:nn{ }\sassertionname}
4263         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4264     }
4265     \stex_if_smsmode:F {
4266         \clist_set:No \l_tmpa_clist \sassertiontype
4267         \tl_clear:N \l_tmpa_tl
4268         \clist_map_inline:Nn \l_tmpa_clist {
4269             \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4270                 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4271             }
4272         }
4273         \tl_if_empty:NTF \l_tmpa_tl {
4274             \__stex_statements_sassertion_end:
4275         }{
4276             \l_tmpa_tl
4277         }
4278         \end{stex_annotate_env}
4279     }
4280 }

```

\stexpatchassertion

```

4281
4282 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4283     \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4284         (\sassertiontitle)
4285     }~}
4286 }
4287 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
4288
4289 \newcommand\stexpatchassertion[3] [] {
4290     \str_set:Nx \l_tmpa_str{ #1 }
4291     \str_if_empty:NTF \l_tmpa_str {
4292         \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4293         \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4294     }{

```



```

4295     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4296     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4297   }
4298 }

```

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

\inlineass inline:

```

4299 \keys_define:nn {stex / inlineass }{
4300   type      .str_set_x:N = \sassertiontype,
4301   id        .str_set_x:N = \sassertionid,
4302   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4303   name      .str_set_x:N = \sassertionname
4304 }
4305 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4306   \str_clear:N \sassertiontype
4307   \str_clear:N \sassertionid
4308   \str_clear:N \sassertionname
4309   \clist_clear:N \l__stex_statements_sassertion_for_clist
4310   \keys_set:nn { stex / inlineass }{ #1 }
4311 }
4312 \NewDocumentCommand \inlineass { 0{} m } {
4313   \begingroup
4314   \__stex_statements_inlineass_args:n{ #1 }
4315   \str_if_empty:NTF \sassertionid {
4316     \str_if_empty:NF \sassertionname {
4317       \stex_ref_new_doc_target:n {}
4318     }
4319   } {
4320     \stex_ref_new_doc_target:n \sassertionid
4321   }
4322
4323   \stex_if_smsmode:TF{
4324     \str_if_empty:NF \sassertionname {
4325       \stex_symdecl_do:nn{}{\sassertionname}
4326       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4327     }
4328   }{
4329     \seq_clear:N \l_tmpa_seq
4330     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4331       \str_if_eq:nnF{ ##1 }{}{
4332         \stex_get_symbol:n { ##1 }
4333         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4334           \l_stex_get_symbol_uri_str
4335         }
4336       }
4337     }
4338     \exp_args:Nnx
4339     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4340       \str_if_empty:NF \sassertiontype {
4341         \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4342       }
4343       #2
4344       \str_if_empty:NF \sassertionname {

```

```

4345     \stex_symdecl_do:nn{}{\sassertionname}
4346     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4347   }
4348 }
4349 }
4350 \endgroup
4351 \stex_smsmode_do:
4352 }

```

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

33.3 Examples

`sexample`

```

4353
4354 \keys_define:nn {stex / sexample }{
4355   type      .str_set_x:N = \exampletype,
4356   id        .str_set_x:N = \sexampleid,
4357   title     .tl_set:N     = \sexamplename,
4358   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4359 }
4360 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4361   \str_clear:N \sexampletype
4362   \str_clear:N \sexampleid
4363   \tl_clear:N \sexamplename
4364   \clist_clear:N \l__stex_statements_sexample_for_clist
4365   \keys_set:nn { stex / sexample }{ #1 }
4366 }
4367
4368 \NewDocumentEnvironment{sexample}{0{}}{
4369   \__stex_statements_sexample_args:n{ #1 }
4370   \stex_if_smsmode:F {
4371     \seq_clear:N \l_tmpa_seq
4372     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4373       \str_if_eq:nnF{ ##1 }{ }{
4374         \stex_get_symbol:n { ##1 }
4375         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4376           \l_stex_get_symbol_uri_str
4377         }
4378       }
4379     }
4380     \exp_args:Nnnx
4381     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4382     \str_if_empty:NF \sexampletype {
4383       \stex_annotate_invisible:nnn{type}{\sexampletype}{ }
4384     }
4385     \clist_set:Nn \l_tmpa_clist \sexampletype
4386     \tl_clear:N \l_tmpa_tl
4387     \clist_map_inline:Nn \l_tmpa_clist {
4388       \tl_if_exist:cT {\__stex_statements_sexample_##1_start:}{
4389         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sexample_##1_start:}}
4390       }
4391     }

```

```

4392 \tl_if_empty:NTF \l_tmpa_tl {
4393   \__stex_statements_sexample_start:
4394 }{
4395   \l_tmpa_tl
4396 }
4397 }
4398 \str_if_empty:NF \sexampleid {
4399   \stex_ref_new_doc_target:n \sexampleid
4400 }
4401 \stex_smsmode_do:
4402 }{
4403   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn}{\sexamplename} }
4404   \stex_if_smsmode:F {
4405     \clist_set:Nn \l_tmpa_clist \sexamplename
4406     \tl_clear:N \l_tmpa_tl
4407     \clist_map_inline:Nn \l_tmpa_clist {
4408       \tl_if_exist:cT {\__stex_statements_sexample_##1_end:}{
4409         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sexample_##1_end:}}
4410       }
4411     }
4412     \tl_if_empty:NTF \l_tmpa_tl {
4413       \__stex_statements_sexample_end:
4414     }{
4415       \l_tmpa_tl
4416     }
4417     \end{stex_annotate_env}
4418   }
4419 }

```

`\stexpatchexample`

```

4420
4421 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4422   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplename {
4423     (\sexamplename)
4424   }~}
4425 }
4426 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4427
4428 \newcommand\stexpatchexample[3]{} {
4429   \str_set:Nx \l_tmpa_str{ #1 }
4430   \str_if_empty:NTF \l_tmpa_str {
4431     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4432     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4433   }{
4434     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4435     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4436   }
4437 }

```

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

`\inlineex` inline:

```

4438 \keys_define:nn {stex / inlineex }{
4439   type .str_set_x:N = \sexamplename,

```

```

4440 id      .str_set_x:N = \sexampleid,
4441 for      .clist_set:N = \l__stex_statements_sexample_for_clist ,
4442 name     .str_set_x:N = \sexamplename
4443 }
4444 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
4445   \str_clear:N \sexamplotype
4446   \str_clear:N \sexampleid
4447   \str_clear:N \sexamplename
4448   \clist_clear:N \l__stex_statements_sexample_for_clist
4449   \keys_set:nn { stex / inlineex }{ #1 }
4450 }
4451 \NewDocumentCommand \inlineex { 0{} m } {
4452   \begingroup
4453   \__stex_statements_inlineex_args:n{ #1 }
4454   \str_if_empty:NF \sexampleid {
4455     \stex_ref_new_doc_target:n \sexampleid
4456   }
4457   \stex_if_smsmode:TF{
4458     \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4459   ){
4460     \seq_clear:N \l_tmpa_seq
4461     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4462       \str_if_eq:nnF{ ##1 }{ }{
4463         \stex_get_symbol:n { ##1 }
4464         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4465           \l_stex_get_symbol_uri_str
4466         }
4467       }
4468     }
4469     \exp_args:Nnx
4470     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
4471       \str_if_empty:NF \sexamplotype {
4472         \stex_annotate_invisible:nnn{type}{\sexamplotype}{ }
4473       }
4474       #2
4475       \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{ }\sexamplename } }
4476     }
4477   }
4478   \endgroup
4479   \stex_smsmode_do:
4480 }

```

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

33.4 Logical Paragraphs

`sparagraph`

```

4481 \keys_define:nn { stex / sparagraph } {
4482   id      .str_set_x:N = \sparagraphid ,
4483   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
4484   type    .str_set_x:N = \sparagraphtype ,
4485   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4486   from    .tl_set:N    = \sparagraphfrom ,

```

```

4487 to .tl_set:N = \sparagraphto ,
4488 start .tl_set:N = \l_stex_sparagraph_start_tl ,
4489 name .str_set:N = \sparagraphname
4490 }
4491
4492 \cs_new_protected:Nn \stex_sparagraph_args:n {
4493 \tl_clear:N \l_stex_sparagraph_title_tl
4494 \tl_clear:N \sparagraphfrom
4495 \tl_clear:N \sparagraphto
4496 \tl_clear:N \l_stex_sparagraph_start_tl
4497 \str_clear:N \sparagraphid
4498 \str_clear:N \sparagraphtype
4499 \clist_clear:N \l__stex_statements_sparagraph_for_clist
4500 \str_clear:N \sparagraphname
4501 \keys_set:nn { stex / sparagraph }{ #1 }
4502 }
4503 \newif\if@in@omtext\@in@omtextfalse
4504
4505 \NewDocumentEnvironment {sparagraph} { 0{} } {
4506 \stex_sparagraph_args:n { #1 }
4507 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4508 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
4509 }{
4510 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
4511 }
4512 \@in@omtexttrue
4513 \stex_if_smsmode:F {
4514 \seq_clear:N \l_tmpa_seq
4515 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4516 \str_if_eq:nnF{ ##1 }{}{
4517 \stex_get_symbol:n { ##1 }
4518 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4519 \l_stex_get_symbol_uri_str
4520 }
4521 }
4522 }
4523 \exp_args:Nnnx
4524 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
4525 \str_if_empty:NF \sparagraphtype {
4526 \stex_annotate_invisible:nnn{type}{\sparagraphtype}{ }
4527 }
4528 \str_if_empty:NF \sparagraphfrom {
4529 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
4530 }
4531 \str_if_empty:NF \sparagraphto {
4532 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
4533 }
4534 \clist_set:N \l_tmpa_clist \sparagraphtype
4535 \tl_clear:N \l_tmpa_tl
4536 \clist_map_inline:Nn \sparagraphtype {
4537 \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
4538 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
4539 }
4540 }

```

```

4541 \tl_if_empty:NTF \l_tmpa_tl {
4542   \__stex_statements_sparagraph_start:
4543 }{
4544   \l_tmpa_tl
4545 }
4546 }
4547 \clist_set:No \l_tmpa_clist \sparagraphtype
4548 \str_if_empty:NTF \sparagraphid {
4549   \str_if_empty:NTF \sparagraphname {
4550     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4551       \stex_ref_new_doc_target:n {}
4552     }
4553   } {
4554     \stex_ref_new_doc_target:n {}
4555   }
4556 } {
4557   \stex_ref_new_doc_target:n \sparagraphid
4558 }
4559 \exp_args:NNx
4560 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4561   \clist_map_inline:Nn \__stex_statements_sparagraph_for_clist {
4562     \str_if_eq:nnF{ ##1 }{ }{
4563       \stex_get_symbol:n { ##1 }
4564       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4565     }
4566   }
4567 }
4568 \stex_smsmode_do:
4569 \ignorespacesandpars
4570 }{
4571   \str_if_empty:NF \sparagraphname {
4572     \stex_symdecl_do:nn{ }\sparagraphname}
4573   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4574 }
4575 \stex_if_smsmode:F {
4576   \clist_set:No \l_tmpa_clist \sparagraphtype
4577   \tl_clear:N \l_tmpa_tl
4578   \clist_map_inline:Nn \l_tmpa_clist {
4579     \tl_if_exist:cT {\__stex_statements_sparagraph_##1_end:}{
4580       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sparagraph_##1_end:}}
4581     }
4582   }
4583   \tl_if_empty:NTF \l_tmpa_tl {
4584     \__stex_statements_sparagraph_end:
4585   }{
4586     \l_tmpa_tl
4587   }
4588   \end{stex_annotate_env}
4589 }
4590 }

```

\stexpatchparagraph

```

4591
4592 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {

```

```

4593 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
4594   \tl_if_empty:NF \l_stex_sparagraph_title_tl {
4595     \titleemph{\l_stex_sparagraph_title_tl}:~
4596   }
4597 }{
4598   \titleemph{\l_stex_sparagraph_start_tl}~
4599 }
4600 }
4601 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
4602
4603 \newcommand\stexpatchparagraph[3] [] {
4604   \str_set:Nx \l_tmpa_str{ #1 }
4605   \str_if_empty:NTF \l_tmpa_str {
4606     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
4607     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
4608   }{
4609     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
4610     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
4611   }
4612 }
4613
4614 \keys_define:nn { stex / inlinepara } {
4615   id      .str_set:N = \sparagraphid ,
4616   type    .str_set:N = \sparagraphtype ,
4617   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
4618   from    .tl_set:N   = \sparagraphfrom ,
4619   to      .tl_set:N   = \sparagraphto ,
4620   name    .str_set:N   = \sparagraphname
4621 }
4622 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
4623   \tl_clear:N \sparagraphfrom
4624   \tl_clear:N \sparagraphto
4625   \str_clear:N \sparagraphid
4626   \str_clear:N \sparagraphtype
4627   \clist_clear:N \l__stex_statements_sparagraph_for_clist
4628   \str_clear:N \sparagraphname
4629   \keys_set:nn { stex / inlinepara }{ #1 }
4630 }
4631 \NewDocumentCommand \inlinepara { 0{} m } {
4632   \begingroup
4633   \__stex_statements_inlinepara_args:n{ #1 }
4634   \clist_set:No \l_tmpa_clist \sparagraphtype
4635   \str_if_empty:NTF \sparagraphid {
4636     \str_if_empty:NTF \sparagraphname {
4637       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
4638         \stex_ref_new_doc_target:n {}
4639       }
4640     } {
4641       \stex_ref_new_doc_target:n {}
4642     }
4643   } {
4644     \stex_ref_new_doc_target:n \sparagraphid
4645   }
4646   \stex_if_smsmode:TF{

```

```

4647 \str_if_empty:NF \sparagraphname {
4648   \stex_symdecl_do:nn{}{\sparagraphname}
4649   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4650 }
4651 }{
4652   \seq_clear:N \l_tmpa_seq
4653   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
4654     \str_if_eq:nnF{ ##1 }{}{
4655       \stex_get_symbol:n { ##1 }
4656       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4657         \l_stex_get_symbol_uri_str
4658       }
4659     }
4660   }
4661   \exp_args:Nnx
4662   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
4663     \str_if_empty:NF \sparagraphtype {
4664       \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
4665     }
4666     \str_if_empty:NF \sparagraphfrom {
4667       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
4668     }
4669     \str_if_empty:NF \sparagraphto {
4670       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
4671     }
4672     \str_if_empty:NF \sparagraphname {
4673       \stex_symdecl_do:nn{}{\sparagraphname}
4674       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
4675     }
4676     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
4677       \clist_map_inline:Nn \l_tmpa_seq {
4678         \stex_ref_new_sym_target:n {##1}
4679       }
4680     }
4681     #2
4682   }
4683 }
4684 \endgroup
4685 \stex_smsmode_do:
4686 }
4687

```

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

symboldoc

```

4688 \NewDocumentEnvironment{symboldoc}{m}{
4689   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
4690   \seq_clear:N \l_tmpb_seq
4691   \seq_map_inline:Nn \l_tmpa_seq {
4692     \str_if_eq:nnF{ ##1 }{}{
4693       \stex_get_symbol:n { ##1 }
4694       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
4695         \l_stex_get_symbol_uri_str
4696       }
4697     }
4698   }
4699 }

```



```

4697     }
4698   }
4699   \par
4700   \exp_args:Nnnx
4701   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
4702   ){
4703     \end{stex_annotate_env}
4704   }
4705 </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

```
4706 <*package>
4707 <@@=stex_sproof>
4708
4709 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
4710
```

34.2 Proofs

We first define some keys for the proof environment.

```
4711 \keys_define:nn { stex / spf } {
4712   id          .str_set:N = \l__stex_sproof_spf_id_str,
4713   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
4714   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
4715   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
4716   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
4717   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
4718   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
4719   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
4720   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
4721   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
4722 }
4723 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
4724   \str_clear:N \l__stex_sproof_spf_id_str
4725   \tl_clear:N \l__stex_sproof_spf_display_tl
4726   \tl_clear:N \l__stex_sproof_spf_for_tl
4727   \tl_clear:N \l__stex_sproof_spf_from_tl
4728   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
4729   \tl_clear:N \l__stex_sproof_spf_type_tl
4730   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹³EDNOTE: need an implementation for L^AT_EX_ML

```

4731 \tl_clear:N \l__stex_sproof_spf_continues_tl
4732 \tl_clear:N \l__stex_sproof_spf_functions_tl
4733 \tl_clear:N \l__stex_sproof_spf_method_tl
4734 \keys_set:nn { stex / spf }{ #1 }
4735 }

```

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

```

4736 \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.

```

4737 \newcount\count_ten
4738 \newenvironment{pst@with@label}[1]{
4739   \edef\pst@label{#1}
4740   \advance\count_ten by 1\relax
4741   \count_ten=1
4742 }{
4743   \advance\count_ten by -1\relax
4744 }

```

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

```

4745 \def\the@pst@label{
4746   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
4747 }

```

(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.

```

4748 \keys_define:nn { stex / pstlabel }{
4749   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
4750   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
4751   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
4752 }
4753 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

4754 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
4755 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
4756 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
4757 }
4758 \__stex_sproof_pstlabel_args:n {}
4759 \newcommand\setpstlabelstyle[1]{
4760   \__stex_sproof_pstlabel_args:n {#1}
4761 }
4762 \newcommand\setpstlabelstyledefault{%
4763   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
4764 }

```

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

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

```

4765 \ExplSyntaxOff
4766 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
4767 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
4768 \def\pst@make@label@short#1#2{#2}
4769 \def\pst@make@label@empty#1#2{}
4770 \ExplSyntaxOn
4771 \def\pstlabelstyle#1{%
4772   \def\pst@make@label{\use:c{pst@make@label@#1}}%
4773 }%
4774 \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.

```

4775 \def\next@pst@label{%
4776   \global\advance\count\count10 by 1%
4777 }%

```

(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

```

4778 \def\sproof@box{
4779   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
4780 }
4781 \def\spf@proofend{\sproof@box}
4782 \def\sproofend{
4783   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
4784     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
4785   }
4786 }
4787 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

4788 \def\spf@proofsketch@kw{Proof Sketch}
4789 \def\spf@proof@kw{Proof}
4790 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

4791 \AddToHook{begindocument}{
4792   \ltx@ifpackageloaded{babel}{
4793     \makeatletter
4794     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4795     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4796       \input{sproof-ngerman.ldf}
4797     }
4798     \clist_if_in:NnT \l_tmpa_clist {finnish}{
4799       \input{sproof-finnish.ldf}
4800     }
4801     \clist_if_in:NnT \l_tmpa_clist {french}{
4802       \input{sproof-french.ldf}
4803     }
4804     \clist_if_in:NnT \l_tmpa_clist {russian}{
4805       \input{sproof-russian.ldf}
4806     }
4807     \makeatother
4808   }{}
4809 }

```

`spfsketch`

```

4810 \newcommand\spfsketch[2][]{
4811   \__stex_sproof_spf_args:n{#1}
4812   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4813     \titleemph{
4814       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4815         \spf@proofsketch@kw
4816       }{
4817         \l__stex_sproof_spf_type_tl
4818       }
4819     }:
4820   }
4821   {~#2}
4822   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
4823   \sproofend
4824 }

```

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

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

```

4825 \newenvironment{spfeq}[2][]{
4826   \__stex_sproof_spf_args:n{#1}
4827   %\sref@target
4828   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
4829     \titleemph{
4830       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
4831         \spf@proof@kw
4832       }{

```

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

¹⁵EDNOTE: document above

```

4833     \l__stex_sproof_spf_type_tl
4834   }
4835   }:
4836 }
4837 {-#2}
4838 \begin{displaymath}\begin{array}{rcll}
4839 }{
4840 \end{array}\end{displaymath}
4841 }

```

(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.

```

4842 \newenvironment{spf@proof}[2] []{
4843   \l__stex_sproof_spf_args:n{#1}
4844   %\sref@target
4845   \count_ten=10
4846   \par\noindent
4847   \tl_if_eq:NNTF \l__stex_sproof_spf_display_tl\spf@flow{
4848     \titleemph{
4849       \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {
4850         \spf@proof@kw
4851       }{
4852         \l__stex_sproof_spf_type_tl
4853       }
4854     }:
4855   }
4856   {-#2}
4857   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
4858   \def\pst@label{}
4859   \newcount\pst@count% initialize the labeling mechanism
4860   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
4861   }{
4862     \end{pst@with@label}\end{description}
4863   }
4864   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
4865   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

4866 \newcommand\spfidea[2] []{
4867   \l__stex_sproof_spf_args:n{#1}
4868   \titleemph{
4869     \tl_if_empty:NNTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
4870       \l__stex_sproof_spf_type_tl
4871     }:
4872   }-#2
4873   \sproofend
4874 }

```

(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
4875 \newenvironment{spfstep}[1][]{
4876   \_stex_sproof_spf_args:n{#1}
4877   \@in@omtexttrue
4878   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4879     \item[\the@pst@label]
4880   }
4881   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
4882     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
4883   }
4884   %\sref@label{id{\pst@label}}
4885   \ignorespacesandpars
4886 }{
4887   \next@pst@label\ignorespacesandpars
4888 }

```

sproofcomment

```

4889 \newenvironment{sproofcomment}[1][]{
4890   \_stex_sproof_spf_args:n{#1}
4891   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4892     \item[\the@pst@label]
4893   }
4894 }{
4895   \next@pst@label
4896 }

```

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.

```

4897 \newenvironment{subproof}[2][]{
4898   \_stex_sproof_spf_args:n{#1}
4899   \def\@test{#2}
4900   \ifx\@test\empty\else
4901     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4902       \item[\the@pst@label]
4903     }{#2}
4904   \fi
4905   \begin{pst@with@label}{\pst@label,\number\count_ten}
4906 }{
4907   \end{pst@with@label}\next@pst@label
4908 }

```

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

```

4909 \newenvironment{spfcases}[2][]{
4910   \def\@test{#1}
4911   \ifx\@test\empty
4912     \begin{subproof}[method=by-cases]{#2}

```

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

```

4913 \else
4914   \begin{subproof}[#1,method=by-cases]{#2}
4915 \fi
4916 }{
4917   \end{subproof}
4918 }

```

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

```

4919 \newenvironment{spfcase}[2] [] {
4920   \__stex_sproof_spf_args:n{#1}
4921   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4922     \item[\the@pst@label]
4923   }
4924   \def\@test{#2}
4925   \ifx\@test\@empty
4926   \else
4927     {\titleemph{#2}:~}
4928   \fi
4929   \begin{pst@with@label}{\pst@label,\number\count_ten}
4930 }{
4931   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4932     \sproofend
4933   }
4934   \end{pst@with@label}
4935   \next@pst@label
4936 }

```

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

```

4937 \newcommand\spfcasesketch[3] [] {
4938   \__stex_sproof_spf_args:n{#1}
4939   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
4940     \item[\the@pst@label]
4941   }
4942   \def\@test{#2}
4943   \ifx\@test\@empty
4944   \else
4945     {\titleemph{#2}:~}
4946   \fi#3
4947   \next@pst@label
4948 }%

```

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.

```

4949 \keys_define:nn { stex / just }{
4950   id          .str_set:x:N = \l__stex_sproof_just_id_str,
4951   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
4952   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
4953   args        .tl_set:N    = \l__stex_sproof_just_args_tl
4954 }

```


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

`justification`

```
4955 \newenvironment{justification}[1] [] {}{}
```

`\premise`

```
4956 \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.

```
4957 \newcommand\justarg[2] [] {#2}
```

```
4958 \end{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

```
4959 <*package>
4960
4961 %%%%%%%%%%% others.dtx %%%%%%%%%%%
4962
4963 <@@=stex_others>
      Warnings and error messages
4964 % None

\MSC Math subject classifier

4965 \NewDocumentCommand \MSC {m} {
4966 % TODO
4967 }

(End definition for \MSC. This function is documented on page ??.)
      Patching tikzinput, if loaded
4968 \@ifpackageloaded{tikzinput}{
4969 \RequirePackage{stex-tikzinput}
4970 }{}
4971 </package>
```

Chapter 36

STEX -Metatheory Implementation

```
4972 <*package>
4973 <@@=stex_modules>
4974
4975 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
4976
4977 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
4978 \begingroup
4979 \stex_module_setup:nn{
4980   ns=\c_stex_metatheory_ns_str,
4981   meta=NONE
4982 }{Metatheory}
4983 \stex_reactivate_macro:N \symdecl
4984 \stex_reactivate_macro:N \notation
4985 \stex_reactivate_macro:N \symdef
4986 \ExplSyntaxOff
4987 \csname stex_suppress_html:n\endcsname{
4988   % is-a (a:A, a \in A, a is an A, etc.)
4989   \symdecl[args=ai]{isa}
4990   \notation[typed]{isa}{#1 \comp{:} #2}{##1 \comp, ##2}
4991   \notation[in]{isa}{#1 \comp\in #2}{##1 \comp, ##2}
4992   \notation[pred]{isa}{#2\comp(#1 \comp)}{##1 \comp, ##2}
4993
4994   % bind (\forall, \Pi, \lambda etc.)
4995   \symdecl[args=Bi]{bind}
4996   \notation[forall]{bind}{\comp\forall #1.\;#2}{##1 \comp, ##2}
4997   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{##1 \comp, ##2}
4998   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
4999
5000   % dummy variable
5001   \symdecl{dummyvar}
5002   \notation[underscore]{dummyvar}{\comp\_}
5003   \notation[dot]{dummyvar}{\comp\cdot}
5004   \notation[dash]{dummyvar}{\comp{\rm --}}
5005
5006   %fromto (function space, Hom-set, implication etc.)
```

```

5007 \symdecl[args=ai]{fromto}
5008 \notation[xarrow]{fromto}{#1 \comp\to #2}{##1 \comp\times ##2}
5009 \notation[arrow]{fromto}{#1 \comp\to #2}{##1 \comp\to ##2}
5010
5011 % mapto (lambda etc.)
5012 \%symdecl[args=Bi]{mapto}
5013 \%notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
5014 \%notation[lambda]{mapto}{\comp\lambda #1 \comp.\; #2}{#1 \comp, #2}
5015 \%notation[lambdau]{mapto}{\comp\lambda_{#1} \comp.\; #2}{#1 \comp, #2}
5016
5017 % function/operator application
5018 \symdecl[args=ia]{apply}
5019 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5020 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{##1 \; ; ##2}
5021
5022 % ‘type’ of all collections (sets, classes, types, kinds)
5023 \symdecl{collection}
5024 \notation[U]{collection}{\comp{\mathcal{U}}}
5025 \notation[set]{collection}{\comp{\textsf{Set}}}
5026
5027 % collection of propositions/booleans/truth values
5028 \symdecl[name=proposition]{prop}
5029 \notation[prop]{prop}{\comp{\rm prop}}
5030 \notation[BOOL]{prop}{\comp{\rm BOOL}}
5031
5032 % sequences
5033 \symdecl[args=1]{seqtype}
5034 \notation[kleene]{seqtype}{#1^{\comp\ast}}
5035
5036 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
5037 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
5038
5039 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{##1\comp,##2}
5040 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses},#2}{##1\comp,##2}
5041 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses},#2\comp{\,\ellipses},#3}
5042
5043 % letin (‘let’, local definitions, variable substitution)
5044 \symdecl[args=bii]{letin}
5045 \notation[let]{letin}{\comp{\rm let}}\;#1\comp{=#2}\; \comp{\rm in}}\;#3}
5046 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
5047 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
5048
5049 % structures
5050 \symdecl*[args=1]{module-type}
5051 \notation{module-type}{\mathtt{MOD} #1}
5052 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
5053 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5054
5055 }
5056 \ExplSyntaxOn
5057 \stex_add_to_current_module:n{
5058   \let\appa\apply
5059   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5060   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}

```

```

5061 \def\livar{\csname sequence-index\endcsname[li]}
5062 \def\uivar{\csname sequence-index\endcsname[ui]}
5063 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5064 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5065 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5066 }
5067 \__stex_modules_end_module:
5068 \endgroup
5069 \</package>

```

Chapter 37

Tikzinput Implementation

```
5070 <*package>
5071
5072 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5073
5074 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
5075 \RequirePackage{l3keys2e}
5076
5077 \keys_define:nn { tikzinput } {
5078   image .bool_set:N = \c_tikzinput_image_bool,
5079   image .default:n = false ,
5080   unknown .code:n = {}
5081 }
5082
5083 \ProcessKeysOptions { tikzinput }
5084
5085 \bool_if:NTF \c_tikzinput_image_bool {
5086   \RequirePackage{graphicx}
5087
5088   \providecommand\usetikzlibrary[]{}
5089   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
5090 }{
5091   \RequirePackage{tikz}
5092   \RequirePackage{standalone}
5093
5094   \newcommand \tikzinput [2] [] {
5095     \setkeys{Gin}{#1}
5096     \ifx \Gin@ewidth \Gin@exclamation
5097       \ifx \Gin@eheight \Gin@exclamation
5098         \input { #2 }
5099       \else
5100         \resizebox{!}{ \Gin@eheight }{
5101           \input { #2 }
5102         }
5103       \fi
5104     \else
5105       \ifx \Gin@eheight \Gin@exclamation
5106         \resizebox{ \Gin@ewidth }{!}{
5107           \input { #2 }
```

```

5108     }
5109     \else
5110         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5111             \input { #2 }
5112         }
5113     \fi
5114 \fi
5115 }
5116 }
5117
5118 \newcommand \ctikzinput [2] [] {
5119     \begin{center}
5120         \tikzinput [ #1 ] { #2 }
5121     \end{center}
5122 }
5123
5124 \@ifpackageloaded{stex}{
5125     \RequirePackage{stex-tikzinput}
5126 }{}
5127
5128 </package>
5129 <*stex>
5130 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
5131 \RequirePackage{stex}
5132 \RequirePackage{tikzinput}
5133
5134 \newcommand\mhtikzinput [2] [] {%
5135     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5136     \stex_in_repository:nn\Gin@mhrepos{
5137         \tikzinput [ #1 ] {\mhp{##1}{#2}}
5138     }
5139 }
5140 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
5141 </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.

```
5142 \*cls)
5143 \@@=document_structure)
5144 \ProvidesExplClass{document-structure}{2022/02/10}{3.0}{Modular Document Structure Class}
5145 \RequirePackage{l3keys2e,expl-keystr-compat}
```

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`

```
5146 \keys_define:nn{ document-structure / pkg }{
5147   class      .str_set_x:N = \c_document_structure_class_str,
5148   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5149   report     .code:n      = {
5150     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5151     \str_set:Nn \c_document_structure_class_str {report}
5152   },
5153   book       .code:n      = {
5154     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5155     \str_set:Nn \c_document_structure_class_str {book}
5156   },
5157   bookpart   .code:n      = {
5158     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
5159     \str_set:Nn \c_document_structure_class_str {book}
5160     \str_set:Nn \c_document_structure_topsect_str {chapter}
5161   },
```



```

5162 docopt      .str_set_x:N = \c_document_structure_docopt_str,
5163 unknown     .code:n      = {
5164   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
5165 }
5166 }
5167 \ProcessKeysOptions{ document-structure / pkg }
5168 \str_if_empty:NT \c_document_structure_class_str {
5169   \str_set:Nn \c_document_structure_class_str {article}
5170 }
5171 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
5172   {\c_document_structure_class_str}
5173

```

38.3 Beefing up the document environment

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

```

5174 \RequirePackage{document-structure}
5175 \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

```

5176 \keys_define:nn { document-structure / document }{
5177   id .str_set_x:N = \c_document_structure_document_id_str
5178 }
5179 \let\__document_structure_orig_document=\document
5180 \renewcommand{\document}[1][]{
5181   \keys_set:nn{ document-structure / document }{ #1 }
5182   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
5183   \__document_structure_orig_document
5184 }

```

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

```

5185 }
5186 \</cls>

```

38.4 Implementation: document-structure Package

```

5187 \<*package>
5188 \ProvidesExplPackage{document-structure}{2022/02/10}{3.0}{Modular Document Structure}
5189 \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

```

5190
5191 \keys_define:nn{ document-structure / pkg }{
5192   class      .str_set_x:N = \c_document_structure_class_str,
5193   topsect     .str_set_x:N = \c_document_structure_topsect_str,
5194   % showignores .bool_set:N = \c_document_structure_showignores_bool,
5195 }
5196 \ProcessKeysOptions{ document-structure / pkg }
5197 \str_if_empty:NT \c_document_structure_class_str {
5198   \str_set:Nn \c_document_structure_class_str {article}
5199 }
5200 \str_if_empty:NT \c_document_structure_topsect_str {
5201   \str_set:Nn \c_document_structure_topsect_str {section}
5202 }

```

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

```

5203 \RequirePackage{xspace}
5204 \RequirePackage{comment}
5205 \AddToHook{begindocument}{
5206   \ltx@ifpackageloaded{babel}{
5207     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5208     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5209       \makeatletter\input{omdoc-ngerman.ldf}\makeatother
5210     }
5211   }{}
5212 }

```

`\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.

```

5213 \int_new:N \l_document_structure_section_level_int
5214 \str_case:VnF \c_document_structure_topsect_str {
5215   {part}}{
5216     \int_set:Nn \l_document_structure_section_level_int {0}
5217   }
5218   {chapter}}{
5219     \int_set:Nn \l_document_structure_section_level_int {1}
5220   }
5221 }{
5222   \str_case:VnF \c_document_structure_class_str {
5223     {book}}{
5224       \int_set:Nn \l_document_structure_section_level_int {0}
5225     }
5226     {report}}{
5227       \int_set:Nn \l_document_structure_section_level_int {0}
5228     }
5229   }{
5230     \int_set:Nn \l_document_structure_section_level_int {2}
5231   }
5232 }

```

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

```
5233 \def\current@section@level{document}%
5234 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
5235 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
5236 \cs_new_protected:Npn \skipomgroup {
5237   \ifcase\l_document_structure_section_level_int
5238   \or\stepcounter{part}
5239   \or\stepcounter{chapter}
5240   \or\stepcounter{section}
5241   \or\stepcounter{subsection}
5242   \or\stepcounter{subsubsection}
5243   \or\stepcounter{paragraph}
5244   \or\stepcounter{subparagraph}
5245   \fi
5246 }
```

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

`blindomgroup`

```
5247 \newcommand\at@begin@blindomgroup[1]{%
5248 \newenvironment{blindomgroup}
5249 {
5250   \int_incr:N\l_document_structure_section_level_int
5251   \at@begin@blindomgroup\l_document_structure_section_level_int
5252 }{}}
```

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

```
5253 \newcommand\omgroup@nonum[2]{
5254   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
5255   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
5256 }
```

(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.

```
5257 \newcommand\omgroup@num[2]{
```

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

```

5258 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
5259   \@nameuse{#1}{#2}
5260 }{
5261   \cs_if_exist:NTF\rdfmata@sectioning{
5262     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
5263   }{
5264     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
5265   }
5266 }
5267 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
5268 }

```

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

omgroup

```

5269 \keys_define:nn { document-structure / omgroup }{
5270   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
5271   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
5272   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
5273   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
5274   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
5275   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
5276   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
5277   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
5278   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
5279   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
5280 }
5281 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
5282   \str_clear:N \l__document_structure_omgroup_id_str
5283   \str_clear:N \l__document_structure_omgroup_date_str
5284   \clist_clear:N \l__document_structure_omgroup_creators_clist
5285   \clist_clear:N \l__document_structure_omgroup_contributors_clist
5286   \tl_clear:N \l__document_structure_omgroup_srccite_tl
5287   \tl_clear:N \l__document_structure_omgroup_type_tl
5288   \tl_clear:N \l__document_structure_omgroup_short_tl
5289   \tl_clear:N \l__document_structure_omgroup_display_tl
5290   \tl_clear:N \l__document_structure_omgroup_intro_tl
5291   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
5292   \keys_set:nn { document-structure / omgroup } { #1 }
5293 }

```

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.

```

5294 \newif\if@mainmatter\@mainmattertrue
5295 \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.

```

5296 \keys_define:nn { document-structure / sectioning }{
5297   name .str_set_x:N = \l__document_structure_sect_name_str ,
5298   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
5299   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
5300   clear .default:n = {true} ,
5301   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

5302   num      .default:n    = {true}
5303 }
5304 \cs_new_protected:Nn \__document_structure_sect_args:n {
5305   \str_clear:N \l__document_structure_sect_name_str
5306   \str_clear:N \l__document_structure_sect_ref_str
5307   \bool_set_false:N \l__document_structure_sect_clear_bool
5308   \bool_set_false:N \l__document_structure_sect_num_bool
5309   \keys_set:nn { document-structure / sectioning } { #1 }
5310 }
5311 \newcommand\omdoc@sectioning[3][]{
5312   \__document_structure_sect_args:n {#1}
5313   \let\omdoc@sect@name\l__document_structure_sect_name_str
5314   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
5315   \if@mainmatter% numbering not overridden by frontmatter, etc.
5316     \bool_if:NTF \l__document_structure_sect_num_bool {
5317       \omgroup@num{#2}{#3}
5318     }{
5319       \omgroup@nonum{#2}{#3}
5320     }
5321     \def\current@section@level{\omdoc@sect@name}
5322   \else
5323     \omgroup@nonum{#2}{#3}
5324   \fi
5325 }% 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.

```

5326 \newcommand\omgroup@redefine@addtocontents[1]{%
5327 %\edef\__document_structureimport{#1}%
5328 %\@for\@I:=\__document_structureimport\do{%
5329 %\edef\@path{\csname module@\@I @path\endcsname}%
5330 %\@ifundefined{tf@toc}\relax%
5331 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
5332 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
5333 %\def\addcontentsline##1##2##3{%
5334 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5335 %\else% hyperref.sty not loaded
5336 %\def\addcontentsline##1##2##3{%
5337 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
5338 %\fi
5339 }% hypreref.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.

```

5340 \newenvironment{omgroup}[2][]{% keys, title
5341 {
5342   \__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.

```

5343 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
5344   \omgroup@redefine@addtocontents{
5345     \@ifundefined{module@id}\used@modules%

```

```

5346      %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
5347    }
5348  }

```

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

```

5349  \int_incr:N\l_document_structure_section_level_int
5350  \ifcase\l_document_structure_section_level_int
5351    \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
5352    \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
5353    \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
5354    \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
5355    \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
5356    \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
5357    \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
5358  \fi
5359  \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
5360  \str_if_empty:NF \l__document_structure_omgroup_id_str {
5361    \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
5362  }
5363  }% for customization
5364  {}

```

and finally, we localize the sections

```

5365  \newcommand\omdoc@part@kw{Part}
5366  \newcommand\omdoc@chapter@kw{Chapter}
5367  \newcommand\omdoc@section@kw{Section}
5368  \newcommand\omdoc@subsection@kw{Subsection}
5369  \newcommand\omdoc@subsubsection@kw{Subsubsection}
5370  \newcommand\omdoc@paragraph@kw{paragraph}
5371  \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`

```

5372  \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).

```

5373  \cs_if_exist:NTF\frontmatter{
5374    \let\__document_structure_orig_frontmatter\frontmatter
5375    \let\frontmatter\relax
5376  }{
5377    \tl_set:Nn\__document_structure_orig_frontmatter{
5378      \clearpage
5379      \@mainmatterfalse
5380      \pagenumbering{roman}

```

```

5381 }
5382 }
5383 \cs_if_exist:NTF\backmatter{
5384   \let\__document_structure_orig_backmatter\backmatter
5385   \let\backmatter\relax
5386 }{
5387   \tl_set:Nn\__document_structure_orig_backmatter{
5388     \clearpage
5389     \@mainmatterfalse
5390     \pagenumbering{roman}
5391   }
5392 }

```

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.

```

5393 \newenvironment{frontmatter}{
5394   \__document_structure_orig_frontmatter
5395 }{
5396   \cs_if_exist:NTF\mainmatter{
5397     \mainmatter
5398   }{
5399     \clearpage
5400     \@mainmattertrue
5401     \pagenumbering{arabic}
5402   }
5403 }

```

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

```

5404 \newenvironment{backmatter}{
5405   \__document_structure_orig_backmatter
5406 }{
5407   \cs_if_exist:NTF\mainmatter{
5408     \mainmatter
5409   }{
5410     \clearpage
5411     \@mainmattertrue
5412     \pagenumbering{arabic}
5413   }
5414 }

```

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

```

5415 \@mainmattertrue\pagenumbering{arabic}

```

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

```

5416 \def \c__document_structure_document_str{document}
5417 \newcommand\afterprematurestop{}
5418 \def\prematurestop@endomgroup{
5419   \unless\ifx\@currenvir\c__document_structure_document_str
5420     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter\expandafter
5421     \expandafter\prematurestop@endomgroup

```

```

5422 \fi
5423 }
5424 \providecommand\prematurestop{
5425 \message{Stopping~sTeX~processing~prematurely}
5426 \prematurestop@endomgroup
5427 \afterprematurestop
5428 \end{document}
5429 }

```

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

38.8 Global Variables

\setSGvar set a global variable

```

5430 \RequirePackage{etoolbox}
5431 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

5432 \newrobustcmd\useSGvar[1]{%
5433 \@ifundefined{sTeX@Gvar@#1}
5434 {\PackageError{document-structure}
5435 {The sTeX Global variable #1 is undefined}
5436 {set it with \protect\setSGvar}}
5437 \@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.

```

5438 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
5439 \@ifundefined{sTeX@Gvar@#1}
5440 {\PackageError{document-structure}
5441 {The sTeX Global variable #1 is undefined}
5442 {set it with \protect\setSGvar}}
5443 {\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.

```
5444 \*cls)
5445 \@@=notesslides)
5446 \ProvidesExplClass{notesslides}{2022/02/10}{3.0}{notesslides Class}
5447 \RequirePackage{l3keys2e,expl-keystr-compat}
5448
5449 \keys_define:nn{notesslides / cls}{
5450   class .code:n = {
5451     \PassOptionsToClass{\CurrentOption}{document-structure}
5452     \str_if_eq:nnT{#1}{book}{
5453       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5454     }
5455     \str_if_eq:nnT{#1}{report}{
5456       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
5457     }
5458   },
5459   notes .bool_set:N = \c__notesslides_notes_bool ,
5460   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
5461   unknown .code:n = {
5462     \PassOptionsToClass{\CurrentOption}{document-structure}
5463     \PassOptionsToClass{\CurrentOption}{beamer}
5464     \PassOptionsToPackage{\CurrentOption}{notesslides}
5465   }
5466 }
5467 \ProcessKeysOptions{ notesslides / cls }
5468 \bool_if:NTF \c__notesslides_notes_bool {
5469   \PassOptionsToPackage{notes=true}{notesslides}
5470 }{
5471   \PassOptionsToPackage{notes=false}{notesslides}
5472 }
5473 \</cls)
```

now we do the same for the notesslides package.

```

5474 <*package>
5475 \ProvidesExplPackage{notesslides}{2022/02/10}{3.0}{notesslides Package}
5476 \RequirePackage{l3keys2e,expl-keystr-compat}
5477
5478 \keys_define:nn{notesslides / pkg}{
5479   topsect      .str_set_x:N = \c__notesslides_topsect_str,
5480   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
5481   notes        .bool_set:N = \c__notesslides_notes_bool ,
5482   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
5483   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
5484   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
5485   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
5486   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
5487   unknown      .code:n      = {
5488     \PassOptionsToClass{\CurrentOption}{stex}
5489     \PassOptionsToClass{\CurrentOption}{tikzinput}
5490   }
5491 }
5492 \ProcessKeysOptions{ notesslides / pkg }
5493 \newif\ifnotes
5494 \bool_if:NTF \c__notesslides_notes_bool {
5495   \notesttrue
5496 }{
5497   \notesfalse
5498 }
5499

```

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

```

5500 \str_if_empty:NTF \c__notesslides_topsect_str {
5501   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
5502 }{
5503   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
5504 }
5505 </package>

```

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

```

5506 <*cls>
5507 \bool_if:NTF \c__notesslides_notes_bool {
5508   \LoadClass{document-structure}
5509 }{
5510   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
5511   \newcounter{Item}
5512   \newcounter{paragraph}
5513   \newcounter{subparagraph}
5514   \newcounter{Hfootnote}
5515   \RequirePackage{document-structure}
5516 }

```

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

```

5517 \RequirePackage{notesslides}
5518 </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.

```

5519 \*package>
5520 \bool_if:NT \c__notesslides_notes_bool {
5521   \RequirePackage{a4wide}
5522   \RequirePackage{marginnote}
5523   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
5524   \RequirePackage{mdframed}
5525   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
5526   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
5527 }
5528 \RequirePackage{stex-tikzinput}
5529 \RequirePackage{etoolbox}
5530 \RequirePackage{amssymb}
5531 \RequirePackage{amsmath}
5532 \RequirePackage{comment}
5533 \RequirePackage{textcomp}
5534 \RequirePackage{url}
5535 \RequirePackage{graphicx}
5536 \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

```

5537 \bool_if:NT \c__notesslides_notes_bool {
5538   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
5539 }

```

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

```

5540 \newcounter{slide}
5541 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
5542 \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.

```

5543 \bool_if:NTF \c__notesslides_notes_bool {
5544   \renewenvironment{note}{\ignorespaces}{}
5545 }{
5546   \excludcomment{note}
5547 }

```

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.

```
5548 \bool_if:NT \c__notesslides_notes_bool {
5549   \newlength{\slideframewidth}
5550   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
5551 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
5552   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
5553     \bool_set_true:N #1
5554   }{
5555     \bool_set_false:N #1
5556   }
5557 }
5558 \keys_define:nn{notesslides / frame}{
5559   label .str_set_x:N = \l__notesslides_frame_label_str,
5560   allowframebreaks .code:n = {
5561     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
5562   },
5563   allowdisplaybreaks .code:n = {
5564     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
5565   },
5566   fragile .code:n = {
5567     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
5568   },
5569   shrink .code:n = {
5570     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
5571   },
5572   squeeze .code:n = {
5573     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
5574   },
5575   t .code:n = {
5576     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
5577   },
5578 }
5579 \cs_new_protected:Nn \__notesslides_frame_args:n {
5580   \str_clear:N \l__notesslides_frame_label_str
5581   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
5582   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
5583   \bool_set_true:N \l__notesslides_frame_fragile_bool
5584   \bool_set_true:N \l__notesslides_frame_shrink_bool
5585   \bool_set_true:N \l__notesslides_frame_squeeze_bool
5586   \bool_set_true:N \l__notesslides_frame_t_bool
5587   \keys_set:nn { notesslides / frame }{ #1 }
5588 }
```

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

```
5589 \renewenvironment{frame}[1][]{
5590   \__notesslides_frame_args:n{#1}
5591   \sffamily
5592   \stepcounter{slide}
5593   \def\@currentlabel{\theslide}
5594   \str_if_empty:NF \l__notesslides_frame_label_str {
5595     \label{\l__notesslides_frame_label_str}
```

5596 }
5597

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

5597 \def\itemize@level{outer}
5598 \def\itemize@outer{outer}
5599 \def\itemize@inner{inner}
5600 \renewcommand\newpage{\addtocounter{framenum}{1}}
5601 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
5602 \renewenvironment{itemize}{
5603 \ifx\itemize@level\itemize@outer
5604 \def\itemize@label{\rhd\$}
5605 \fi
5606 \ifx\itemize@level\itemize@inner
5607 \def\itemize@label{\$\scriptstyle\rhd\$}
5608 \fi
5609 \begin{list}
5610 {\itemize@label}
5611 {\setlength{\labelsep}{.3em}
5612 \setlength{\labelwidth}{.5em}
5613 \setlength{\leftmargin}{1.5em}
5614 }
5615 \edef\itemize@level{\itemize@inner}
5616 }{
5617 \end{list}
5618 }

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

5619 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth=100pt]
5620 }{
5621 \medskip\miko@slidelabel\end{mdframed}
5622 }

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

\frametitle

5623 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
5624 }

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

EdN:21

\pause 21

5625 \bool_if:NT \c__notesslides_notes_bool {
5626 \newcommand\pause{
5627 }

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

nparagraph

5628 \bool_if:NTF \c__notesslides_notes_bool {
5629 \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
5630 }{
5631 \excludecomment{nparagraph}
5632 }

²¹EdNOTE: MK: fake it in notes mode for now

```

nomgroup
5633 \bool_if:NTF \c__notesslides_notes_bool {
5634   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
5635 }{
5636   \excludecomment{nomgroup}
5637 }

ntheorem
5638 \bool_if:NTF \c__notesslides_notes_bool {
5639   \newenvironment{ntheorem}[1] [] {\begin{sdefinition}[#1]}{\end{sdefinition}}
5640 }{
5641   \excludecomment{ntheorem}
5642 }

nassertion
5643 \bool_if:NTF \c__notesslides_notes_bool {
5644   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}{\end{sassertion}}
5645 }{
5646   \excludecomment{nassertion}
5647 }

nsproof
5648 \bool_if:NTF \c__notesslides_notes_bool {
5649   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
5650 }{
5651   \excludecomment{nsproof}
5652 }

nexample
5653 \bool_if:NTF \c__notesslides_notes_bool {
5654   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}{\end{sexample}}
5655 }{
5656   \excludecomment{nexample}
5657 }

\inputref@*skip We customize the hooks for in \inputref.
5658 \def\inputref@preskip{\smallskip}
5659 \def\inputref@postskip{\medskip}

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

\inputref*
5660 \let\orig@inputref\inputref
5661 \def\inputref{\@ifstar\ninputref\orig@inputref}
5662 \newcommand\ninputref[2] [] {
5663   \bool_if:NT \c__notesslides_notes_bool {
5664     \orig@inputref[#1]{#2}
5665   }
5666 }

(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>}`.

```

5667 \newlength{\slidelogoheight}
5668
5669 \bool_if:NTF \c__notesslides_notes_bool {
5670   \setlength{\slidelogoheight}{.4cm}
5671 }{
5672   \setlength{\slidelogoheight}{1cm}
5673 }
5674 \newsavebox{\slidelogo}
5675 \sbox{\slidelogo}{\TeX}
5676 \newrobustcmd{\setslidelogo}[1]{
5677   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
5678 }
```

(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.

```

5679 \def\source{Michael Kohlhase}% customize locally
5680 \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.

```

5681 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
5682 \newsavebox{\cclogo}
5683 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
5684 \newif\ifcchref\cchreffalse
5685 \AtBeginDocument{
5686   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
5687 }
5688 \def\licensing{
5689   \ifcchref
5690     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
5691   \else
5692     {\usebox{\cclogo}}
5693   \fi
5694 }
5695 \newrobustcmd{\setlicensing}[2][]{
5696   \def@url{#1}
5697   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
5698   \ifx@url@empty
5699     \def\licensing{{\usebox{\cclogo}}}
5700   \else
5701     \def\licensing{
```

```

5702     \ifcchref
5703     \href{#1}{\usebox{\cclogo}}
5704     \else
5705     {\usebox{\cclogo}}
5706     \fi
5707   }
5708 \fi
5709 }

```

(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

```

5710 \newrobustcmd\miko@slidelabel{
5711   \vbox to \slidelogoheight{
5712     \vss\hbox to \slidewidth
5713     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
5714   }
5715 }

```

(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.

```

5716 \def\Gin@mhrepos{}
5717 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
5718 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
5719 \newrobustcmd\frameimage[2][]{
5720   \stepcounter{slide}
5721   \bool_if:NT \c__notesslides_frameimages_bool {
5722     \def\Gin@ewidth{}\setkeys{Gin}{#1}
5723     \bool_if:NF \c__notesslides_notes_bool { \vfill }
5724     \begin{center}
5725       \bool_if:NTF \c__notesslides_fiboxed_bool {
5726         \fbox{
5727           \ifx\Gin@ewidth\@empty
5728             \ifx\Gin@mhrepos\@empty
5729               \mhgraphics[width=\slidewidth,#1]{#2}
5730             \else
5731               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5732             \fi
5733           \else% \Gin@ewidth empty
5734             \ifx\Gin@mhrepos\@empty
5735               \mhgraphics[#1]{#2}
5736             \else
5737               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5738             \fi
5739           \fi% \Gin@ewidth empty
5740         }
5741       }{
5742         \ifx\Gin@ewidth\@empty

```

²²EdNOTE: see that we can use the themes for the slides some day. This is all fake.


```

5743         \ifx\Gin@mhrepos\empty
5744             \mhgraphics[width=\slidewidth,#1]{#2}
5745         \else
5746             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
5747         \fi
5748         \ifx\Gin@mhrepos\empty
5749             \mhgraphics[#1]{#2}
5750         \else
5751             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
5752         \fi
5753     \fi% Gin@ewidth empty
5754 }
5755 \end{center}
5756 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
5757 \bool_if:NF \c__notesslides_notes_bool { \vfill }
5758 }
5759 } % 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.

```

5760 \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

```

5761 \AddToHook{begindocument}{
5762     \definecolor{green}{rgb}{0,.5,0}
5763     \definecolor{purple}{cmk}{.3,1,0,.17}
5764 }

```

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`.

```

5765 % \def\STpresent#1{\textcolor{blue}{#1}}
5766 \def\defemph#1{\textcolor{magenta}{#1}}
5767 \def\symrefemph#1{\textcolor{cyan}{#1}}
5768 \def\compemph#1{\textcolor{blue}{#1}}
5769 \def\titleemph#1{\textcolor{blue}{#1}}
5770 \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.

```

5771 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
5772 \def\smalltextwarning{
5773     \pgfuseimage{miko@small@dbend}
5774     \xspace
5775 }
5776 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

5777 \newrobustcmd\textwarning{
5778   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
5779   \xspace
5780 }
5781 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
5782 \newrobustcmd\bigtextwarning{
5783   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
5784   \xspace
5785 }

```

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

```

5786 \newrobustcmd\putgraphicsat[3]{
5787   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
5788 }
5789 \newrobustcmd\putat[2]{
5790   \begin{picture}(0,0)\put(#1){#2}\end{picture}
5791 }

```

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`.

```

5792 \bool_if:NT \c__notesslides_sectocframes_bool {
5793   \str_if_eq:VnTF \__notesslidesstopsect{part}{
5794     \newcounter{chapter}\counterwithin*{section}{chapter}
5795   }{
5796     \str_if_eq:VnT\__notesslidesstopsect{chapter}{
5797       \newcounter{chapter}\counterwithin*{section}{chapter}
5798     }
5799   }
5800 }

```

`\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
5801 \def\part@prefix{}
5802 \@ifpackageloaded{document-structure}{}{
5803   \str_case:VnF \__notesslidesstopsect {
5804     {part}{
5805       \int_set:Nn \l_document_structure_section_level_int {0}
5806       \def\thesection{\arabic{chapter}.\arabic{section}}
5807       \def\part@prefix{\arabic{chapter}.}
5808     }
5809     {chapter}{
5810       \int_set:Nn \l_document_structure_section_level_int {1}
5811       \def\thesection{\arabic{chapter}.\arabic{section}}
5812       \def\part@prefix{\arabic{chapter}.}
5813     }
5814   }{
5815     \int_set:Nn \l_document_structure_section_level_int {2}
5816     \def\part@prefix{}

```

```

5817 }
5818 }
5819
5820 \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

```

5821 \renewenvironment{omgroup}[2][]{
5822   \__document_structure_omgroup_args:n { #1 }
5823   \int_incr:N \l_document_structure_section_level_int
5824   \bool_if:NT \c__notesslides_sectocframes_bool {
5825     \stepcounter{slide}
5826     \begin{frame}[noframenumbering]
5827       \vfill\Large\centering
5828       \red{
5829         \ifcase\l_document_structure_section_level_int\or
5830           \stepcounter{part}
5831           \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
5832           \def\currentsectionlevel{\omdoc@part@kw}
5833         \or
5834           \stepcounter{chapter}
5835           \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
5836           \def\currentsectionlevel{\omdoc@chapter@kw}
5837         \or
5838           \stepcounter{section}
5839           \def\__notesslideslabel{\part@prefix\arabic{section}}
5840           \def\currentsectionlevel{\omdoc@section@kw}
5841         \or
5842           \stepcounter{subsection}
5843           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
5844           \def\currentsectionlevel{\omdoc@subsection@kw}
5845         \or
5846           \stepcounter{subsubsection}
5847           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
5848           \def\currentsectionlevel{\omdoc@subsubsection@kw}
5849         \or
5850           \stepcounter{paragraph}
5851           \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}.\arabic{paragraph}}
5852           \def\currentsectionlevel{\omdoc@paragraph@kw}
5853         \else
5854           \def\__notesslideslabel{}
5855           \def\currentsectionlevel{\omdoc@paragraph@kw}
5856         \fi% end ifcase
5857         \__notesslideslabel%\sref@label@id\__notesslideslabel
5858         \quad #2%
5859       }%
5860       \vfill%
5861     \end{frame}%
5862   }
5863   \str_if_empty:NF \l__document_structure_omgroup_id_str {
5864     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str

```

```

5865     }
5866   }{}
5867 }

```

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

```

5868 \def\inserttheorembodyfont{\normalfont}
5869 %\bool_if:NF \c__notesslides_notes_bool {
5870 %   \defbeamertemplate{theorem begin}{miko}
5871 %   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
5872 %     \ifx\inserttheoremaddition@empty\else\ (\inserttheoremaddition)\fi%
5873 %     \inserttheorempunctuation\inserttheorembodyfont\hspace}
5874 %   \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

5875 % \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.

```

5876 % \expandafter\def\csname Parent2\endcsname{}
5877 %}
5878
5879 \AddToHook{begindocument}{ % this does not work for some reason
5880   \setbeamertemplate{theorems}[ams style]
5881 }
5882 \bool_if:NT \c__notesslides_notes_bool {
5883   \renewenvironment{columns}[1][]{%
5884     \par\noindent%
5885     \begin{minipage}%
5886       \slidewidth\centering\leavevmode%
5887   }{%
5888     \end{minipage}\par\noindent%
5889   }%
5890   \newsavebox\columnbox%
5891   \renewenvironment<>{column}[2][]{%
5892     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
5893   }{%
5894     \end{minipage}\end{lrbox}\usebox\columnbox%
5895   }%
5896 }
5897 \bool_if:NTF \c__notesslides_noproblems_bool {
5898   \newenvironment{problems}{}{}
5899 }{
5900   \excludacomment{problems}
5901 }

```

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.

```

5902 \gdef\printexcursions{}
5903 \newcommand\excursionref[2]{% label, text

```

```

5904 \bool_if:NT \c__notesslides_notes_bool {
5905   \begin{sparagraph}[title=Excursion]
5906     #2 \sref[fallback=the appendix]{#1}.
5907   \end{sparagraph}
5908 }
5909 }
5910 \newcommand\activate@excursion[2][]{
5911   \gappto\printexcursions{\inputref{#1}{#2}}
5912 }
5913 \newcommand\excursion[4][]{% repos, label, path, text
5914   \bool_if:NT \c__notesslides_notes_bool {
5915     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
5916   }
5917 }

```

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

\excursiongroup

```

5918 \keys_define:nn{notesslides / excursiongroup }{
5919   id          .str_set_x:N = \l__notesslides_excursion_id_str,
5920   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
5921   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
5922 }
5923 \cs_new_protected:Nn \__notesslides_excursion_args:n {
5924   \tl_clear:N \l__notesslides_excursion_intro_tl
5925   \str_clear:N \l__notesslides_excursion_id_str
5926   \str_clear:N \l__notesslides_excursion_mhrepos_str
5927   \keys_set:nn {notesslides / excursiongroup }{ #1 }
5928 }
5929 \newcommand\excursiongroup[1][]{
5930   \__notesslides_excursion_args:n{ #1 }
5931   \ifdefempty\printexcursions{}% only if there are excursions
5932   {\begin{note}
5933     \begin{omgroup}[#1]{Excursions}%
5934     \ifdefempty\l__notesslides_excursion_intro_tl{\
5935       \inputref[\l__notesslides_excursion_mhrepos_str]{
5936         \l__notesslides_excursion_intro_tl
5937       }
5938     }
5939     \printexcursions%
5940     \end{omgroup}
5941     \end{note}}
5942 }
5943 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
5944 \</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.

```
5945 <*package>
5946 <@@=problems>
5947 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
5948 \RequirePackage{l3keys2e,expl-keystr-compatible}
5949
5950 \keys_define:nn { problem / pkg }{
5951   notes      .default:n   = { true },
5952   notes      .bool_set:N  = \c__problems_notes_bool,
5953   gnotes     .default:n   = { true },
5954   gnotes     .bool_set:N  = \c__problems_gnotes_bool,
5955   hints      .default:n   = { true },
5956   hints      .bool_set:N  = \c__problems_hints_bool,
5957   solutions  .default:n   = { true },
5958   solutions  .bool_set:N  = \c__problems_solutions_bool,
5959   pts        .default:n   = { true },
5960   pts        .bool_set:N  = \c__problems_pts_bool,
5961   min        .default:n   = { true },
5962   min        .bool_set:N  = \c__problems_min_bool,
5963   boxed      .default:n   = { true },
5964   boxed      .bool_set:N  = \c__problems_boxed_bool,
5965   unknown    .code:n      = {}
5966 }
5967 \newif\ifsolutions
5968
5969 \ProcessKeysOptions{ problem / pkg }
5970 \bool_if:NTF \c__problems_solutions_bool {
5971   \solutionstrue
5972 }{
5973   \solutionsfalse
5974 }
```

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

```
5975 \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.

```
5976 \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.

```
5977 \def\prob@problem@kw{Problem}
5978 \def\prob@solution@kw{Solution}
5979 \def\prob@hint@kw{Hint}
5980 \def\prob@note@kw{Note}
5981 \def\prob@gnote@kw{Grading}
5982 \def\prob@pt@kw{pt}
5983 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
5984 \AddToHook{begindocument}{
5985   \ltx@ifpackageloaded{babel}{
5986     \makeatletter
5987     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5988     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5989       \input{problem-ngerman.ldf}
5990     }
5991     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5992       \input{problem-finnish.ldf}
5993     }
5994     \clist_if_in:NnT \l_tmpa_clist {french}{
5995       \input{problem-french.ldf}
5996     }
5997     \clist_if_in:NnT \l_tmpa_clist {russian}{
5998       \input{problem-russian.ldf}
5999     }
6000     \makeatother
6001   }{ }
6002 }
```

40.2 Problems and Solutions

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

```
6003 \keys_define:nn{ problem / problem }{
6004   id      .str_set:x:N = \l__problems_prob_id_str,
6005   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6006   min     .tl_set:N    = \l__problems_prob_min_tl,
6007   title   .tl_set:N    = \l__problems_prob_title_tl,
6008   type    .tl_set:N    = \l__problems_prob_type_tl,
6009   refnum  .int_set:N    = \l__problems_prob_refnum_int
6010 }
6011 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6012 \str_clear:N \l__problems_prob_id_str
6013 \tl_clear:N \l__problems_prob_pts_tl
6014 \tl_clear:N \l__problems_prob_min_tl
6015 \tl_clear:N \l__problems_prob_title_tl
6016 \tl_clear:N \l__problems_prob_type_tl
6017 \int_zero_new:N \l__problems_prob_refnum_int
6018 \keys_set:nn { problem / problem }{ #1 }
6019 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6020   \let\l__problems_prob_refnum_int\undefined
6021 }
6022 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6023 \newcounter{problem}
6024 \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.

```

6025 \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

```

6026 \newcommand\prob@number{
6027   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6028     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6029   }{
6030     \int_if_exist:NTF \l__problems_prob_refnum_int {
6031       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6032     }{
6033       \prob@label\theproblem
6034     }
6035   }
6036 }

```

(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.

```

6037 \newcommand\prob@title[3]{%
6038   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6039     #2 \l__problems_inclprob_title_tl #3
6040   }{
6041     \tl_if_exist:NTF \l__problems_prob_title_tl {
6042       #2 \l__problems_prob_title_tl #3
6043     }{
6044       #1
6045     }
6046   }
6047 }

```


(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.

```

6048 \def\prob@heading{
6049   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6050   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
6051 }

```

(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

```

6052 \newenvironment{sproblem}[1][{}]{
6053   \__problems_prob_args:n{#1}%\sref@target%
6054   \@in@omtexttrue% we are in a statement (for inline definitions)
6055   \stepcounter{problem}\record@problem
6056   \def\current@section@level{\prob@problem@kw}
6057   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6058     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6059   }{
6060     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6061   }
6062   \str_if_exist:NTF \l__problems_inclprob_id_str {
6063     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6064   }{
6065     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6066   }
6067
6068
6069   \clist_set:No \l_tmpa_clist \sproblemtype
6070   \tl_clear:N \l_tmpa_tl
6071   \clist_map_inline:Nn \l_tmpa_clist {
6072     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6073       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6074     }
6075   }
6076   \tl_if_empty:NTF \l_tmpa_tl {
6077     \__problems_sproblem_start:
6078   }{
6079     \l_tmpa_tl
6080   }
6081   \stex_ref_new_doc_target:n \sproblemid
6082 }{
6083   \clist_set:No \l_tmpa_clist \sproblemtype
6084   \tl_clear:N \l_tmpa_tl
6085   \clist_map_inline:Nn \l_tmpa_clist {
6086     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6087       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6088     }
6089   }

```

```

6089 }
6090 \tl_if_empty:NTF \l_tmpa_tl {
6091   \__problems_sproblem_end:
6092 }{
6093   \l_tmpa_tl
6094 }
6095
6096
6097 \smallskip
6098 }
6099
6100
6101 \cs_new_protected:Nn \__problems_sproblem_start: {
6102   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6103 }
6104 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6105
6106 \newcommand\stexpatchproblem[3][] {
6107   \str_set:Nx \l_tmpa_str{ #1 }
6108   \str_if_empty:NTF \l_tmpa_str {
6109     \tl_set:Nn \__problems_sproblem_start: { #2 }
6110     \tl_set:Nn \__problems_sproblem_end: { #3 }
6111   }{
6112     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6113     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6114   }
6115 }
6116
6117
6118 \bool_if:NT \c__problems_boxed_bool {
6119   \surroundwithmdframed{problem}
6120 }

```

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

```

6121 \def\record@problem{
6122   \protected@write\@auxout{}
6123   {
6124     \string\@problem{\prob@number}
6125     {
6126       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6127         \l__problems_inclprob_pts_tl
6128       }{
6129         \l__problems_prob_pts_tl
6130       }
6131     }%
6132     {
6133       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6134         \l__problems_inclprob_min_tl
6135       }{
6136         \l__problems_prob_min_tl
6137       }
6138     }
6139   }
6140 }

```

(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).

```
6141 \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.

```
6142 \keys_define:nn { problem / solution }{
6143   id                .str_set_x:N = \l__problems_solution_id_str ,
6144   for               .tl_set:N   = \l__problems_solution_for_tl ,
6145   height           .dim_set:N   = \l__problems_solution_height_dim ,
6146   creators         .clist_set:N = \l__problems_solution_creators_clist ,
6147   contributors     .clist_set:N = \l__problems_solution_contributors_clist ,
6148   srccite          .tl_set:N    = \l__problems_solution_srccite_tl
6149 }
6150 \cs_new_protected:Nn \__problems_solution_args:n {
6151   \str_clear:N \l__problems_solution_id_str
6152   \tl_clear:N \l__problems_solution_for_tl
6153   \tl_clear:N \l__problems_solution_srccite_tl
6154   \clist_clear:N \l__problems_solution_creators_clist
6155   \clist_clear:N \l__problems_solution_contributors_clist
6156   \dim_zero:N \l__problems_solution_height_dim
6157   \keys_set:nn { problem / solution }{ #1 }
6158 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
6159 \newcommand\@startsolution[1][{}]{
6160   \__problems_solution_args:n { #1 }
6161   \@in@omtexttrue% we are in a statement.
6162   \bool_if:NF \c__problems_boxed_bool { \hrule }
6163   \smallskip\noindent
6164   {\textbf\prob@solution@kw : \enspace}
6165   \begin{small}
6166   \def\current@section@level{\prob@solution@kw}
6167   \ignorespacesandpars
6168 }
```

\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.

```
6169 \newcommand\startsolutions{
6170   \specialcomment{solution}{\@startsolution}{
6171     \bool_if:NF \c__problems_boxed_bool {
6172       \hrule\medskip
6173     }
6174     \end{small}%
6175   }
6176   \bool_if:NT \c__problems_boxed_bool {
6177     \surroundwithmdframed{solution}
6178   }
6179 }
```

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

\stopsolutions

```
6180 \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.

```
6181 \ifsolutions
6182   \startsolutions
6183 \else
6184   \stopsolutions
6185 \fi
```

exnote

```
6186 \bool_if:NTF \c__problems_notes_bool {
6187   \newenvironment{exnote}[1][]{
6188     \par\smallskip\hrule\smallskip
6189     \noindent\textbf{\prob@note@kw : }\small
6190   }{
6191     \smallskip\hrule
6192   }
6193 }{
6194   \excludecomment{exnote}
6195 }
```

hint

```
6196 \bool_if:NTF \c__problems_notes_bool {
6197   \newenvironment{hint}[1][]{
6198     \par\smallskip\hrule\smallskip
6199     \noindent\textbf{\prob@hint@kw :~ }\small
6200   }{
6201     \smallskip\hrule
6202   }
6203   \newenvironment{exhint}[1][]{
6204     \par\smallskip\hrule\smallskip
6205     \noindent\textbf{\prob@hint@kw :~ }\small
6206   }{
6207     \smallskip\hrule
6208   }
6209 }{
6210   \excludecomment{hint}
6211   \excludecomment{exhint}
6212 }
```

gnote

```
6213 \bool_if:NTF \c__problems_notes_bool {
6214   \newenvironment{gnote}[1][]{
6215     \par\smallskip\hrule\smallskip
6216     \noindent\textbf{\prob@gnote@kw : }\small
6217   }{
6218     \smallskip\hrule
6219   }
6220 }{
6221   \excludecomment{gnote}
6222 }
```

40.3 Multiple Choice Blocks

EdN:23

mcb 23

```
6223 \newenvironment{mcb}{
6224   \begin{enumerate}
6225 }{
6226   \end{enumerate}
6227 }
```

we define the keys for the mcc macro

```
6228 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
6229   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
6230     \bool_set_true:N #1
6231   }{
6232     \bool_set_false:N #1
6233   }
6234 }
6235 \keys_define:nn { problem / mcc }{
6236   id          .str_set:x:N = \l__problems_mcc_id_str ,
6237   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
6238   T           .default:n   = { true } ,
6239   T           .bool_set:N   = \l__problems_mcc_t_bool ,
6240   F           .default:n   = { true } ,
6241   F           .bool_set:N   = \l__problems_mcc_f_bool ,
6242   Ttext       .code:n      = {
6243     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
6244   } ,
6245   Ftext       .code:n      = {
6246     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
6247   }
6248 }
6249 \cs_new_protected:Nn \l__problems_mcc_args:n {
6250   \str_clear:N \l__problems_mcc_id_str
6251   \tl_clear:N \l__problems_mcc_feedback_tl
6252   \bool_set_true:N \l__problems_mcc_t_bool
6253   \bool_set_true:N \l__problems_mcc_f_bool
6254   \bool_set_true:N \l__problems_mcc_Ttext_bool
6255   \bool_set_false:N \l__problems_mcc_Ftext_bool
6256   \keys_set:nn { problem / mcc }{ #1 }
6257 }
```

\mcc

```
6258 \newcommand\mcc[2][] {
6259   \l__problems_mcc_args:n{ #1 }
6260   \item #2
6261   \ifsolutions
6262     \\\
6263     \bool_if:NT \l__problems_mcc_t_bool {
6264       % TODO!
6265       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
6266     }
6267     \bool_if:NT \l__problems_mcc_f_bool {
```

²³EdNOTE: MK: maybe import something better here from a dedicated MC package

```

6268      % TODO!
6269      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
6270    }
6271    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
6272      !
6273    }{
6274      \l__problems_mcc_feedback_tl
6275    }
6276    \fi
6277  } %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.

```

6278
6279 \keys_define:nn{ problem / inclproblem }{
6280   id      .str_set:N = \l__problems_inclprob_id_str,
6281   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
6282   min     .tl_set:N  = \l__problems_inclprob_min_tl,
6283   title   .tl_set:N  = \l__problems_inclprob_title_tl,
6284   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
6285   type    .tl_set:N  = \l__problems_inclprob_type_tl,
6286   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
6287 }
6288 \cs_new_protected:Nn \l__problems_inclprob_args:n {
6289   \str_clear:N \l__problems_prob_id_str
6290   \tl_clear:N \l__problems_inclprob_pts_tl
6291   \tl_clear:N \l__problems_inclprob_min_tl
6292   \tl_clear:N \l__problems_inclprob_title_tl
6293   \tl_clear:N \l__problems_inclprob_type_tl
6294   \int_zero_new:N \l__problems_inclprob_refnum_int
6295   \str_clear:N \l__problems_inclprob_mhrepos_str
6296   \keys_set:nn { problem / inclproblem }{ #1 }
6297   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
6298     \let\l__problems_inclprob_pts_tl\undefined
6299   }
6300   \tl_if_empty:NT \l__problems_inclprob_min_tl {
6301     \let\l__problems_inclprob_min_tl\undefined
6302   }
6303   \tl_if_empty:NT \l__problems_inclprob_title_tl {
6304     \let\l__problems_inclprob_title_tl\undefined
6305   }
6306   \tl_if_empty:NT \l__problems_inclprob_type_tl {
6307     \let\l__problems_inclprob_type_tl\undefined
6308   }
6309   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
6310     \let\l__problems_inclprob_refnum_int\undefined
6311   }
6312 }

```

```

6313
6314 \cs_new_protected:Nn \__problems_inclprob_clear: {
6315   \let\l__problems_inclprob_id_str\undefined
6316   \let\l__problems_inclprob_pts_tl\undefined
6317   \let\l__problems_inclprob_min_tl\undefined
6318   \let\l__problems_inclprob_title_tl\undefined
6319   \let\l__problems_inclprob_type_tl\undefined
6320   \let\l__problems_inclprob_refnum_int\undefined
6321   \let\l__problems_inclprob_mhrepos_str\undefined
6322 }
6323 \__problems_inclprob_clear:
6324
6325 \newcommand\includeproblem[2][ ]{
6326   \__problems_inclprob_args:n{ #1 }
6327   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
6328     \input{#2}
6329   }{
6330     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
6331       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
6332     }
6333   }
6334   \__problems_inclprob_clear:
6335 }

```

(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.

```

6336 \AddToHook{enddocument}{
6337   \bool_if:NT \c__problems_pts_bool {
6338     \message{Total:~\arabic{pts}~points}
6339   }
6340   \bool_if:NT \c__problems_min_bool {
6341     \message{Total:~\arabic{min}~minutes}
6342   }
6343 }

```

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

```

6344 \def\pts#1{
6345   \bool_if:NT \c__problems_pts_bool {
6346     \marginpar{#1~\prob@pt@kw}
6347   }
6348 }
6349 \def\min#1{
6350   \bool_if:NT \c__problems_min_bool {
6351     \marginpar{#1~\prob@min@kw}
6352   }
6353 }

```

`\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.

```

6354 \newcounter{pts}
6355 \def\show@pts{
6356   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6357     \bool_if:NT \c__problems_pts_bool {
6358       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
6359       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
6360     }
6361   }{
6362     \tl_if_exist:NT \l__problems_prob_pts_tl {
6363       \bool_if:NT \c__problems_pts_bool {
6364         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
6365         \addtocounter{pts}{\l__problems_prob_pts_tl}
6366       }
6367     }
6368   }
6369 }

```

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

`\show@min`

```

6370 \newcounter{min}
6371 \def\show@min{
6372   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6373     \bool_if:NT \c__problems_min_bool {
6374       \marginpar{\l__problems_inclprob_min_tl\ min}
6375       \addtocounter{min}{\l__problems_inclprob_min_tl}
6376     }
6377   }{
6378     \tl_if_exist:NT \l__problems_prob_min_tl {
6379       \bool_if:NT \c__problems_min_bool {
6380         \marginpar{\l__problems_prob_min_tl\ min}
6381         \addtocounter{min}{\l__problems_prob_min_tl}
6382       }
6383     }
6384   }
6385 }
6386 \</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.

```
6387 <@@=hwexam>
6388 <*cls>
6389 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6390 \RequirePackage{l3keys2e,expl-keystr-compatible}
6391 \DeclareOption*{
6392   \PassOptionsToClass{\CurrentOption}{document-structure}
6393   \PassOptionsToPackage{\CurrentOption}{stex}
6394   \PassOptionsToPackage{\CurrentOption}{hwexam}
6395   \PassOptionsToPackage{\CurrentOption}{tikzinput}
6396 }
6397 \ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the L^AT_EXML bindings, we make sure the right packages are loaded.

```
6398 \LoadClass{document-structure}
6399 \RequirePackage{stex}
6400 \RequirePackage{hwexam}
6401 \RequirePackage{tikzinput}
6402 \RequirePackage{graphicx}
6403 \RequirePackage{a4wide}
6404 \RequirePackage{amssymb}
6405 \RequirePackage{amstext}
6406 \RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors

```

6407 \newcommand\assig@default@type{\hwexam@assignment@kw}
6408 \def\document@hwexamtype{\assig@default@type}
6409 <@@=document_structure>
6410 \keys_define:nn { document-structure / document }{
6411 id .str_set_x:N = \c_document_structure_document_id_str,
6412 hwexamtype .tl_set:N = \document@hwexamtype
6413 }
6414 <@@=hwexam>
6415 </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.

```
6416 \*package>
6417 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
6418 \RequirePackage{l3keys2e,expl-keystr-compat}
6419
6420 \newif\iftest\testfalse
6421 \DeclareOption{test}{\testtrue}
6422 \newif\ifmultiple\multiplefalse
6423 \DeclareOption{multiple}{\multipletrue}
6424 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
6425 \ProcessOptions
```

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

```
6426 \RequirePackage{keyval}[1997/11/10]
6427 \RequirePackage{problem}
```

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

```
6428 \newcommand\hwexam@assignment@kw{Assignment}
6429 \newcommand\hwexam@given@kw{Given}
6430 \newcommand\hwexam@due@kw{Due}
6431 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
6432 blank~for~extra~space}
6433 \def\hwexam@minutes@kw{minutes}
6434 \newcommand\correction@probs@kw{prob.}
6435 \newcommand\correction@pts@kw{total}
6436 \newcommand\correction@reached@kw{reached}
6437 \newcommand\correction@sum@kw{Sum}
6438 \newcommand\correction@grade@kw{grade}
6439 \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

```

6440 \AddToHook{begindocument}{
6441 \ltx@ifpackageloaded{babel}{
6442 \makeatletter
6443 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6444 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6445 \input{hwexam-ngerman.ldf}
6446 }
6447 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6448 \input{hwexam-finnish.ldf}
6449 }
6450 \clist_if_in:NnT \l_tmpa_clist {french}{
6451 \input{hwexam-french.ldf}
6452 }
6453 \clist_if_in:NnT \l_tmpa_clist {russian}{
6454 \input{hwexam-russian.ldf}
6455 }
6456 \makeatother
6457 }{}
6458 }
6459

```

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.

```

6460 \newcounter{assignment}
6461 \numberproblemsin{assignment}
6462 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

6463 \keys_define:nn { hwexam / assignment } {
6464 id .str_set:N = \l__hwexam_assign_id_str,
6465 number .int_set:N = \l__hwexam_assign_number_int,
6466 title .tl_set:N = \l__hwexam_assign_title_tl,
6467 type .tl_set:N = \l__hwexam_assign_type_tl,
6468 given .tl_set:N = \l__hwexam_assign_given_tl,
6469 due .tl_set:N = \l__hwexam_assign_due_tl,
6470 loadmodules .code:n = {
6471 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
6472 }
6473 }
6474 \cs_new_protected:Nn \__hwexam_assignment_args:n {
6475 \str_clear:N \l__hwexam_assign_id_str
6476 \int_set:Nn \l__hwexam_assign_number_int {-1}
6477 \tl_clear:N \l__hwexam_assign_title_tl
6478 \tl_clear:N \l__hwexam_assign_type_tl
6479 \tl_clear:N \l__hwexam_assign_given_tl
6480 \tl_clear:N \l__hwexam_assign_due_tl
6481 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

6482 \keys_set:nn { hwexam / assignment }{ #1 }
6483 }

```

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.

```

6484 \newcommand\given@due[2]{
6485 \bool_lazy_all:nF {
6486 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6487 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6488 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6489 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6490 }{ #1 }
6491
6492 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
6493 \tl_if_empty:NF \l__hwexam_assign_given_tl {
6494 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
6495 }
6496 }{
6497 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
6498 }
6499
6500 \bool_lazy_or:nnF {
6501 \bool_lazy_and_p:nn {
6502 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6503 }{
6504 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6505 }
6506 }{
6507 \bool_lazy_and_p:nn {
6508 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
6509 }{
6510 \tl_if_empty_p:V \l__hwexam_assign_due_tl
6511 }
6512 }{ ,~ }
6513
6514 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
6515 \tl_if_empty:NF \l__hwexam_assign_due_tl {
6516 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
6517 }
6518 }{
6519 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
6520 }
6521
6522 \bool_lazy_all:nF {
6523 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
6524 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
6525 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
6526 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
6527 }{ #2 }
6528 }

```

`\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.

```

6529 \newcommand\assignment@title[3]{
6530 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
6531 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
6532 #1
6533 }{
6534 #2\l__hwexam_assign_title_tl#3
6535 }
6536 }{
6537 #2\l__hwexam_inclasssign_title_tl#3
6538 }
6539 }

```

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

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

```

6540 \newcommand\assignment@number{
6541 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
6542 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6543 \arabic{assignment}
6544 } {
6545 \int_use:N \l__hwexam_assign_number_int
6546 }
6547 }{
6548 \int_use:N \l__hwexam_inclasssign_number_int
6549 }
6550 }

```

(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.

```

6551 \newenvironment{assignment}[1][]{
6552 \__hwexam_assignment_args:n { #1 }
6553 %\sref@target
6554 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
6555 \global\stepcounter{assignment}
6556 }{
6557 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
6558 }
6559 \setcounter{problem}{0}
6560 \def\current@section@level{\document@hwexamtype}
6561 %\sref@label@id{\document@hwexamtype \thesection}
6562 \begin{@assignment}
6563 }{
6564 \end{@assignment}
6565 }

```

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

```

6566 \def\ass@title{
6567 \protect\document@hwexamtype~\arabic{assignment}
6568 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
6569 }
6570 \ifmultiple
6571 \newenvironment{@assignment}{
6572 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
6573 \begin{omgroup}[loadmodules]{\ass@title}
6574 }{
6575 \begin{omgroup}{\ass@title}
6576 }
6577 }{
6578 \end{omgroup}
6579 }

```

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

```

6580 \else
6581 \newenvironment{@assignment}{
6582 \begin{center}\bf
6583 \Large@title\strut\\
6584 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}\{;\}\{;\}
6585 \large\given@due{--;\}\{;\}--}
6586 \end{center}
6587 }{}
6588 \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 `inclassig` keys after the input.

```

6589 \keys_define:nn { hwexam / inclassignment } {
6590 %id .str_set_x:N = \l__hwexam_assign_id_str,
6591 number .int_set:N = \l__hwexam_inclassign_number_int,
6592 title .tl_set:N = \l__hwexam_inclassign_title_tl,
6593 type .tl_set:N = \l__hwexam_inclassign_type_tl,
6594 given .tl_set:N = \l__hwexam_inclassign_given_tl,
6595 due .tl_set:N = \l__hwexam_inclassign_due_tl,
6596 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
6597 }
6598 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
6599 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
6600 \tl_clear:N \l__hwexam_inclassign_title_tl
6601 \tl_clear:N \l__hwexam_inclassign_type_tl
6602 \tl_clear:N \l__hwexam_inclassign_given_tl
6603 \tl_clear:N \l__hwexam_inclassign_due_tl
6604 \str_clear:N \l__hwexam_inclassign_mhrepos_str
6605 \keys_set:nn { hwexam / inclassignment }{ #1 }
6606 }
6607 \__hwexam_inclassignment_args:n {}
6608
6609 \newcommand\inputassignment[2][ ]{

```

```

6610 \_hwexam_inclassnment_args:n { #1 }
6611 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
6612 \input{#2}
6613 }{
6614 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
6615 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
6616 }
6617 }
6618 \_hwexam_inclassnment_args:n {}
6619 }
6620 \newcommand\includeassignment[2][]{
6621 \newpage
6622 \inputassignment[#1]{#2}
6623 }

```

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

42.4 Typesetting Exams

\quizheading

```

6624 \ExplSyntaxOff
6625 \newcommand\quizheading[1]{%
6626 \def\@tas{#1}%
6627 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
6628 \ifx\@tas\@empty\else%
6629 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
6630 \fi%
6631 }
6632 \ExplSyntaxOn

```

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

\testheading

```

6633
6634 \def\hwexamheader{\input{hwexam-default.header}}
6635
6636 \def\hwexamminutes{
6637 \tl_if_empty:NTF \testheading@duration {
6638 {\testheading@min}~\hwexam@minutes@kw
6639 }{
6640 \testheading@duration
6641 }
6642 }
6643
6644 \keys_define:nn { hwexam / testheading } {
6645 min .tl_set:N = \testheading@min,
6646 duration .tl_set:N = \testheading@duration,
6647 reqpts .tl_set:N = \testheading@reqpts,
6648 tools .tl_set:N = \testheading@tools
6649 }
6650 \cs_new_protected:Nn \_hwexam_testheading_args:n {
6651 \tl_clear:N \testheading@min
6652 \tl_clear:N \testheading@duration

```



```

6653 \tl_clear:N \testheading@reqpts
6654 \tl_clear:N \testheading@tools
6655 \keys_set:nn { hwexam / testheading }{ #1 }
6656 }
6657 \newenvironment{testheading}[1][]{
6658   \__hwexam_testheading_args:n{ #1 }
6659   \newcount\check@time\check@time=\testheading@min
6660   \advance\check@time by -\theassignment@totalmin
6661   \newif\if@bonuspoints
6662   \tl_if_empty:NTF \testheading@reqpts {
6663     \@bonuspointsfalse
6664   }{
6665     \newcount\bonus@pts
6666     \bonus@pts=\theassignment@totalpts
6667     \advance\bonus@pts by -\testheading@reqpts
6668     \edef\bonus@pts{\the\bonus@pts}
6669     \@bonuspointstrue
6670   }
6671   \edef\check@time{\the\check@time}
6672
6673   \makeatletter\hwexamheader\makeatother
6674 }{
6675   \newpage
6676 }

```

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

\testspace

```

6677 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

6678 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

6679 \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.

```

6680 <@=problems>
6681 \renewcommand\@problem[3]{
6682   \stepcounter{assignment@probs}
6683   \def\__problemspts{#2}
6684   \ifx\__problemspts\@empty\else
6685     \addtocounter{assignment@totalpts}{#2}
6686   \fi
6687   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
6688   \xdef\correction@probs{\correction@probs & #1}%
6689   \xdef\correction@pts{\correction@pts & #2}
6690   \xdef\correction@reached{\correction@reached &}

```

```

6691 }
6692 <@@=hwexam>

```

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

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

```

6693 \newcounter{assignment@probs}
6694 \newcounter{assignment@totalpts}
6695 \newcounter{assignment@totalmin}
6696 \def\correction@probs{\correction@probs@kw}
6697 \def\correction@pts{\correction@pts@kw}
6698 \def\correction@reached{\correction@reached@kw}
6699 \stepcounter{assignment@probs}
6700 \newcommand\correction@table{
6701 \resizebox{\textwidth}{!}{%
6702 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
6703 &\multicolumn{\theassignment@probs}{c|}{}%|
6704 {\footnotesize\correction@forgrading@kw} &\\ \hline
6705 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
6706 \correction@pts & \theassignment@totalpts & \\ \hline
6707 \correction@reached & & \[.7cm]\hline
6708 \end{tabular}}
6709 </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}

```